p-ISSN 2407-7658 https://journal.unnes.ac.id/journals/sji/index e-ISSN 2460-0040

Sort Filter Skyline in Movie Recommendation Based on Individual Preferences: Performance and Time Complexity Analysis

Alvi Nur Fadhilah[1*], Triawan Adi Cahyanto[2], Ilham Saifudin[3]
[1,2,3]Department of Informatics Engineering, Faculty of Engineering, Universitas Muhammadiyah Jember, Indonesia

Abstract.
Purpose: This study seeks to deliver accurate, customized movie recommendations using the Sort Filter Skyline (SFS) algorithm. The approach considers factors like budget, box office earnings, popularity, runtime, and audience ratings to align closely with each user's specific preferences.
Methods: The Sort Filter Skyline (SFS) algorithm is employed, designed to identify and recommend items different from others within the dataset. Initially, the data undergoes preparation through pre-processing before being analyzed to compute entropy using the entropy formula. Before carrying out the dominance test, the SFS algorithm organizes the data based on entropy values.
Result: In this research, 176 skyline objects were identified from a dataset containing 4,803 movies, including well-known titles like "Avatar" and "Titanic." The Skyline Filter Sort (SFS) algorithm pinpointed these objects within 4 seconds. Additionally, evaluation results using synthetic data, as depicted in the data visualization, revealed that the number of attributes increased from 1 to 7. The dataset size grew, and the execution time also rose—from 18 seconds to 170 minutes. Despite this increase, the algorithm demonstrated efficient performance with optimized processing times.
Novelty: This study showcases the successful application of the SFS algorithm for generating personalized movie recommendations while tackling the difficulty of aligning viewer preferences with the extensive selection of films available. The findings offer important insights into enhancing recommendation systems by implementing algorithms efficiently and managing execution time complexity, contributing fresh perspectives to the field.

Keywords: Skyline query, Sort filter skyline, Recommendation system, Individual preference, Complexity analysis
Received July 2024 / **Revised** September 2024 / **Accepted** September 2024

INTRODUCTION

Amid the hustle and bustle of daily life, entertainment serves a crucial purpose in maintaining life's equilibrium, allowing the mind to take a break from constant work-related thoughts. As an artistic expression deeply rooted in human culture, film is a widely enjoyed form of entertainment. In Indonesia, the film industry began its journey in 1900, evolving significantly to become what it is today [1]. The growth of the contemporary film industry is characterized not only by a rise in production volume but also by the expansion of genres. In April alone, over 15 movies were screened in theaters, and annually, hundreds of films are launched, both in cinemas and across streaming platforms like Netflix, Disney Hotstar, and Viu [2]. The film industry is experiencing rapid growth, with new movies being produced nearly every day, leading to a constant stream of new releases almost every week. Films appeal to a wide range of audiences, spanning all age groups from young children to seniors. However, the abundance of movie options presents a challenge for viewers in selecting films that align with their personal tastes [3], [4]. For instance, a person might prefer films with substantial production budgets and high box office earnings, meaning recommendations cannot rely solely on ratings from other viewers. As a result, an algorithm is required to tailor movie suggestions based on individual preferences. Personalized recommendations ensure that everyone receives film suggestions that align closely with their unique tastes [5]. In this scenario, researchers employ the skyline algorithm to generate highly accurate recommendations, thereby enhancing the film viewing experience for all.

*Corresponding author.
Email addresses: alvinurf12@mail.com (Fadhilah)*, triawanac@unmuhjember.ac.id (Cahyanto), ilham.saifudin@unmuhjember.ac.id (Saifudin)
DOI: 10.15294/sji.v11i3.8474

The Skyline operator, introduced by [6], identifies a group of objects that are not dominated by others based on specific criteria. The goal of the skyline query algorithm is to find objects that align with a wide range of user preferences. This algorithm returns a set of skyline objects by filtering out those dominated by others within the dataset. One variant of the skyline query algorithm is the Sort Filter Skyline (SFS), which utilizes the entropy function to organize data and minimize dominance comparisons [7]. The Skyline algorithm is an evolution of the Block Nested Loops (BNL) algorithm, which involves lengthy iterations to search and compare each dataset value. SFS enhances this process by incorporating data sorting to streamline comparisons and reduce search time. This efficiency in handling large datasets provides a sense of reassurance and confidence in the SFS algorithm [8], [9], [10], [11].

Numerous studies have explored the application of the Sort-Filter-Skyline (SFS) algorithm for object selection and recommendation purposes. For instance, in research [12], SFS was applied to prioritize distributing personal protective equipment (PPE) in West Java Province. This study introduced a modified version of the SFS algorithm to enhance dominance measurement by incorporating a selection process for regions lacking hospitals. The comparison between two models, MS1 (standard SFS) and MS2 (modified SFS with the added selection process), revealed differences in the number of skyline objects produced and execution time, with MS2 demonstrating greater efficiency. Another study [13] proposed a novel algorithm, Attribute-Order-Preserving-Free-SFS (AOPF-SFS), to tackle the challenges of skyline query processing on encrypted Cloud data. AOPF-SFS builds on SFS by enabling query processing without preserving attribute order, introducing the eSkyline prototype system. The findings indicate that AOPF-SFS outperforms the original SFS's efficiency and effectiveness for encrypted data processing.

Our research stands apart from previous studies in both application and context. While [12] adapted the SFS algorithm for distributing personal protective equipment (PPE) and [13] developed AOPF-SFS for encrypted data handling, our study applies the SFS algorithm specifically to the realm of movie recommendations. We incorporate attributes such as budget, box office revenue, popularity, runtime, and audience ratings. Extensive data preprocessing and normalization were carried out, followed by testing the algorithm's execution time complexity using a synthetic dataset to analyze how variations in attributes and dataset size impact execution time.

The dataset utilized in this research was sourced from the TMDB movie database, available through Kaggle. This platform has provided movie and TV data since 2008 [14]. In this study, the researchers analyzed the TMDB dataset using Python programming to generate movie recommendations that are easily understandable by the public [15], [16], [17], [18], [19], [20], [21]. The complexity of data processing time, influenced by the number of attributes and the dataset size, significantly impacts the duration of processing within the SFS algorithm, as demonstrated by [22] in their research.

METHODS

Research methods involve structured and systematic approaches to collect valid data for scientific purposes. In this study, multiple stages are followed to achieve optimal outcomes. The process includes the following steps:

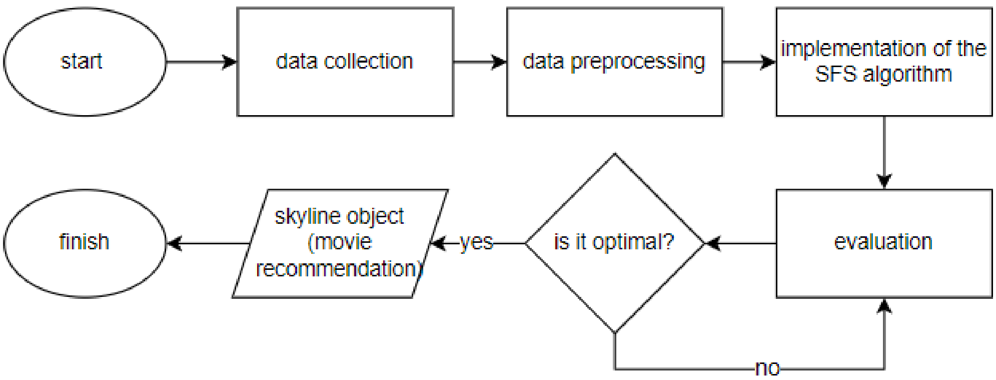


Figure 1. Stages of cPanel security system research

Data collection

The dataset utilized in this study is sourced from the Kaggle website [14], a platform owned by Google that offers public datasets for data science research. A total of 4,803 data entries were used for analysis. The critical attributes selected for the algorithm's calculations are as follows:

Table 1. Movie dataset attributes

No	Attribute	Data Type	Description
1.	id	integer	Unique movie ID used for identification
2.	revenue	integer	Total revenue generated by the movie
3.	popularity	integer	Measure of the movie's popularity among viewers
4.	budget	integer	Cost incurred to produce the movie
5.	runtime	float	Duration of the movie in minutes
6.	vote_average	float	Average rating score given by viewers
7.	vote_count	integer	Total number of votes received by the movie

The time complexity is evaluated using synthetic data of 50,000 independent, correlated, and anti-correlated entries. The implementation involves data normalization, entropy calculation, and dominance testing. This process ultimately identifies the skyline objects within the movie dataset.

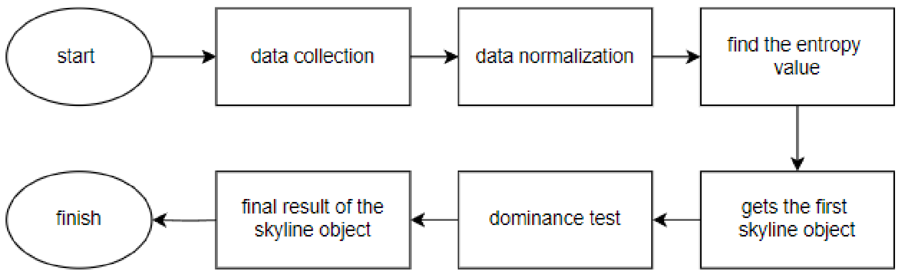


Figure 2. Implementation of the SFS algorithm [23]

Data preprocessing

The preprocessing phase uses Python functions to prepare the dataset before applying the SFS algorithm. Data preprocessing ensures that the dataset is high-quality and consistent [24]. The steps involved in this preprocessing stage include the following [25] Replacing missing or NaN values with zero, removing tables that are unrelated to the selected attributes, Converting columns into numeric data types. This process aims to prevent interruptions in the algorithm's execution caused by missing or NaN values or duplicate entries. Ensuring data quality is crucial for producing accurate and reliable analysis results.

Data normalization

Following the preprocessing stage, the attribute values of the dataset are normalized to bring them within a smaller, consistent range [26], such as from 0 to 1. Normalization is essential for ensuring that the entropy values of candidate objects fall within a suitable range, facilitating accurate comparison and selection [27]. The normalization technique involves multiplying each candidate object's attribute values by a factor, such as 0.0000000001, to ensure that the resulting entropy values remain between 0 and 1.

Entropy value calculation

The subsequent step involves calculating entropy values, which is necessary to identify the initial skyline object. The entropy is computed using the following Equation (1): [28]

$$E(t) = \sum_{i=1}^{k} \mathrm{In}(t[a_i] + 1) \qquad (1)$$

Where:

- E(t) is the entropy of object t
- a_i is the value of each attribute of object t
- ln is the natural logarithm

Once the entropy values are calculated, the objects are arranged in descending order based on their entropy scores. The object $t[a_i]$ with the highest entropy value will be designated as the first skyline object. The remaining objects will undergo a dominance test by being compared against the identified skyline objects.

If another object dominates object t[a_i], it will be removed. If not, it will be designated as the next skyline object.

Dominance testing

An object is deemed dominant if it matches or exceeds the quality of another object across all dimensions and surpasses it in at least one dimension [29]. The dominance test involves comparing objects across all attributes through the entire dataset. An object is considered dominant if it is superior in one or more attributes while being equal to or better than others in all remaining attributes.

Final skyline objects

Once the dominance testing is finished, the final set of skyline objects is extracted from the dataset. These objects represent the optimal selections, as other items across all defined dimensions do not outperform them. These skyline objects form the basis for the final recommendations provided to users.

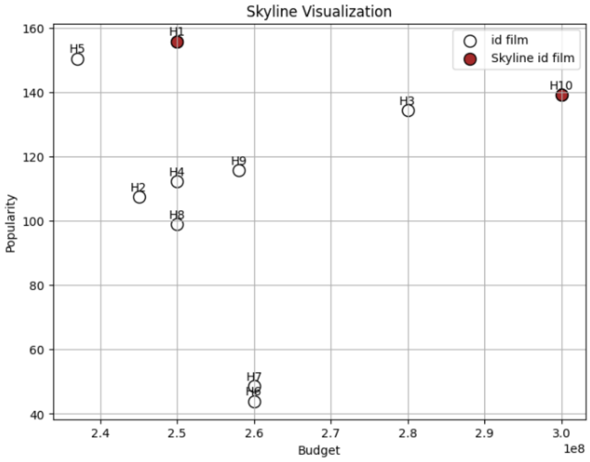


Figure 3. Illustration skyline

The figure above demonstrates the process of identifying skyline objects using two attributes: Budget (X-axis) and Popularity (Y-axis). Each point on the graph corresponds to a movie, with the red-marked points representing the skyline objects. These films are unique from any other in both budget and popularity.

White circles depict movies that are dominated by at least one skyline object. These films, for example, may have higher budgets but lower popularity or similar expenses with less popularity and thus are excluded from the skyline. Red circles indicate the final skyline objects—movies that offer the best combination of budget and popularity without being outperformed in either category. For instance, movie H1 is part of the skyline due to its highest popularity at a relatively lower budget. At the same time, H10 is included because it excels in both high budget and popularity.

RESULTS AND DISCUSSIONS

This study aims to generate movie recommendations by optimizing key attributes such as popularity, substantial revenue, large production budgets, and high audience ratings. The analysis follows the research process outlined in Figure 1, with detailed descriptions for each phase. The Sort Filter Skyline (SFS) algorithm, a key component of this study, is particularly effective in generating movie recommendations by considering these attributes and their interplay.

Data preprocessing

This phase outlines the utilization of multiple Python libraries for handling and processing the dataset. The following steps are undertaken:

Filling missing or NaN values with zero

Our approach to handling incomplete data within the dataset is meticulous, ensuring that any missing or NaN values are replaced with 0, as demonstrated in Tables 2 and 3. This attention to detail minimizes the potential for errors or inaccuracies in the analysis.

Table 2. Missing values

No	Column	Number of missing values
1.	Budget	0
2.	Genres	0
3.	homepage	3091
4.	Id	0
5.	Keywords	0
6.	Original_language	0
7.	Original_title	0
8.	Overview	3
9.	Popularity	0
10.	Production_companies	0
11.	Production_countries	0
12.	Release_date	1
13.	Revenue	0
14.	Runtime	2
15.	Spoken_languages	0
16.	Status	0
17.	Tagline	844
18.	title	0
19.	Vote_average	0
20.	Vote_count	0

Several attributes contained missing or NaN values, including homepage, overview, release_date, runtime, and tagline. Table 3 presents the outcome of replacing these missing values with 0. This substitution serves as a default for numeric attributes like runtime and release_date, signifying the lack of available data. By doing this, the model or algorithm can function without errors while maintaining the overall data distribution.

Table 3. Filling missing values

No	Column	Number of missing values
1.	Budget	0
2.	Genres	0
3.	homepage	0
4.	Id	0
5.	Keywords	0
6.	Original_language	0
7.	Original_title	0
8.	Overview	0
9.	Popularity	0
10.	Production_companies	0
11.	Production_countries	0
12.	Release_date	0
13.	Revenue	0
14.	Runtime	0
15.	Spoken_languages	0
16.	Status	0
17.	Tagline	0
18.	title	0
19.	Vote_average	0
20.	Vote_count	0

Eliminating unnecessary data attributes

To address the removal of irrelevant data attributes, feature selection is employed. This method focuses on identifying and retaining the most pertinent attributes for analysis or prediction while discarding those with minimal contribution. In this case, the primary emphasis is on production costs, revenues, and movie ratings attributes. As a result, attributes like "genres", "homepage", "keywords", "original_language", "overview", "production_companies", "production_countries", "release_date", "spoken_languages", "status", "tagline", "original_title", and "title" are excluded from the dataset. The following steps are taken: Attributes considered necessary for the analysis are selected based on their relevance to production costs, revenues, and film ratings. The attributes retained are id, budget, popularity, revenue, runtime, vote_average, and vote_count. And then these attributes are chosen because they are critical indicators of a movie's success and are commonly used in movie recommendation systems. Attributes that are not directly related to the analysis, such as genres and homepage, are removed using the column drop technique in programming languages such as Python (in libraries such as Pandas) or by utilizing other data preparation tools. This removal reduces data complexity and ensures that only relevant attributes are used in the calculation.

Converting columns to numeric data types

Before implementing the SFS algorithm, it is crucial to convert the data columns into numeric formats to prevent errors during processing. This conversion ensures uniformity and precision in mathematical operations, as the algorithm relies on numeric inputs for its calculations. The algorithm may encounter issues or functions improperly when the data consists of non-numeric formats, such as text or mixed types. Additionally, numeric computations are typically faster and more efficient than text-based ones, making this conversion essential for optimizing performance and reducing computational overhead. Data accuracy is preserved by converting data to the correct numeric type, such as turning the age column into integers or floats. Non-numeric columns are transformed into numeric types using Python's `as type (float)` function. Table 4 provides an overview of the data types within the movie dataset.

Table 4. Data type information

No	Column	Dtype
1.	Budget	int
2.	Genres	object
3.	homepage	object
4.	Id	int
5.	Keywords	object
6.	Original_language	object
7.	Original_title	object
8.	Overview	object
9.	Popularity	float
10.	Production_companies	object
11.	Production_countries	object
12.	Release_date	object
13.	Revenue	int
14.	Runtime	float
15.	Spoken_languages	object
16.	Status	object
17.	Tagline	object
18.	title	object
19.	Vote_average	float
20.	Vote_count	int

Implementation of the SFS algorithm

Normalizing data

Normalization is a key step in our process, ensuring that all attributes are on a consistent scale. This efficient process prevents those with more extensive ranges from disproportionately influencing the analysis, and is achieved by adjusting each attribute's value by multiplying it by 0.000000001, as illustrated in Figure 4.

```
duplikat_dataset['normalized_budget'] = duplikat_dataset['budget'] * 0.000000001
duplikat_dataset['normalized_popularity'] = duplikat_dataset['popularity'] * 0.0000001
duplikat_dataset['normalized_revenue'] = duplikat_dataset['revenue'] * 0.000000001
duplikat_dataset['normalized_runtime'] = duplikat_dataset['runtime'] * 0.0000001
duplikat_dataset['normalized_vote_average'] = duplikat_dataset['vote_average'] * 0.000000001
duplikat_dataset['normalized_vote_count'] = duplikat_dataset['vote_count'] * 0.000000001
```

Figure 4. Python function for data normalization

Only the values within the attributes are subject to normalization. Each attribute's value is scaled by a factor of 0.000000001 to ensure consistency across all attributes. The outcome of this normalization process is displayed in Figure 5.

Entrophy value calculation

The normalized data is used to compute entropy values. The object with the highest entropy is identified as the first skyline object. Calculating entropy values involves multiple steps, as depicted in Figures 5 to 7, confirming that the object with the highest entropy surpasses others in the dataset.

a. Step 1

This verification step shows that the entropy values generated from the Python program code match the manual calculation results in Excel. The result of this verification can be seen in Figure 5.

T

id	budget	popularity	malisasi_bud	rmalisasi_popular	entropi
19995	237000000	150,43758	0,237	0,00000015	0,212689243
285	300000000	139,08262	0,300	0,00000014	0,262364404
206647	245000000	107,37679	0,245	0,00000011	0,21913564
49026	250000000	112,31295	0,250	0,00000011	0,223143661
49529	260000000	43,926995	0,260	0,00000004	0,231111761
559	258000000	115,69981	0,258	0,00000012	0,229523278
38757	260000000	48,681969	0,260	0,00000005	0,231111771
99861	280000000	134,27923	0,280	0,00000013	0,246860208
767	250000000	98,885637	0,250	0,0000001	0,223143651
209112	250000000	155,79045	0,250	0,00000016	0,223143711

T'

id	budget	popularity	rmalisasi_bud	nalisasi_popul	entropi
285	300000000	139,082615	0,300	0,00000014	0,2623644
99861	280000000	134,279229	0,280	0,00000013	0,24686021
38757	260000000	48,681969	0,260	0,00000005	0,23111177
49529	260000000	43,926995	0,260	0,00000004	0,23111176
559	258000000	115,699814	0,258	0,00000012	0,22952328
209112	250000000	155,790452	0,250	0,00000016	0,22314371
49026	250000000	112,31295	0,250	0,00000011	0,22314366
767	250000000	98,885637	0,250	0,0000001	0,22314365
206647	245000000	107,376788	0,245	0,00000011	0,21913564
19995	237000000	150,437577	0,237	0,00000015	0,21268924

Figure 5. Step 1 verification

T represents the outcome of the entropy calculations, while T' shows the sorted results in descending order. This sorting highlights the object with the highest entropy value, which is then selected as the initial skyline object.

b. Step 2

The second verification step aims to identify and highlight the object with the highest entropy value, which will be assigned as the first skyline object. A skyline object outperforms all other objects across every attribute.

S

id	budget	popularity	ormalisasi_budge	rmalisasi_popular	entropi
285	300000000	139,082615	0,300	0,00000014	0,262364404

T'

id	budget	popularity	normalisasi_budget	normalisasi_popularity	entropi
285	300000000	139,082615	0,300	0,00000014	0,2623644
99861	280000000	134,279229	0,280	0,00000013	0,24686021
38757	260000000	48,681969	0,260	0,00000005	0,23111177
49529	260000000	43,926995	0,260	0,00000004	0,23111176
559	258000000	115,699814	0,258	0,00000012	0,22952328
209112	250000000	155,790452	0,250	0,00000016	0,22314371
49026	250000000	112,31295	0,250	0,00000011	0,22314366
767	250000000	98,885637	0,250	0,0000001	0,22314365
206647	245000000	107,376788	0,245	0,00000011	0,21913564
19995	237000000	150,437577	0,237	0,00000015	0,21268924

Figure 6. Step 2 verification

As shown in Figure 6, the object with ID 285 holds the highest entropy value among all the data points. This indicates that ID 285 surpasses other objects with lower attribute values. The symbol "S" represents a skyline object, and the Figure illustrates how object ID 285 is transferred from tuple T' to tuple S, signifying its status as a skyline object.

c. Step 3

The goal of Step 3 verification is to evaluate other objects in the dataset through a dominance test. This test compares all data rows with lower entropy values than object ID 285. Each object's attributes are examined to determine if the values are superior in every attribute or if at least one attribute is better than the others. Suppose an object shows superiority across all attributes or excels in at least one. In that case, it is considered to dominate the others and is classified as a skyline object.

S

id	budget	popularity	ormalisasi_budge	rmalisasi_popular	entropi
285	300000000	139,082615	0,300	0,00000014	0,262364404
209112	250000000	155,790452	0,250	0,00000016	0,223143711

T'

id	budget	popularity	normalisasi_budget	normalisasi_popularity	entropi
285	300000000	139,082615	0,300	0,00000014	0,2623644
99861	280000000	134,279229	0,280	0,00000013	0,24686021
38757	260000000	48,681969	0,260	0,00000005	0,23111177
49529	260000000	43,926995	0,260	0,00000004	0,23111176
559	258000000	115,699814	0,258	0,00000012	0,22952328
209112	250000000	155,790452	0,250	0,00000016	0,22314371
49026	250000000	112,31295	0,250	0,00000011	0,22314366
767	250000000	98,885637	0,250	0,0000001	0,22314365
206647	245000000	107,376788	0,245	0,00000011	0,21913564
19995	237000000	150,437577	0,237	0,00000015	0,21268924

Figure 7. Step 3 verification

In Figure 7, a sample study with 10 data points identifies two skyline objects: ID 285 and 209112. These IDs are classified as skyline objects because no other objects in the dataset outperform them across all evaluated attributes, such as budget and popularity.

d. Final skyline objects

Once the dominance test is completed for all objects, the final skyline objects are identified. As illustrated in Figure 11, there are two skyline objects: ID 285 and 209112. These objects dominate all others with inferior attribute values. Based on the sample data that includes all attributes, ID 285 corresponds to "Pirates of the Caribbean: At World's End." In contrast, ID 209112 represents "Batman vs Superman: Dawn of Justice."

Table 5. Test results for 10 data points

ID	Movie title
285	Pirates of the Caribbean: At World's End
209112	Batman vs Superman: Down of Justice

Implementation results of the complete dataset

In applying the Sort Filter Skyline (SFS) algorithm for movie recommendations, the researchers used a complete dataset containing 4,803 data entries and seven attributes: id, budget, popularity, revenue, runtime, vote_average, and vote_count. These attributes were processed to calculate entropy using the entropy formula. The results of the entropy calculation for the entire dataset are displayed in Figure 8.

```
            id       budget  popularity       revenue  runtime  vote_average  \
0      19995.0  237000000.0  150.437577  2.787965e+09    162.0           7.2
1        285.0  300000000.0  139.082615  9.610000e+08    169.0           6.9
2     206647.0  245000000.0  107.376788  8.806746e+08    148.0           6.3
3      49026.0  250000000.0  112.312950  1.084939e+09    165.0           7.6
4      49529.0  260000000.0   43.926995  2.841391e+08    132.0           6.1
...        ...          ...         ...           ...      ...           ...
4798    9367.0     220000.0   14.269792  2.040920e+06     81.0           6.6
4799   72766.0       9000.0    0.642552  0.000000e+00     85.0           5.9
4800  231617.0          0.0    1.444476  0.000000e+00    120.0           7.0
4801  126186.0          0.0    0.857008  0.000000e+00     98.0           5.7
4802   25975.0          0.0    1.929883  0.000000e+00     90.0           6.3

      vote_count  normalized_budget  normalized_popularity  \
0        11800.0           0.237000           1.504376e-05
1         4500.0           0.300000           1.390826e-05
2         4466.0           0.245000           1.073768e-05
3         9106.0           0.250000           1.123130e-05
4         2124.0           0.260000           4.392699e-06
...          ...                ...                    ...
4798       238.0           0.000220           1.426979e-06
4799         5.0           0.000009           6.425520e-08
4800         6.0           0.000000           1.444476e-07
4801         7.0           0.000000           8.570080e-08
4802        16.0           0.000000           1.929883e-07

      normalized_revenue  normalized_runtime  normalized_vote_average  \
0               2.787965            0.000016             7.200000e-09
1               0.961000            0.000017             6.900000e-09
2               0.880675            0.000015             6.300000e-09
3               1.084939            0.000016             7.600000e-09
4               0.284139            0.000013             6.100000e-09
...                  ...                 ...                      ...
4798            0.002041            0.000008             6.600000e-09
4799            0.000000            0.000008             5.900000e-09
4800            0.000000            0.000012             7.000000e-09
4801            0.000000            0.000010             5.700000e-09
4802            0.000000            0.000009             6.300000e-09

      normalized_vote_count   entropy
0              1.180000e-05  1.544561
1              4.500000e-06  0.935854
2              4.466000e-06  0.850796
3              9.106000e-06  0.957920
4              2.124000e-06  0.481220
...                     ...       ...
4798           2.380000e-07  0.002269
4799           5.000000e-09  0.000018
4800           6.000000e-09  0.000012
4801           7.000000e-09  0.000010
4802           1.600000e-08  0.000009

[4803 rows x 14 columns]
```

Figure 8. Entropy calculation results using the complete dataset

Following the entropy calculation, the data is arranged in descending order to identify the first skyline object from the dataset. The results of this sorting process are displayed in Figure 9.

```
            id       budget  popularity       revenue  runtime  vote_average  \
0      19995.0  237000000.0  150.437577  2.787965e+09    162.0           7.2
25       597.0  200000000.0  100.025899  1.845034e+09    194.0           7.5
7      99861.0  280000000.0  134.279229  1.405404e+09    141.0           7.3
16     24428.0  220000000.0  144.448633  1.519558e+09    143.0           7.4
44    168259.0  190000000.0  102.322217  1.506249e+09    137.0           7.3
...        ...          ...         ...           ...      ...           ...
4633  300327.0          0.0    0.005883  0.000000e+00      0.0           0.0
4118  325140.0          0.0    0.001186  0.000000e+00      0.0           0.0
4553  380097.0          0.0    0.000000  0.000000e+00      0.0           0.0
2656  370980.0   15000000.0    0.738646  0.000000e+00      NaN           7.3
4140  459488.0          2.0    0.050625  0.000000e+00      NaN           0.0

      vote_count  normalized_budget  normalized_popularity  \
0        11800.0       2.370000e-01           1.504376e-05
25        7562.0       2.000000e-01           1.000259e-05
7         6767.0       2.800000e-01           1.342792e-05
16       11776.0       2.200000e-01           1.444486e-05
44        4176.0       1.900000e-01           1.023222e-05
...          ...                ...                    ...
4633         0.0       0.000000e+00           5.883000e-10
4118         0.0       0.000000e+00           1.186000e-10
4553         0.0       0.000000e+00           0.000000e+00
2656        12.0       1.500000e-02           7.386460e-08
4140         0.0       2.000000e-09           5.062500e-09

      normalized_revenue  normalized_runtime  normalized_vote_average  \
0               2.787965            0.000016             7.200000e-09
25              1.845034            0.000019             7.500000e-09
7               1.405404            0.000014             7.300000e-09
16              1.519558            0.000014             7.400000e-09
44              1.506249            0.000014             7.300000e-09
...                  ...                 ...                      ...
4633            0.000000            0.000000             0.000000e+00
4118            0.000000            0.000000             0.000000e+00
4553            0.000000            0.000000             0.000000e+00
2656            0.000000                 NaN             7.300000e-09
4140            0.000000                 NaN             0.000000e+00

      normalized_vote_count       entropy
0              1.180000e-05  1.544561e+00
25             7.562000e-06  1.227934e+00
7              6.767000e-06  1.124612e+00
16             1.177600e-05  1.122975e+00
44             4.176000e-06  1.092769e+00
...                     ...           ...
4633           0.000000e+00  5.883001e-10
4118           0.000000e+00  1.186000e-10
4553           0.000000e+00  0.000000e+00
2656           1.200000e-08           NaN
4140           0.000000e+00           NaN

[4803 rows x 14 columns]
```

Figure 9. Sorting result of entropy using the complete dataset

As shown in Figure 10, ID 19995 holds the highest entropy value of 1.544561, making it the first skyline object in this analysis. The subsequent step involves performing a dominance test to identify the next skyline object. This test compares the attributes of each row with the results presented in Figure 10.

Skyline records:

	id	budget	popularity	revenue	runtime	vote_average	vote_count	normalized_budget	normalized_popularity	normalized_revenue	normalized_runtime	normalizec
0	19995.0000000000	237000000.0000000000	150.4375770000	2787965087.0000000000	162.0000000000	7.2000000000	11800.0000000000	0.2370000000	0.0000150438	2.7879650870	0.0000162000	
1	597.0000000000	200000000.0000000000	100.0258990000	1845034188.0000000000	194.0000000000	7.5000000000	7562.0000000000	0.2000000000	0.0000100026	1.8450341880	0.0000194000	
2	99861.0000000000	280000000.0000000000	134.2792290000	1405403694.0000000000	141.0000000000	7.3000000000	6767.0000000000	0.2800000000	0.0000134279	1.4054036940	0.0000141000	
3	24428.0000000000	220000000.0000000000	144.4486330000	1519557910.0000000000	143.0000000000	7.4000000000	11776.0000000000	0.2200000000	0.0000144449	1.5195579100	0.0000143000	
4	168259.0000000000	190000000.0000000000	102.3222170000	1506249360.0000000000	137.0000000000	7.3000000000	4176.0000000000	0.1900000000	0.0000102322	1.5062493600	0.0000137000	
5	135397.0000000000	150000000.0000000000	418.7085520000	1513528810.0000000000	124.0000000000	6.5000000000	8662.0000000000	0.1500000000	0.0000418709	1.5135288100	0.0000124000	
6	1865.0000000000	380000000.0000000000	135.4138560000	1045713802.0000000000	136.0000000000	6.4000000000	4948.0000000000	0.3800000000	0.0000135414	1.0457138020	0.0000136000	
7	271110.0000000000	250000000.0000000000	198.3723950000	1153304495.0000000000	147.0000000000	7.1000000000	7241.0000000000	0.2500000000	0.0000198372	1.1533044950	0.0000147000	
8	68721.0000000000	200000000.0000000000	77.6820800000	1215439994.0000000000	130.0000000000	6.8000000000	8806.0000000000	0.2000000000	0.0000077682	1.2154399940	0.0000130000	
9	109445.0000000000	150000000.0000000000	165.1253660000	1274219009.0000000000	102.0000000000	7.3000000000	5295.0000000000	0.1500000000	0.0000165125	1.2742190090	0.0000102000	
10	49026.0000000000	250000000.0000000000	112.3129500000	1084939099.0000000000	165.0000000000	7.6000000000	9106.0000000000	0.2500000000	0.0000112313	1.0849390990	0.0000165000	
11	285.0000000000	300000000.0000000000	139.0826150000	961000000.0000000000	169.0000000000	6.9000000000	4500.0000000000	0.3000000000	0.0000139083	0.9610000000	0.0000169000	
12	38356.0000000000	195000000.0000000000	28.5296070000	1123746996.0000000000	154.0000000000	6.1000000000	3299.0000000000	0.1950000000	0.0000028530	1.1237469960	0.0000154000	
13	91314.0000000000	210000000.0000000000	116.8402960000	1091405097.0000000000	165.0000000000	5.8000000000	3095.0000000000	0.2100000000	0.0000116840	1.0914050970	0.0000165000	
14	37724.0000000000	200000000.0000000000	93.0049930000	1108561013.0000000000	143.0000000000	6.9000000000	7604.0000000000	0.2000000000	0.0000093005	1.1085610130	0.0000143000	
15	49051.0000000000	250000000.0000000000	108.8496210000	1021103568.0000000000	169.0000000000	7.0000000000	8297.0000000000	0.2500000000	0.0000108850	1.0211035680	0.0000169000	
16	57158.0000000000	250000000.0000000000	94.3705640000	958400000.0000000000	161.0000000000	7.6000000000	4524.0000000000	0.2500000000	0.0000094371	0.9584000000	0.0000161000	
17	155.0000000000	185000000.0000000000	187.3229270000	1004558444.0000000000	152.0000000000	8.2000000000	12002.0000000000	0.1850000000	0.0000187323	1.0045584440	0.0000152000	
18	209112.0000000000	250000000.0000000000	155.7904520000	873260194.0000000000	151.0000000000	5.7000000000	7004.0000000000	0.2500000000	0.0000155790	0.8732601940	0.0000151000	
19	206647.0000000000	245000000.0000000000	107.3767880000	880674609.0000000000	148.0000000000	6.3000000000	4466.0000000000	0.2450000000	0.0000107377	0.8806746090	0.0000148000	
20	122.0000000000	94000000.0000000000	123.6303320000	1118888979.0000000000	201.0000000000	8.1000000000	8064.0000000000	0.0940000000	0.0000123630	1.1188889790	0.0000201000	
21	211672.0000000000	74000000.0000000000	875.5813050000	1156730962.0000000000	91.0000000000	6.4000000000	4571.0000000000	0.0740000000	0.0000875581	1.1567309620	0.0000091000	
22	278927.0000000000	175000000.0000000000	94.1993160000	966550600.0000000000	106.0000000000	6.7000000000	2892.0000000000	0.1750000000	0.0000094199	0.9665506000	0.0000106000	
23	127585.0000000000	250000000.0000000000	118.0786910000	747862775.0000000000	131.0000000000	7.5000000000	6032.0000000000	0.2500000000	0.0000118079	0.7478627750	0.0000131000	
24	150540.0000000000	175000000.0000000000	128.6559640000	857611174.0000000000	94.0000000000	8.0000000000	6560.0000000000	0.1750000000	0.0000128656	0.8576111740	0.0000094000	
25	27205.0000000000	160000000.0000000000	167.5837100000	825532764.0000000000	148.0000000000	8.1000000000	13752.0000000000	0.1600000000	0.0000167584	0.8255327640	0.0000148000	
26	101299.0000000000	130000000.0000000000	76.3101190000	847423452.0000000000	146.0000000000	7.4000000000	6495.0000000000	0.1300000000	0.0000076310	0.8474234520	0.0000146000	
27	118340.0000000000	170000000.0000000000	481.0986240000	773328629.0000000000	121.0000000000	7.9000000000	9742.0000000000	0.1700000000	0.0000481099	0.7733286290	0.0000121000	
28	672.0000000000	100000000.0000000000	132.3977370000	876688482.0000000000	161.0000000000	7.4000000000	5815.0000000000	0.1000000000	0.0000132398	0.8766884820	0.0000161000	
29	297761.0000000000	175000000.0000000000	90.2379200000	745000000.0000000000	123.0000000000	5.9000000000	7458.0000000000	0.1750000000	0.0000090238	0.7450000000	0.0000123000	
30	120.0000000000	93000000.0000000000	138.0495770000	871368364.0000000000	178.0000000000	8.0000000000	8705.0000000000	0.0930000000	0.0000138050	0.8713683640	0.0000178000	
31	14160.0000000000	175000000.0000000000	92.2019620000	735099082.0000000000	96.0000000000	7.7000000000	6870.0000000000	0.1750000000	0.0000092202	0.7350990820	0.0000096000	
32	328111.0000000000	75000000.0000000000	31.4828720000	875958308.0000000000	87.0000000000	5.9000000000	3462.0000000000	0.0750000000	0.0000031483	0.8759583080	0.0000087000	
33	100402.0000000000	170000000.0000000000	72.2252650000	714766572.0000000000	136.0000000000	7.6000000000	5764.0000000000	0.1700000000	0.0000072225	0.7147665720	0.0000136000	
34	38757.0000000000	260000000.0000000000	48.6819690000	591794936.0000000000	100.0000000000	7.4000000000	3330.0000000000	0.2600000000	0.0000048682	0.5917949360	0.0000100000	
35	119450.0000000000	170000000.0000000000	243.7917430000	710644566.0000000000	130.0000000000	7.3000000000	4410.0000000000	0.1700000000	0.0000243792	0.7106445660	0.0000130000	
36	131631.0000000000	125000000.0000000000	206.2271510000	752100229.0000000000	123.0000000000	6.6000000000	5584.0000000000	0.1250000000	0.0000206227	0.7521002290	0.0000123000	
37	157336.0000000000	165000000.0000000000	724.2477840000	675120017.0000000000	169.0000000000	8.1000000000	10867.0000000000	0.1650000000	0.0000724248	0.6751200170	0.0000169000	
38	177572.0000000000	165000000.0000000000	203.7345900000	652105443.0000000000	102.0000000000	7.8000000000	6135.0000000000	0.1650000000	0.0000203735	0.6521054430	0.0000102000	
39	293660.0000000000	58000000.0000000000	514.5699560000	783112979.0000000000	108.0000000000	7.4000000000	10995.0000000000	0.0580000000	0.0000514570	0.7831129790	0.0000108000	
40	254.0000000000	207000000.0000000000	61.2260100000	550000000.0000000000	187.0000000000	6.6000000000	2337.0000000000	0.2070000000	0.0000061226	0.5500000000	0.0000187000	
41	70160.0000000000	75000000.0000000000	68.5506980000	691210692.0000000000	142.0000000000	6.9000000000	9455.0000000000	0.0750000000	0.0000068551	0.6912106920	0.0000142000	
42	286217.0000000000	108000000.0000000000	167.9328700000	630161890.0000000000	141.0000000000	7.6000000000	7268.0000000000	0.1080000000	0.0000167933	0.6301618900	0.0000141000	
43	10681.0000000000	180000000.0000000000	66.3907120000	521311860.0000000000	98.0000000000	7.8000000000	6296.0000000000	0.1800000000	0.0000066391	0.5213118600	0.0000098000	
44	1452.0000000000	270000000.0000000000	57.9256230000	391081192.0000000000	154.0000000000	5.4000000000	1400.0000000000	0.2700000000	0.0000057926	0.3910811920	0.0000154000	
45	417859.0000000000	130000000.0000000000	20.6787870000	554987477.0000000000	90.0000000000	6.4000000000	451.0000000000	0.1300000000	0.0000020679	0.5549874770	0.0000090000	
46	281957.0000000000	135000000.0000000000	100.6358820000	532950503.0000000000	156.0000000000	7.3000000000	6396.0000000000	0.1350000000	0.0000100636	0.5329505030	0.0000156000	
47	676.0000000000	140000000.0000000000	34.2066900000	449220945.0000000000	183.0000000000	6.6000000000	1791.0000000000	0.1400000000	0.0000034207	0.4492209450	0.0000183000	
48	137113.0000000000	178000000.0000000000	79.4564850000	370541256.0000000000	113.0000000000	7.6000000000	4858.0000000000	0.1780000000	0.0000079456	0.3705412560	0.0000113000	
49	324668.0000000000	120000000.0000000000	62.6412860000	415484914.0000000000	123.0000000000	5.9000000000	2341.0000000000	0.1200000000	0.0000062641	0.4154849140	0.0000123000	
50	1891.0000000000	18000000.0000000000	78.5178300000	538400000.0000000000	124.0000000000	8.2000000000	5879.0000000000	0.0180000000	0.0000078518	0.5384000000	0.0000124000	
51	106646.0000000000	100000000.0000000000	95.0079340000	392000694.0000000000	180.0000000000	7.9000000000	6571.0000000000	0.1000000000	0.0000095008	0.3920006940	0.0000180000	

Figure 10. Skyline record results using the dataset

The dominance test identified 176 data objects as skyline objects, with a processing time of 4 seconds. The results of this test are displayed in Table 6. Based on the original dataset, the corresponding movie titles for these IDs are as follows:

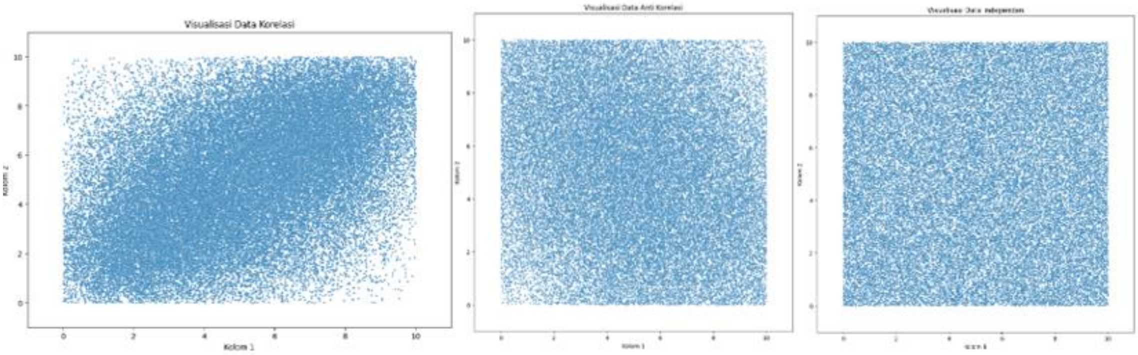
Table 6. Skyline record results

ID	Movie title
19995	Avatar
597	Titanic
99861	Avengers: Age of Ultron
24428	The Avengers
16825	Furious 7
135397	Jurassic World
1865	Pirates of the Caribbean: On Stranger Tides
271110	Captain America: Civil War
68721	Iron Man 3
109445	Frozen

These objects are selected because others do not outperform them across all dimensions. For instance, ID 19995 is identified as the first skyline object due to having the highest entropy among the seven calculated attributes, meaning it dominates all other objects with inferior attribute values. Similarly, ID 597 is chosen as the next skyline object based on the dominance test, as it surpasses others in all attributes. Several factors, including the number of attributes, the dataset size, and the data types, influence the final skyline results. Consequently, results derived from processing two attributes will differ from those using seven attributes, as the skyline algorithm makes recommendations based on specific preferences. For example, if the recommendation prioritizes popularity and user votes, the final set of skyline objects will differ.

Data visualization

To evaluate execution time optimization, this study utilized synthetic datasets comprising 50,000 data rows with varying numbers of attributes, from 1 to 7. The dataset was used to test the skyline algorithm, which aims to find subsets of data not dominated by others across all considered attributes. The synthetic data includes three types of distributions: correlated, anti-correlated, and independent. These distributions are visualized using scatter plots, as illustrated in Figure 11.



(a) Correlation (b) Anti-correlation (c) independent

Figure 11. Visualization of synthetic data distribution

Figure (a) illustrates the visualization of correlated data distribution using two variables, displaying a positive linear relationship—when one variable increases, the other also tends to rise, albeit with some variation. In contrast, Figure (b) represents the anti-correlated data distribution, where a negative linear relationship is evident; as one variable increases, the other typically decreases. Finally, Figure (c) shows the independent data distribution, where no discernible pattern exists between the two variables, indicating an absence of any linear or identifiable relationship between them.

The process of visualizing the execution time complexity of the skyline algorithm is carried out using Python's `matplotlib.pyplot` function. A line chart is used to represent this data, and the pseudocode below outlines the Python function employed to measure the complexity of the skyline algorithm's execution time: [30]

Pseudocode Sort Filter Skyline Algorithm

Input: A set of points P, each point with d dimensions.
Output: A subset of P that forms the Skyline.

```
FUNCTION Sort_Filter_Skyline(P):
    // Step 1: Sort the points
    Sort P lexicographically by each dimension in ascending order.
    // Step 2: Initialize the result set
    Skyline = []
    // Step 3: Filter the points
    FOR each point p IN P:
        Dominated = FALSE
        FOR each point q IN Skyline:
            IF q dominates p:
                Dominated = TRUE
                BREAK
            // Check if p dominates q
            IF p dominates q:
                REMOVE q from Skyline
        IF NOT Dominated:
            ADD p to Skyline
    RETURN Skyline

// Function to check if one point dominates another
FUNCTION dominates(point_a, point_b):
    FOR each dimension i:
        IF point_a[i] > point_b[i]:
            RETURN FALSE
    RETURN TRUE if point_a is strictly better in at least one dimension.
```

The skyline algorithm is executed on subsets of data containing 1 to 7 attributes, using 50,000 samples across three distinct types of data distributions, as illustrated in Figure 13. This test aims to assess variations in the complexity of skyline execution time. In general, faster execution times indicate greater algorithm efficiency. The time taken for the algorithm to complete the skyline computation is recorded for each case. The visualization of the skyline algorithm's execution time complexity, based on synthetic data with correlated, anti-correlated, and independent distributions, is depicted in Figure 13.

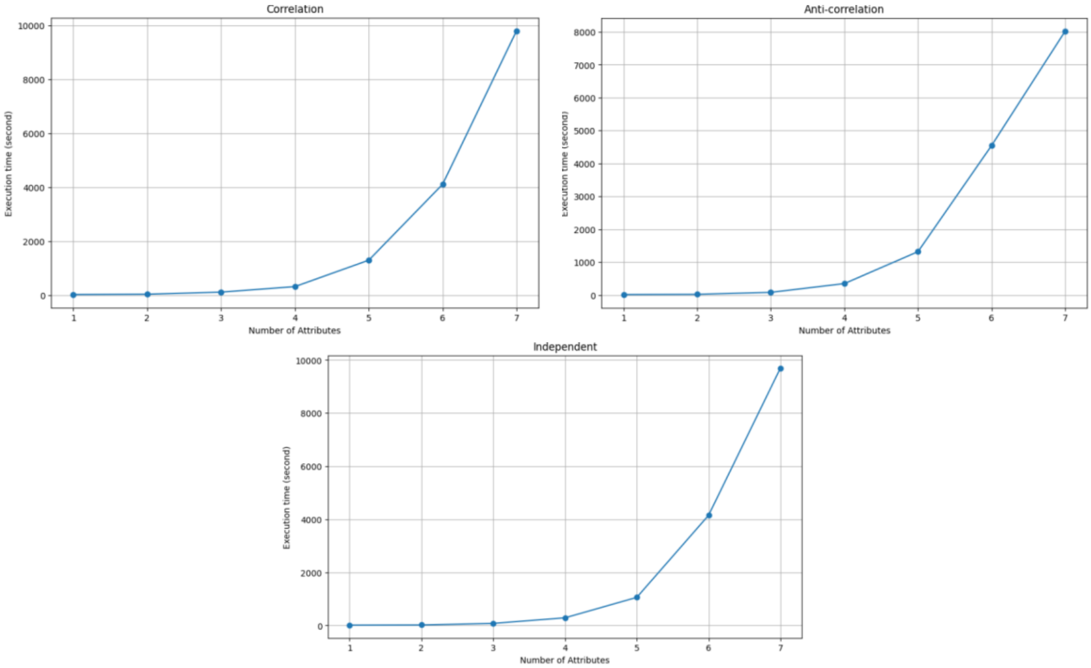


Figure 12. Time complexity

The Figure above illustrates the execution time variations based on different data types. The following explains the three types of time complexities:

Table 7. Execution time

Number of attributes	Correlated	Anti-Correlated	Independent
1	18,712104 second	19,208275 second	21,111396 second
2	26,465694 second	27,333808 second	27,639290 second
3	97,603342 second	100,403097 second	102,252056 second
4	342,991636 second	343,858338 second	351,526076 second
5	1206,378094 second	1240,250683 second	1281,096697 second
6	4215,874336 second	4492,397515 second	4513,496795 second
7	9697,180887 second	10299,508747 second	10568,557912 second

As shown in Figure 12 and Table 7, the execution time for correlated data increases exponentially as the number of attributes grows. A similar pattern is observed with anti-correlated data, though it takes slightly longer to execute than correlated data. Independent data also exhibits an exponential rise in execution time, generally longer than correlated and anti-correlated datasets, especially as the number of attributes increases. All three data types demonstrate that increasing the number of attributes leads to an exponential rise in execution time, with independent data having the longest processing time compared to the other two.

CONCLUSION

This study successfully applied the Sort Filter Skyline (SFS) algorithm to generate personalized movie recommendations based on individual preferences. The study focused on appealing movies due to their high production costs, substantial revenue, popularity, and strong ratings. The results revealed that out of 4,803 movies, 176 skyline objects were identified. Furthermore, time complexity testing using three types of synthetic data—correlated, anti-correlated, and independent—demonstrated that as the dataset and number of attributes increase, the time required to identify skyline objects also grows. While the SFS algorithm proves effective in producing customized movie recommendations, there are challenges related to execution time that must be addressed to enhance the efficiency of the recommendation process. Future research should explore incorporating more attributes and diverse datasets and developing a web-based application to improve the overall user experience.

REFERENCES

[1] A. W. Finaka, S. B. Negara, dan M. I. D. Putra, "Sejarah Hari Film Nasional," *[online]*, 2019. https://indonesiabaik.id/motion_grafis/sejarah-hari-film-nasional.

[2] A. Renner, "April 2024 Movies," *[online]*, 2024. https://www.movieinsider.com/movies/april/2024.

[3] V. Subramaniyaswamy, R. Logesh, M. Chandrashekhar, A. Challa, dan V. Vijayakumar, "A personalised movie recommendation system based on collaborative filtering," *Int. J. High Perform. Comput. Netw.*, vol. 10, hal. 54–63, 2017, doi: https://doi.org/10.1504/IJHPCN.2017.083199 PDF.

[4] M. Goyani dan N. Chaurasiya, "A Review of Movie Recommendation System: Limitations, Survey and Challenges," *Electron. Lett. Comput. Vis. Image Anal.*, vol. 19, no. 3, hal. 18–37, 2020, doi: 10.5565/rev/elcvia.1232.

[5] Z. Wang, X. Yu, N. Feng, dan Z. Wang, "An improved collaborative movie recommendation system using computational intelligence," *J. Vis. Lang. Comput.*, vol. 25, no. 6, hal. 667–675, 2014, doi: 10.1016/j.jvlc.2014.09.011.

[6] S. Borzsonyil dan K. Stocker, "The Skyline Operator *," *Proc. 17th Int. Conf. Data Eng.*, hal. 421–430, 2001.

[7] J. Chomicki, P. Godfrey, J. Gryz, dan D. Liang, "Skyline with presorting," *Proc. - Int. Conf. Data Eng.*, no. April, hal. 717–719, 2003, doi: 10.1109/ICDE.2003.1260846.

[8] S. Shah, A. Thakkar, dan S. Rami, "A Survey Paper on Skyline Query using Recommendation System," *Int. J. Data Min. Emerg. Technol.*, vol. 6, no. 1, hal. 1–6, 2016, doi: 10.5958/2249-3220.2016.00001.x.

[9] Y. Zeng, G. Chen, K. Li, Y. Zhou, X. Zhou, dan K. Li, "M-Skyline: Taking sunk cost and alternative recommendation in consideration for skyline query on uncertain data," *Knowledge-Based Syst.*, vol. 163, hal. 204–213, 2019, doi: 10.1016/j.knosys.2018.08.024.

[10] R. Amin, T. Djatna, Annisa, dan I. S. Sitanggang, "Recommendation system based on skyline query: Current and future research," *2020 Int. Conf. Comput. Sci. Its Appl. Agric. ICOSICA 2020*, 2020, doi: 10.1109/ICOSICA49951.2020.9243225.

[11] H. Zhu, X. Li, Q. Liu, dan Z. Xu, "Top-k Dominating Queries on Skyline Groups," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 7, hal. 1431–1444, 2020, doi: 10.1109/TKDE.2019.2904065.

[12] V. Purwayoga, "Modified skyline query to measure priority region for personal protective equipment recipient of COVID-19 health workers," *J. Teknol. dan Sist. Komput.*, vol. 9, no. 3, hal. 167–173, 2021, doi: 10.14710/jtsiskom.2021.14003.

[13] A. Cuzzocrea, P. Karras, dan A. Vlachou, "Effective and efficient skyline query processing over attribute-order-preserving-free encrypted data in cloud-enabled databases," *Futur. Gener. Comput. Syst.*, vol. 126, hal. 237–251, 2022, doi: 10.1016/j.future.2021.08.008.

[14] T. M. D. (TMDb), "TMDB 5000 Movie Dataset," *[online]*. https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata (diakses Mei 13, 2024).

[15] S. K. Gupta dan A. Suresh, "Movie Recommendation System," *2023 Int. Conf. Comput. Commun. Informatics, ICCCI 2023*, hal. 1–7, 2023, doi: 10.1109/ICCCI56745.2023.10128220.

[16] N. Kapoor dan S. Vishal, "Tools," no. Icces, hal. 883–888, 2020.

[17] MD Rokibul Hasan dan Janatul Ferdous, "Dominance of AI and Machine Learning Techniques in Hybrid Movie Recommendation System Applying Text-to-number Conversion and Cosine Similarity Approaches," *J. Comput. Sci. Technol. Stud.*, vol. 6, no. 1, hal. 94–102, 2024, doi: 10.32996/jcsts.2024.6.1.10.

[18] S. Kumar, K. De, dan P. P. Roy, "Movie Recommendation System Using Sentiment Analysis from Microblogging Data," *IEEE Trans. Comput. Soc. Syst.*, vol. 7, no. 4, hal. 915–923, 2020, doi: 10.1109/TCSS.2020.2993585.

[19] N. P. Sable, A. Yenkikar, dan P. Pandit, "Movie Recommendation System Using Cosine Similarity," *2024 IEEE 9th Int. Conf. Converg. Technol. I2CT 2024*, vol. 7, no. 4, hal. 342–346, 2024, doi: 10.1109/I2CT61223.2024.10543873.

[20] A. Hiro Juni Permana dan A. Toto Wibowo, "Movie Recommendation System Based on Synopsis Using Content-Based Filtering with TF-IDF and Cosine Similarity," *Intl. J. ICT*, vol. 9, no. 2, hal. 1–14, 2023, doi: 10.21108/ijoict.v9i2.747.

[21] D. P. Kumar, A. K. Singh, S. N. Arepu, M. Sarvasuddi, E. Gowtham, dan Y. Sanjana, "Content Based Recommendation System on Movies," *Proc. Second Int. Conf. Emerg. Trends Eng. (ICETE 2023)*, 2023, doi: 10.2991/978-94-6463-252-1_49.

[22] A. Annisa dan S. Khairina, "Location Selection Based on Surrounding Facilities in Google Maps using Sort Filter Skyline Algorithm," *Khazanah Inform. J. Ilmu Komput. dan Inform.*, vol. 7, no.

2, hal. 65–72, 2021, doi: 10.23917/khif.v7i2.12939.
[23] C. H. Loh, Y. C. Chen, C. T. Su, dan S. H. Lin, “Multi-Objective Decision Support for Irrigation Systems Based on Skyline Query,” *Appl. Sci.*, vol. 14, no. 3, 2024, doi: 10.3390/app14031189.
[24] M. Mukhlis, A. Kustiyo, dan A. Suharso, “Peramalan Produksi Pertanian Menggunakan Model Long Short-Term Memory,” *Bina Insa. Ict J.*, vol. 8, no. 1, hal. 22, 2021, doi: 10.51211/biict.v8i1.1492.
[25] H. S. A. Geofani, “Aplikasi Algoritma Apriori dalam Data Mining Penjualan Tanaman Hias,” *J. Informatics Data Min.*, vol. 10, no. 5, hal. 1–16, 2024, [Daring]. Tersedia pada: https://www.researchgate.net/publication/380957197_Aplikasi_Algoritma_Apriori_dalam_Data_Mining_Penjualan_Tanaman_Hias.
[26] D. A. Nasution, H. H. Khotimah, dan N. Chamidah, “Perbandingan Normalisasi Data untuk Klasifikasi Wine Menggunakan Algoritma K-NN,” *Comput. Eng. Sci. Syst. J.*, vol. 4, no. 1, hal. 78, 2019, doi: 10.24114/cess.v4i1.11458.
[27] R. G. Whendasmoro dan J. Joseph, “Analisis Penerapan Normalisasi Data Dengan Menggunakan Z-Score Pada Kinerja Algoritma K-NN,” *JURIKOM (Jurnal Ris. Komputer)*, vol. 9, no. 4, hal. 872, 2022, doi: 10.30865/jurikom.v9i4.4526.
[28] Y. Gulzar, A. A. Alwan, dan S. Turaev, “Optimizing Skyline Query Processing in Incomplete Data,” *IEEE Access*, vol. 7, hal. 178121–178138, 2019, doi: 10.1109/ACCESS.2019.2958202.
[29] M. A. Mohamud *et al.*, “A Systematic Literature Review of Skyline Query Processing Over Data Stream,” *IEEE Access*, vol. 11, no. July, hal. 72813–72835, 2023, doi: 10.1109/ACCESS.2023.3295117.
[30] J. Shomicki, P. Godfrey, J. Gryz, dan D. Liang, “skyline with presorting,” *Proc. 19th Int. Conf. Data Eng. (Cat. No.03CH37405)*, vol. 49, no. 0, hal. 1-33 : 29 pag texts + end notes, appendix, referen, 2003, doi: 10.1109/ICDE.2003.1260846.