Data Mining and Machine Learning Project

Game Recommendation System

Giovanni Paolini - 628051

Introduction

The goal of the project is to implement a game recommendation system to suggest video games to users based on previously liked games. The aim of the application is to cluster similar games together, delivering personalized recommendations.

Dataset Info

The dataset used as a base for the project (**The Steam Games Dataset**) is available on Kaggle[1]
The dataset is composed of 83560 entries where each entry is a Game from the Steam Store page.

Data Preprocessing

Exploration

Attributes

The dataset contains the following attributes:

1. **AppID:** Game ID on the steam website
2. **Name:** Name of the game
3. **Release date:** The date when the game was released
4. **Estimated owners:** Range of how many users own the game
5. **Peak CCU:** All time highest number of Concurrent users
6. **Required age:** Minimum age suggestion for the game
7. **Price:** Price of the game on the Steam Store
8. **DLC count:** Number of extra Downloadable contents for the game
9. **About the game:** Short description of the game
10. **Supported languages:** Available languages for the game
11. **Full audio languages:** Available dub languages for the game
12. **Header image:** Link to the image cover for the game
13. **Website:** Link to the game or game producer official website
14. **Windows:** Availability of the game for windows OS
15. **Mac:** Availability of the game for IOS OS
16. **Linux:** Availability of the game for Linux OS
17. **Metacritic score:** Score for the game from the Metacritic website
18. **User score:** Score for the game from Steam Users
19. **Positive:** Positive votes for the game from Steam Users
20. **Negative:** Negative votes for the game from Steam Users
21. **Achievements:** Number of achievements available for the game
22. **Recommendations:**Number of recommendations for the game
23. **Average playtime forever:** Average playtime from the game release
24. **Average playtime two weeks:** Average playtime in the last 2 weeks
25. **Median playtime forever:** Median playtime from the game release
26. **Median playtime two weeks:** Median playtime in the last 2 weeks
27. **Developers:** Developers of the game
28. **Publishers:** Publisher of the game
29. **Categories:** General categories tags for the game
30. **Genres:** Genres the game belongs to
31. **Tags:** Tags associated to the game
32. **Screenshots:** Link to the Steam screenshot page for the game
33. **Movies:** Link to the Steam captured video page for the game
34. **Support email:** Email for the support service of the game
35. **Support url:** Link to the support service of the game
36. **Notes:** Extra info about the game (Legal or content related)

37. **Reviews:** User reviews for the game
38. **Metacritic url:** Link to the Metacritic page of the game

39. Score rank: Not clearly inferable from the data (int, almost all null entries)

Null values

For some of the attributes, a high number of entries have a null value associated with them. The following table shows the null value count for each attribute that has at least 1 null value:

Attribute	Null Entries	Non-Null Entries
Score rank	83516	44
Metacritic url	79650	3910
Reviews	73844	9716
Notes	70845	12715
Website	44506	39054
Support url	42521	41039
Tags	19986	63574
Support email	13313	70247
Movies	6300	77260
Categories	4456	79104
Publishers	3674	79886
Developers	3455	80105
About the game	3438	80122
Genres	3425	80135
Screenshots	1926	81634
Name	6	83554

Duplicates

Using the AppID attribute, we check for duplicate entries in the dataset.

```
#Check for duplicate games in the dataset
print(df['AppID'].nunique())
```

```
83560
```

The number of unique Application IDs matches the total number of entries in the dataset (83560), so it is safe to assume that there are no duplicates of the same game in the dataset, since the AppID is unique for each game.

Cleaning

Missing values

Looking at the Null values table, there are many features which are null in many entries, but most of those attributes will not be useful for the recommendation system anyway.
With this in mind we can safely drop the following attributes without trying to fill their values:

- **Score Rank**: Since the scope of this attribute is not clear and there are only 44 non-null entries, this attribute will be dropped from the dataset.
- **Metacritic Url**: Since the Link to metacritic will have no use in the recommendation system, the attribute will be dropped from the dataset.
- **Reviews**: Since the reviews have a high count of null entries and they are textual reviews without an associated positive or negative rating, it would be necessary to perform a sentiment analysis to use the reviews as a weight in the recommendation system. For these reasons the attribute will be dropped from the dataset.
- **Notes:** The values for this attribute are mostly legal or warning notes regarding the game, such as “May contain inappropriate content”, which would be redundant with the categories and tags of the game, since the content could be easily inferred from those (i.e. Shooting Category → Violent game). This attribute will be dropped from the dataset.
- **Website**: The website of the game is redundant, since it can be found in the Steam page of the game and for that only the AppID is needed. This attribute will be dropped from the dataset.
- **Support Url** and **Support Email**: These attributes have no use in the context of a recommendation system and they can be found in the Steam page of the game. Both the attributes will be dropped from the dataset.
- **Movies** and **Screenshots**: These attributes have no use in the context of the project, so both of them will be dropped from the dataset.
- **Publishers, Developers** and **About the game**: These can all be found on the Steam page of the game, they will be dropped from the dataset.

Further analysis is needed for the following attributes with null values:

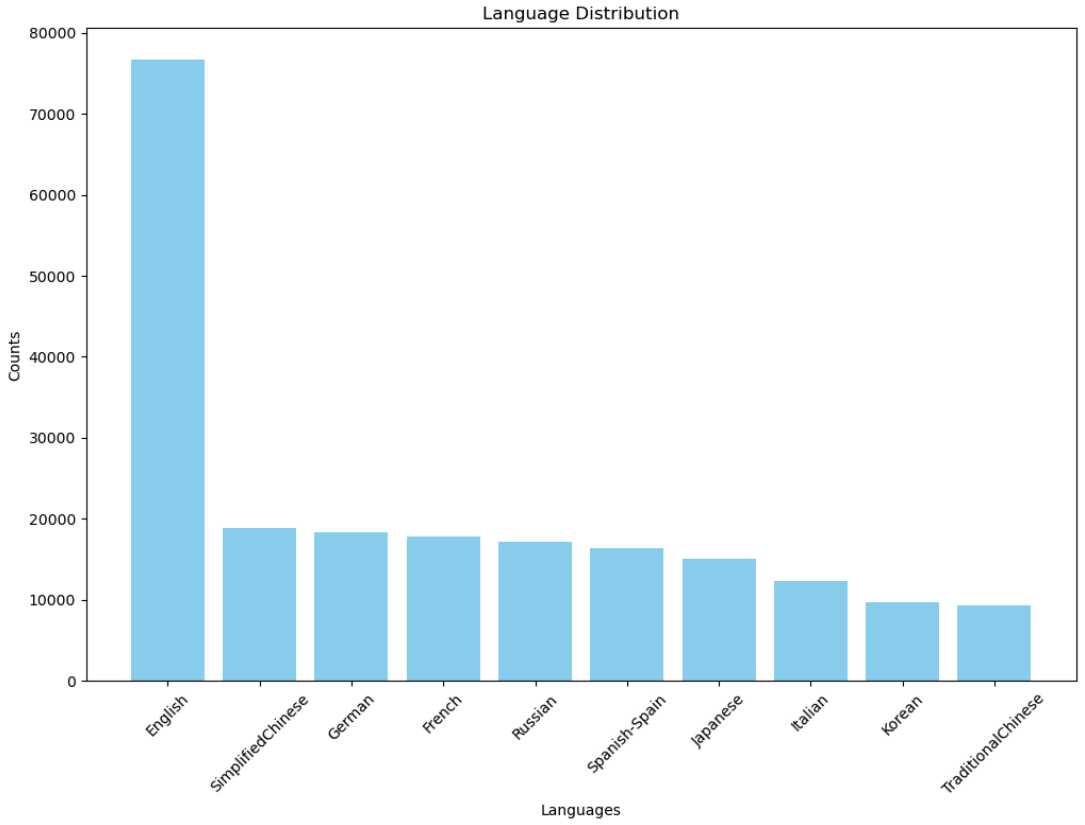
- **Tags,Categories** and **Genres**:For these attributes we have a null count of 19986, 4456, 3425 respectively. Being those attributes really important for the weighting of the recommendation system, further handling is needed. Since the Categories and Genres on steam may be regarded as tags (i.e. Single player, Multiplayer, Role-Playing) a solution can be to collapse all 3 attributes in the Tags column. This way it would be possible to only check for rows which have all 3 values as null-values and filter them.
  After checking which entry has all 3 attributes empty, 3302 rows will be removed from the dataset

- **Name:** The 6 entries which do not have a value for the Name attributed were checked manually using the AppID in the steam store. These games have a blank space as name in the store as well, for two of them the name is present in the description, but since all these games have a peak of concurrent players equals to zero , 5 of them have an estimated owner range of 0-0 and if they have any rating at all it is negative, the entries will be removed from the dataset

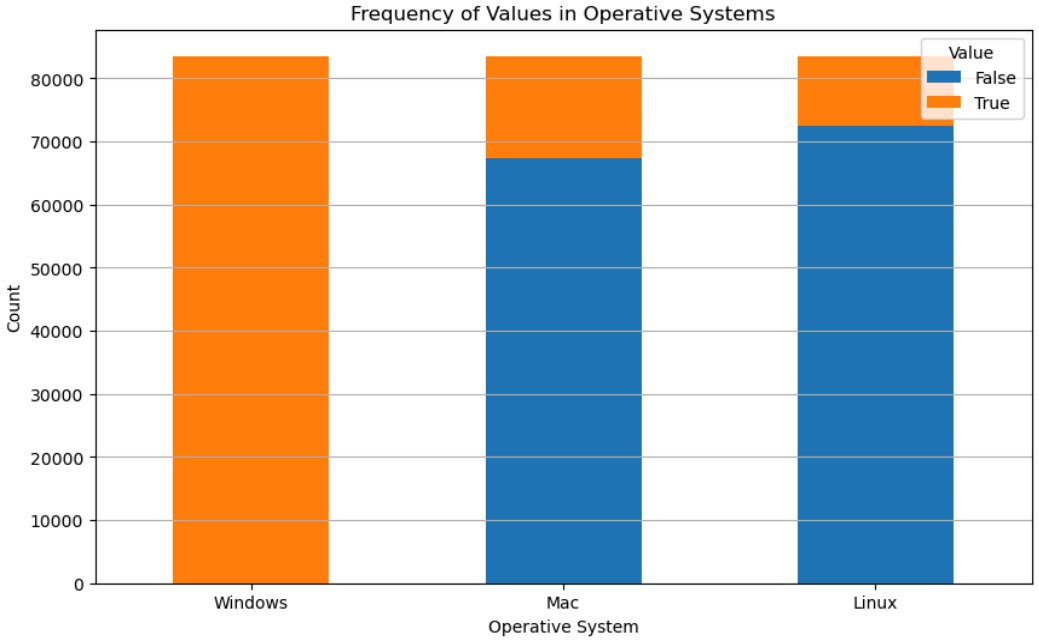
Meaningless Attributes

Some of the attributes in the dataset will have no use in the recommendation system, for this reason the following features will be dropped:

- **Required Age:** This attribute, while may be used to weight the recommendation based on the age of the user, is not very relevant, also if the decision to use an age restriction in the recommendation arises, the tags are enough to determine if a game is suitable below a certain age.
- **Full Audio Languages** and **Supported Languages:** In the Bar chart below the top 10 language frequencies are shown, the vast majority of games support the english Language and there is a big drop in frequency even for other widely spoken languages, for the scope of the project this attribute will be dropped and it will be assumed that all the games support the english language.



- **Windows, Mac , Linux:** As shown in the Bar plot below, only 19.41% of the games are runnable on IOS devices and 13.24% on Linux distros, while 100% of them are supported in Windows.



In the year 2023, only 1.63% of the users were playing on Linux machines and 4.09% on IOS devices (Data from Statista[2]), for this reason all 3 the Attributes will be dropped since it would be not worthy to weight the system based on the Operative system preferred.

- **Header image:** This attribute, containing the link to the cover image of the game, could be used in the final implementation of the recommendation system in the GUI, but for the process of the analysis it will be dropped.
- **Metacritic score:** In the gaming community it is widely accepted that scores from Metacritic are almost always controversial, weighing heavily in favor of well known and established publishers. For the scope of the project only the scores derived from user inputs will be used. This attribute will be dropped from the dataset.
- **Achievements:** The number of achievements available for a game is irrelevant in the recommendation system, the attribute will be dropped from the dataset.
- **Recommendations:** For this attribute, 69837 entries have 0 recommendations, this may be due to the fact that the recommend function is not commonly used on Steam, given the low number of non-zero values, this attribute will be dropped from the dataset.
- **Median playtime forever** and **Median playtime two weeks**: These will be dropped in favor of Average playtime
- **DLC Count**: For this attribute, 71798 entries have a value of 0. For this reason it will be dropped from the dataframe
- **User Score:** For this attribute, 83516 entries have a value of 0. Also it is not useful in the context of the application, since a weighted ratio between positive and negative votes will be used.

At the end of the Cleaning process, the dataset contains **13 Attributes** and **80254 Entries**. Note that for now the collapse of Tags,Genres and Categories was not counted in, as it will be performed during the transformation phase, effectively reducing the attributes to 11.

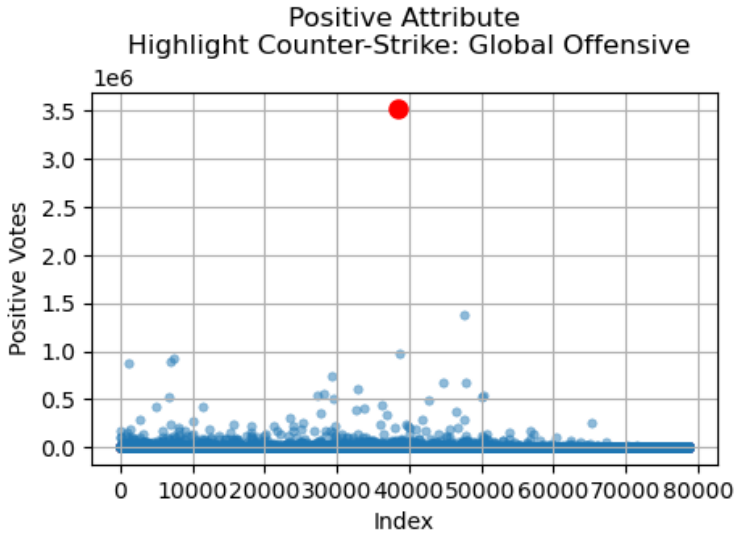
Outliers

Being the dataset about user rating of video games, it is expected to have some extremely popular games and some games which are not played at all. The high amount of votes for these games will heavily influence our analysis, for example, normalizing the attribute **Positive** with min-max normalization without handling outliers, we would get this result:

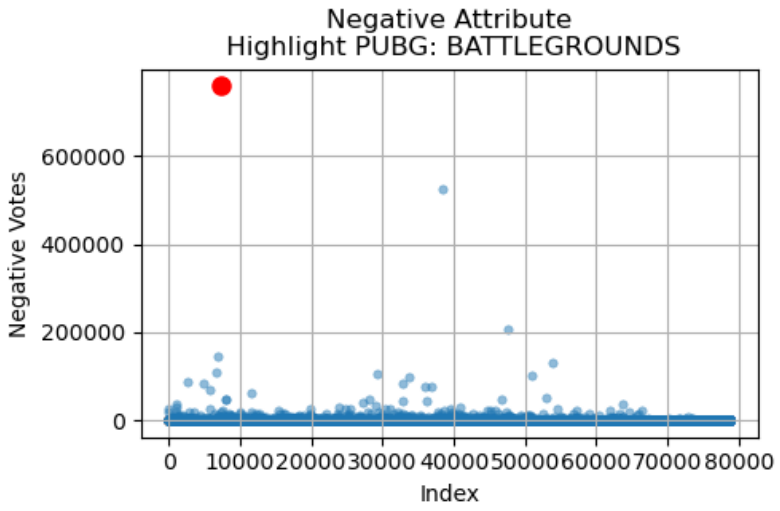
```
count    7.877400e+04
mean     2.554708e-04
std      5.172659e-03
min      0.000000e+00
25%      8.523388e-07
50%      3.125242e-06
75%      1.562621e-05
max      1.000000e+00
Name: Positive, dtype: float64
```

At least 75% of the data has extremely low values, further balancing of the dataset is needed.

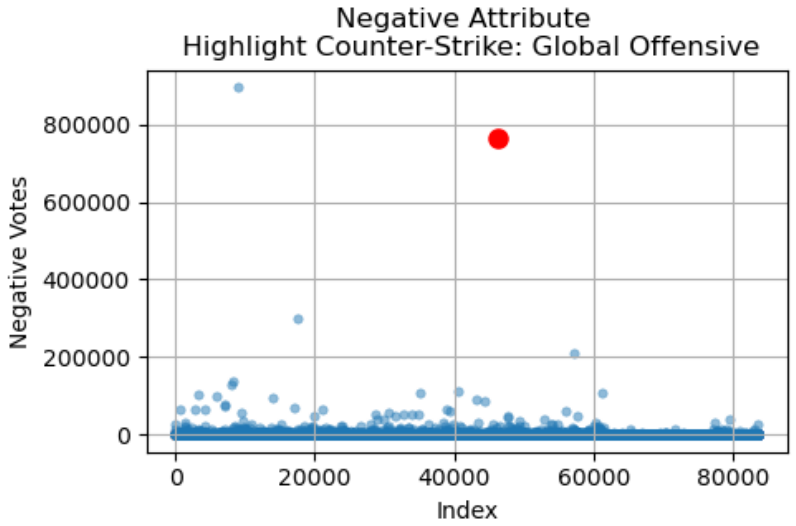
Highest Positive Game (Counter-Strike: Global Offensive)



Game with highest Negative votes (PUBG: BATTLEGROUNDS)

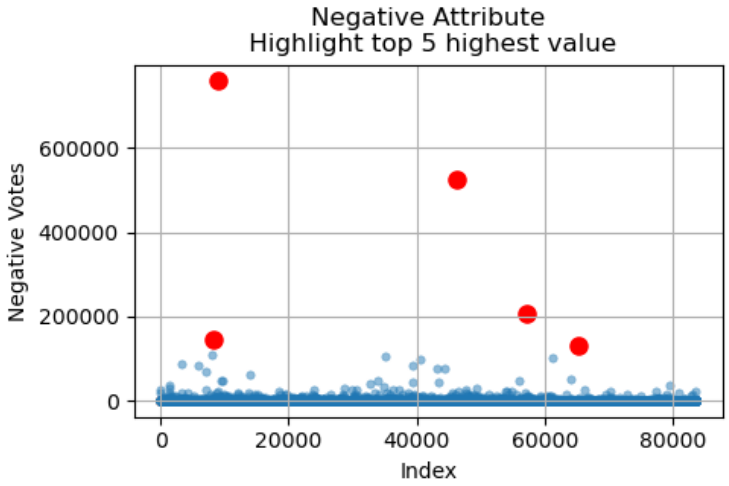


It is worth noting that the closest entry by value to **PUBG: BATTLEGROUNDS** in the Negative attribute plot is **Counter-Strike: Global Offensive**, showing that extremely popular games tend to have both positive and negative high values.



Being the aim of the project developing a recommendation system it is not possible to just remove entries higher than a threshold from the dataset, for example if we removed the games with more than 200k negative vote, the entries highlighted in the plot below won't

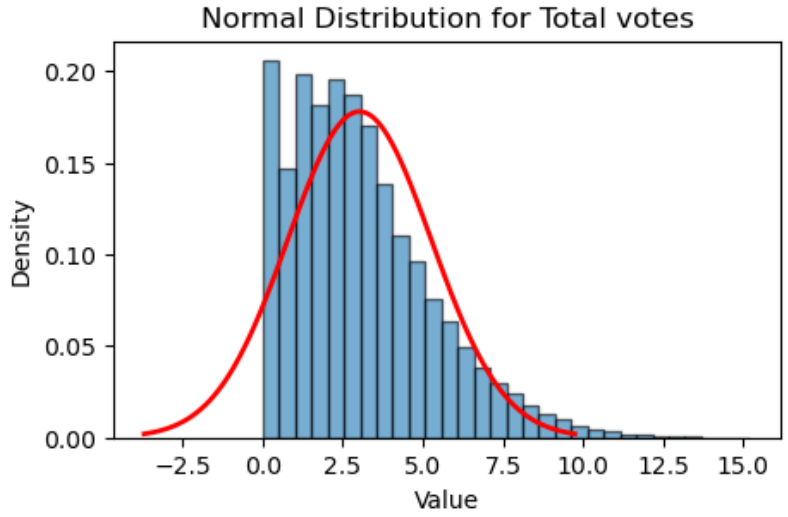
ever be suggested by the system:

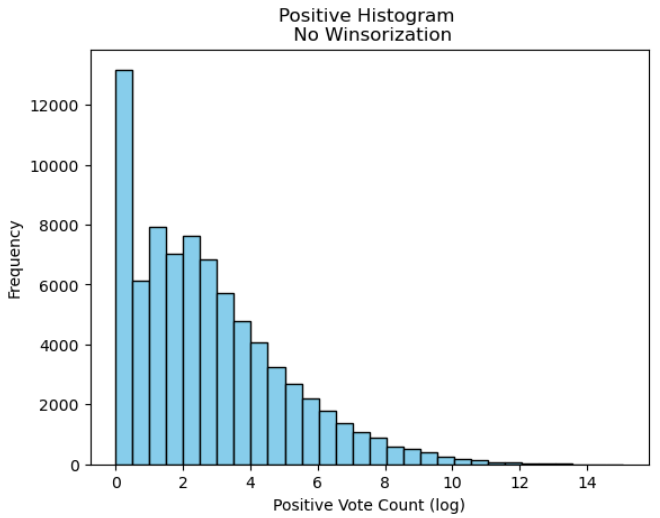


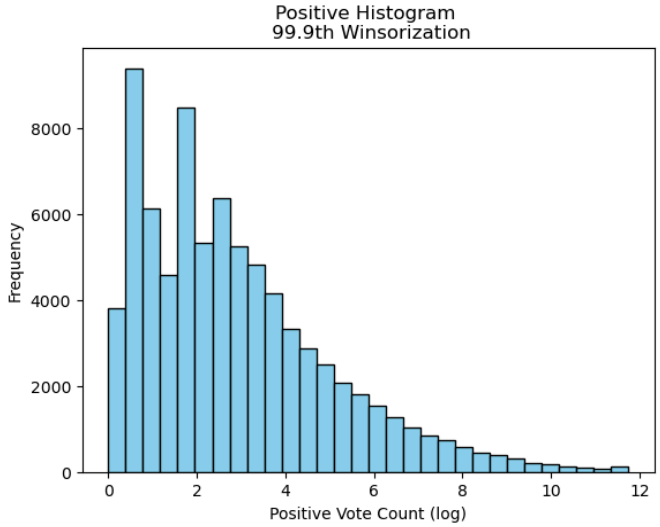
These 5 games alone have a cumulative ownership higher than 225 - 550 Millions of players and are very different in tags , so it would be very likely for at least one of them to show up in the recommendation results.

One approach to bring these entries to a comparable value could be using Winsorization, which caps and sets the lower and upper extreme values to a specified percentile of the data.

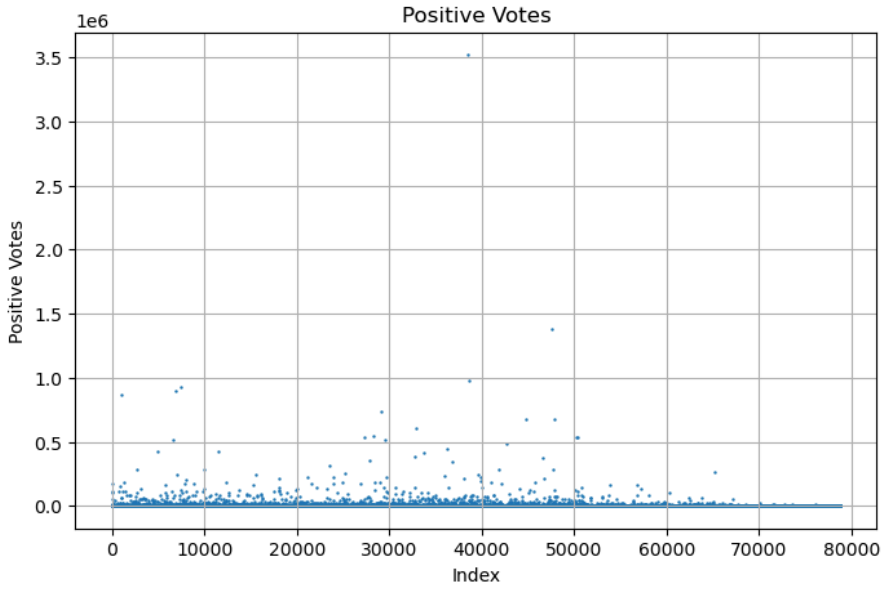
Since the dataset has maximum density at lower values as shown in the plot below, only the top 1% values will be capped.

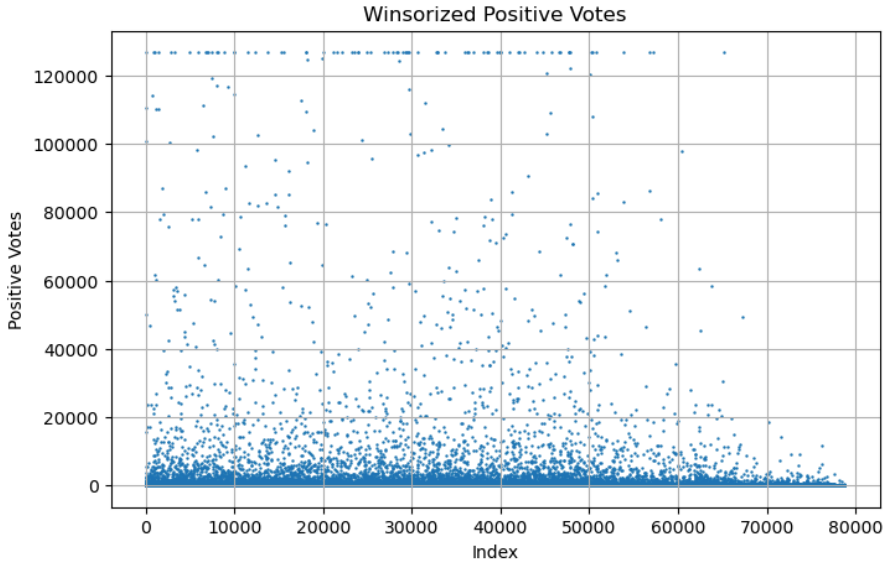


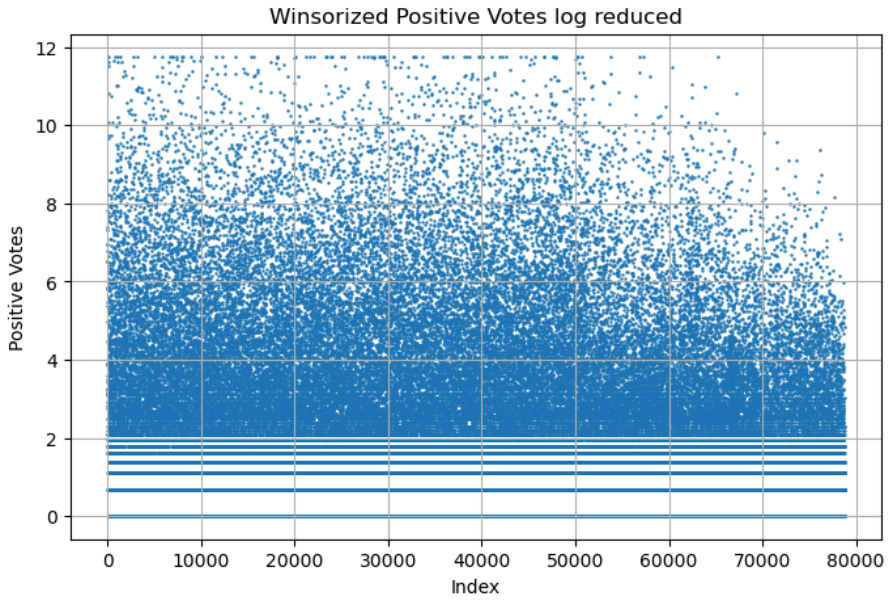




By capping the top 0.1% values at the 99.9th percentile value (31336) the standard deviation was reduced from 25083.14 to 2848.61


Positive Votes
1e6
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Positive Votes
0
10000
20000
30000
40000
50000
60000
70000
80000
Index


Winsorized Positive Votes
120000
100000
80000
60000
40000
20000
0
Positive Votes
0
10000
20000
30000
40000
50000
60000
70000
80000
Index


Winsorized Positive Votes log reduced
12
10
8
6
4
2
0
Positive Votes
0
10000
20000
30000
40000
50000
60000
70000
80000
Index

Note that the straight lines are probably due to very frequent low values for the positive attribute, below the top 5 most frequent values for the positive attribute are presented.

```
Positive
1    9383
2    6130
3    4567
0    3794
4    3371
Name: count, dtype: int64
```

The same process is applied to **Negative, Peak CCU** and **Estimated owners**

Transformation

The **Estimate Owners** attribute contains a string value with a range between two numbers, to normalize this attribute the average of the two values will be used instead

The **Positive** and **Negative** attributes related to the user votes are combined in a single attribute called **Net Votes** which is the difference between positive and negative votes.
Since the data is not balanced and there are some games with a huge number of votes and others with almost zero, it is necessary to create a weighted value attribute.
As shown in the picture below, simply using the ratio of the positive and negative votes is heavily biased towards games with lots of votes.

Exploratory Data Analysis

	Estimated owners	Peak CCU	Price	Positive	Negative	Average playtime forever	Average playtime two weeks	Votes zscore	Net Votes
count	80254.00	80254.00	80254.00	80254.00	80254.00	80254.00	80254.00	80254.00	80254.00
mean	90859.15	141.87	7.49	1016.26	169.22	111.03	11.30	-0.00	847.04
std	1057809.77	5561.84	12.47	25083.14	4710.33	1176.15	194.41	1.00	21589.01
min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-3.14	-66917.00
25%	10000.00	0.00	0.99	1.00	0.00	0.00	0.00	-0.04	0.00
50%	10000.00	0.00	4.99	8.00	2.00	0.00	0.00	-0.04	4.00
75%	10000.00	1.00	9.99	52.00	16.00	0.00	0.00	-0.04	32.00
max	150000000.00	872138.00	999.98	5764420.00	895978.00	145727.00	19159.00	231.46	4997743.00

Estimated Owners:

- The mean value indicates that on average games are owned by ~90k users, but a standard deviation of 1,057,809.77 indicates that ownership numbers vary widely across different games.
- The 25th percentile value suggests that 75% of the games have more than 10.000 owners, but the value stays the same up to the 75th percentile, indicating that only 25% of the games have a large number of players.

Peak CCU:

- The mean value of the Peak Concurrent Users of 141.87 and the std of 5561.84 also imply that there is a wide variation between games in the dataset. The max value of 872138, which is 6141 times higher than the average is a sign that there are few really played games and many games without a player base as well.

Price:

- The maximum value of 999.98$ is very distant from the standard deviation of 12.47, after further investigating the game is “The Leverage Game”, which has 0 Peak CCU, 0 votes and 0 avg playtime forever. Being the Steam Store quite open to indie developers it is not surprising to see outliers from developers putting a game at insane price.

Positive and Negative votes:

- The average values of the Positive votes is almost 10 times higher than the Negative one, this may be due to the bias of being more prone to go back in the Game Shop page and upvote/review a game someone enjoyed while if an user did not like the game it would be more likely to just close it and play something else.
- The standard deviation of the two attributes reflect the same wide difference gap in games that was noted in the previous attributes.
- The extreme values of max Positive and Negative suggest the presence of some games with very high engagement (both positive and negative)

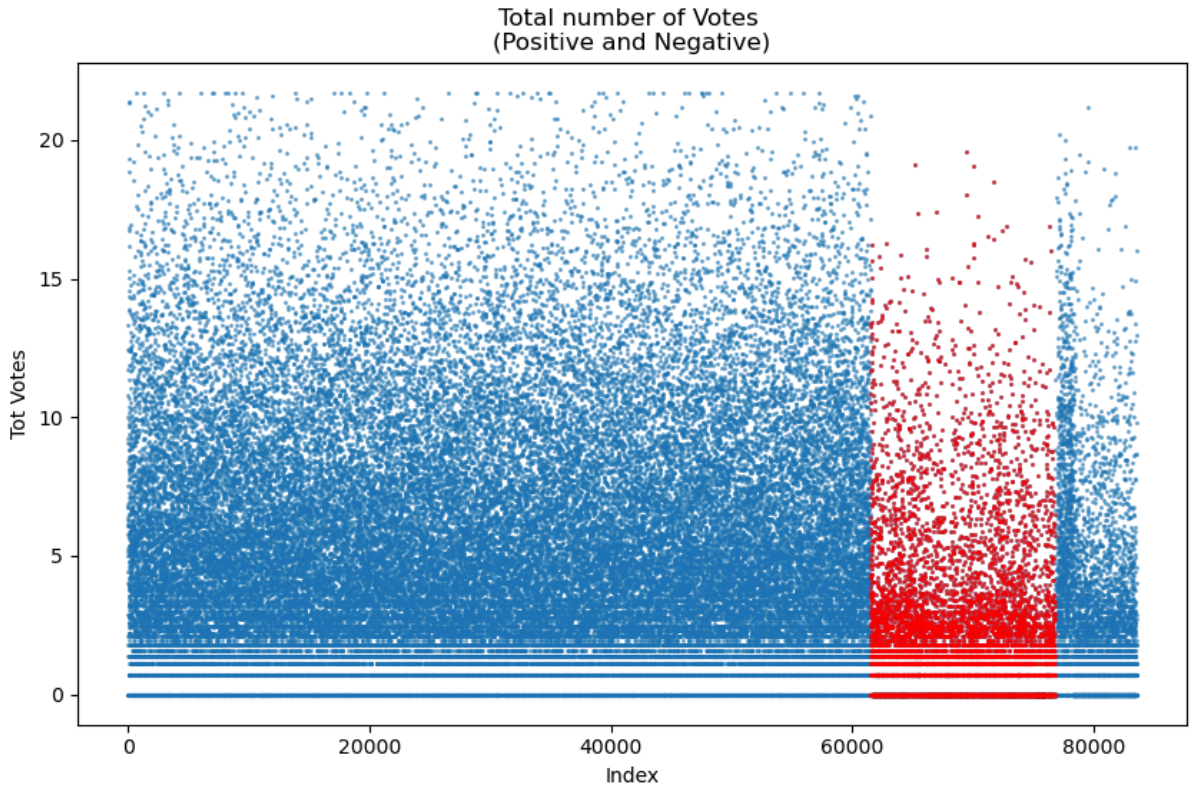
Z-score and Net Votes:

- As expected the z-score mean is 0 and the standard deviation is 1, but the min value of -3.14 and the max value of 231.46 suggest that there are some extreme outliers in the data.
- The net votes mean and standard deviation highlight the disparity in net approval of the games.

The high standard deviation across all the attributes indicates a significant spread in the data, with potentially multiple outliers.

Drop in total votes between the index mark 60000 and 74000

There is an interesting drop in number of votes between the index 60.000 and 74.000



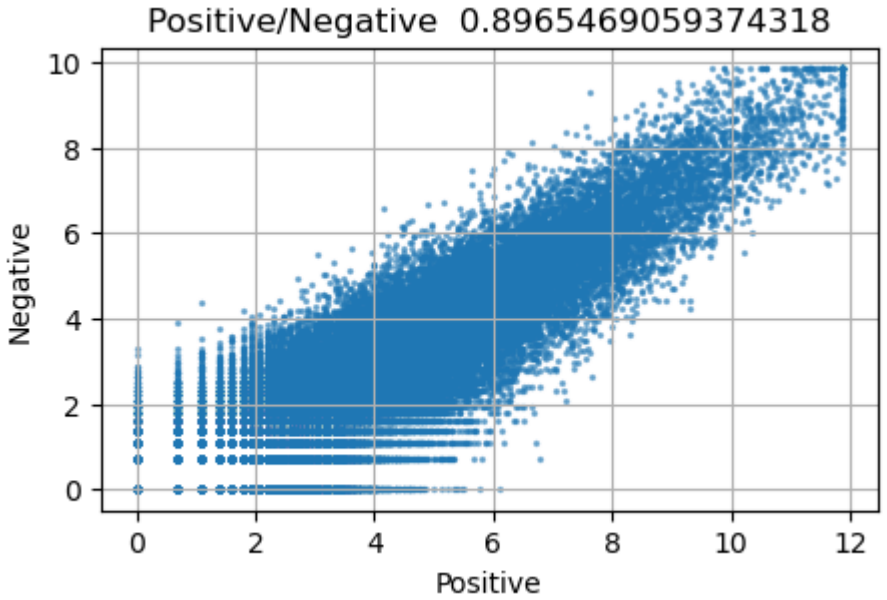
Since the difference between 60000 and 74000 is consistent and considering that the index ordering of the original dataset was not changed, this may be due to changes in data collection methods or the website scraped for that specific amount of time. This is the likely cause since after analyzing the stats of both the blue and the red part nothing relevant emerged, only differences due to the variation in number of entries considered. It is worth noting that for the amount of time entries from 60.000 to 74.000 were collected, there was a significant decrease in total votes and consequently in other attributes depending on the number of players.

Correlation Analysis

Pearson Coefficient for each pair of attributes was calculated and only attributes with a coefficient >0.5 or <-0.5 were added to a list. In the table below the list is presented in descending order.

Attribute 1	Attribute 2	Pearson Coefficient
Positive	Negative	0.90
Estimated owners	Net Votes zscore	0.77
Negative	Average playtime forever	0.71
Positive	Average playtime forever	0.69
Peak CCU	Positive	0.62
Peak CCU	Average playtime two weeks	0.60
Peak CCU	Negative	0.59
Peak CCU	Average playtime forever	0.57
Negative	Days gone	0.55
Positive	Days gone	0.50

Positive - Negative 0.90 : As expected, there is a positive correlation between the positive and negative attribute, the higher the positive vote count, the higher the popularity thus higher the likelihood of the game to have more negative votes.



Estimated owners - Net votes 0.77: Since the net votes directly reflect the popularity of a game, it makes sense that the ownership of a game increases with the number of votes.

Negative - Average playtime forever 0.71: This came as a surprise, it was expected for the positive attribute and to lesser extent to the negative one, but the correlation coefficient seems higher for the negative - avg playtime forever pair. It is not rare to find people with 5000+ hours of playtime writing a very short review such as “bad game” and this became a

popular joke some years ago, this could be one of the reason of this correlation, since the higher the popularity of the game, the higher the number of negative votes, the higher the likelihood of an user to put a joke review.

Positive - Average playtime forever 0.69: As stated in the previous pair, this was expected: the more a game is liked, the more it is played.

Peak CCU - Positive 0.62: Popular games tend to have higher peaks of concurrent players.

Peak CCU - Negative, avg 2 weeks, avg forever: All factors that increase with the popularity.
Days gone - Negative 0.55, Positive 0.50: The longer the game has been on the steam store, the higher the number of votes.

Covariance Analysis

Covariance Coefficient for each pair of attributes was calculated and only attributes with a coefficient >0.5 or <0 were added to a list. In the table below the list is presented in descending order.

Attribute 1	Attribute 2	Covariance Coefficient
Positive	Negative	4.14
Positive	Average playtime forever	3.45
Negative	Average playtime forever	2.93
Peak CCU	Positive	2.07
Peak CCU	Average playtime forever	1.70
Peak CCU	Negative	1.61
Positive	Average playtime two weeks	0.70
Average playtime forever	Average playtime two weeks	0.69
Peak CCU	Average playtime two weeks	0.67
Negative	Average playtime two weeks	0.57
Price	Days gone	-0.00

Most of the pairs with positive covariance value were already discussed and justified in the correlation section, an interesting addition is the **price** and **days gone** pair. It is the only result with negative (even if really small) value.

It makes sense that with time, the price will tend to be lower.

Data Integration

Given the unbalance of the dataset and that it was collected years ago, the tags **‘Name’, ‘Genres’, ‘ Categories’, ‘Positive’, ‘Negative’, ‘Peak CCU',** were updated using the Steam API.

The goal is to reduce the number of entries with 0 values and empty tags, which may be due to the date of the collection or the method of collection.

The complete list of Steam games (180k games) was retrieved through API and information for each game not in the original dataset will be retrieved. Games without activity (0 CCU and positive+negative = 0) will not be considered during the retrieval stage.

By merging the games with a non-zero number of total votes from the Kaggle dataset with the new dataset the total number of entries is 78774.

At this point, the attributes relative to Average playtime and Estimated ownership had not enough variance or were not enough distinctive to be used in the model, furthermore these attributes could not be collected by the use of Steam APIs during the update process, relying on SteamDB, a website not affiliated with steam of which the robot.txt does not allow scraping. Data access for that specific data was asked, as it is stated in the FAQ that it may be requested for academic purposes, but the access was denied. From this point onward the attributes **Average playtime** and **Estimated ownership** will be dropped for all games.

The attribute ‘Tags’ (From Kaggle dataset) was also not available in the JSON returned by the API, but since it was redundant with Categories and Genres, only the latter 2 will be used.

Data update

For entries already present in the Kaggle dataset, the following update criteria were applied.

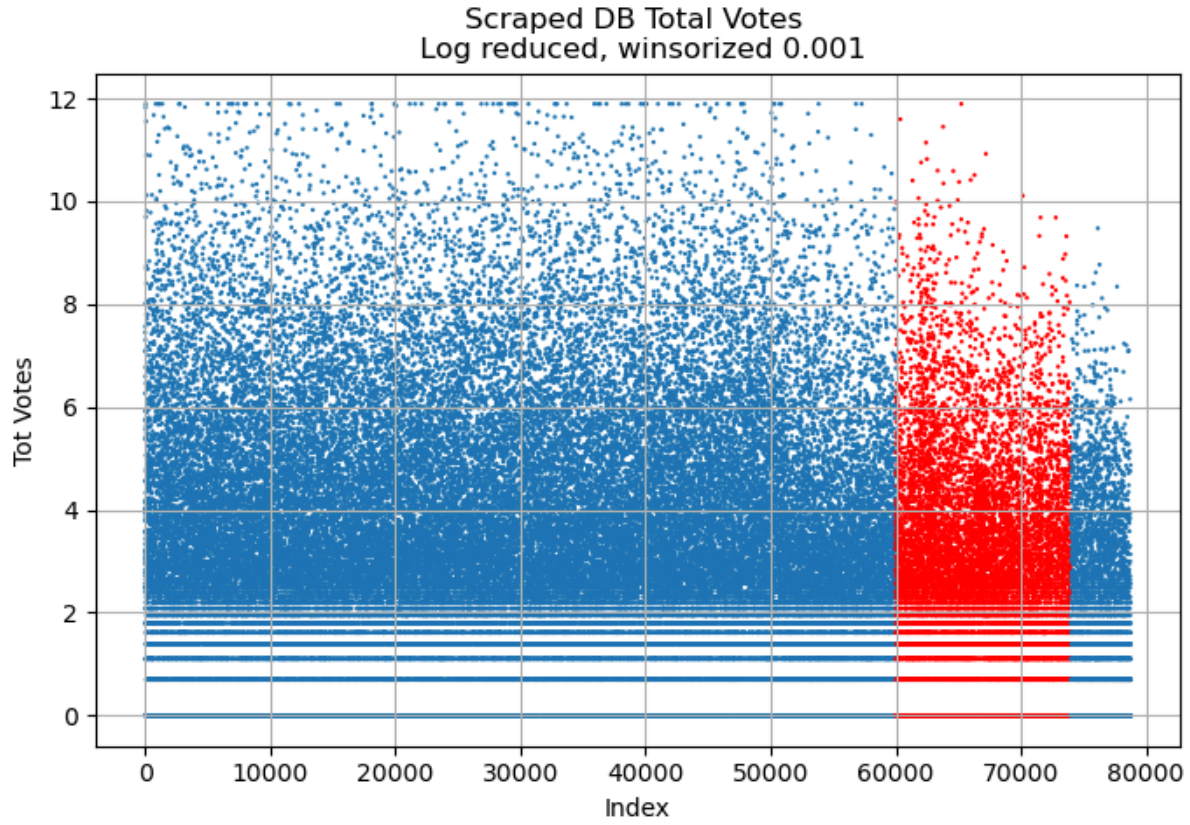
The **Name** attribute is updated since some of the games changed their name during the years, some had placeholder special characters and others didn’t have a name at all.

The **Genres** and **Categories** are updated to fill the empty cells.

Positive and **Negative** update is useful to both move away entries from 0 values and to have more reliable values at the present time.

Peak CCU is updated only for entries which have 0 Peak CCU, the steam API does only allow to get the current number of concurrent players, for statistics on this attribute a collection over a long period of time would be needed (Steam API only allows to retrieve CCU at runtime, not peak). It is available on SteamDB website which collected data multiple times a day for 11 years, but they do not provide an API. For these reasons, Peak CCUs of

value equal to 0 are updated at the number of active players at the time the update algorithm was executed.



After the update of the dataset, the drop in Total votes between 60.000 and 74.000 is not noticeable anymore, further sign that it was probably due to the change in scraping methods or malfunction of the scraping algorithm during the time collection of those 14k entries.

Dimensionality Reduction

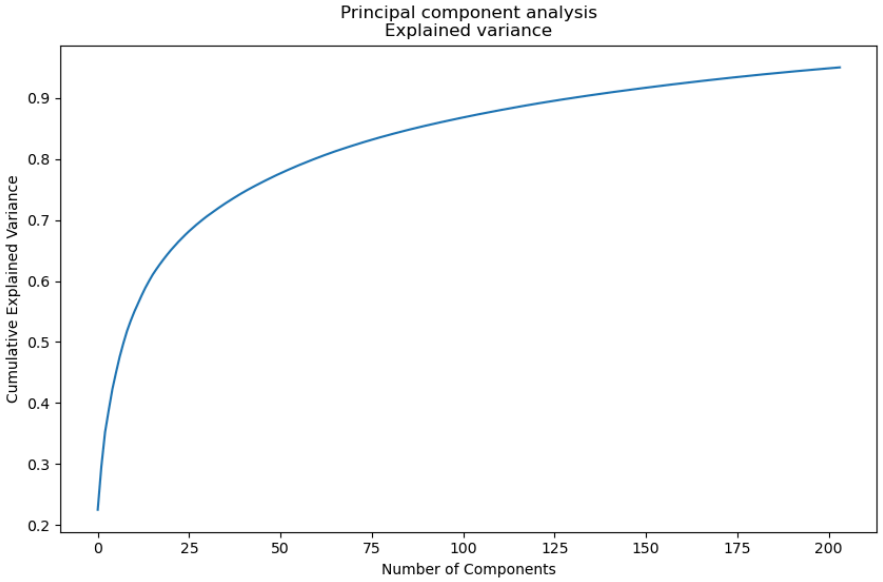
Having a total of 498 features after the encoding of tags, some dimensionality reduction is required in order to boost the model performance and potentially increase its accuracy. Since some of the features are not centered around 0 and for principal component analysis we need them to fall within the same range, these features are standardized to a mean of 0 and a standard deviation of 1, putting them on a similar scale of the tags, which are already in binary.

At this point **Positive** and **Negative** can be dropped from the df, using the **Net Votes** instead.

Price will also be dropped, since the correlation with other features is too low and it looks like that pricing of games is more subjective to what the developer decides rather than

having a statistical significance in the context of a suggestion system (i.e. Many Indie games price well over the 100$ mark or very popular games priced at 10$ or even free).

Principal component analysis: An attempt in using PCA to select relevant features, setting the variance at 95% the features resulted in a cut from 498 to 204.



Radical changes to reduction method

After some tests, it became clear that the variance in a dataset composed predominantly by binary data is not enough to allow PCA to select significant components, thus this approach was dropped in favor of a more “hands on” method.

The following steps were applied to data:

Use only Genres instead of a combination of Genres and Categories:
Genres are way more distinctive than categories for a recommendation system; the difference between a game with tags [Sports,Action] and another with [Roleplay,Action] is higher than between categories such as [Multiplayer, Co-op] and [Multiplayer,VR]. By using only Genres the number of attributes becomes way lower.

Cleaning of game genres:
A further cleaning process was applied to the genres, removing punctuations and special characters.

Translation of game genres:
For some of the entries, the language of the json returned by the api was not english, even when the tag *l=en* was specified in the URL during scraping, for this task Deep Translator from Google Translator was used. Note that the model is non-deterministic, so further handling of tags is needed after the process to collapse the attributes that were still in different languages/mistranslated.

Compute similarity score:
A similarity score was computed between tags and if the value was over a set threshold, the tag in consideration was replaced by the main tag in the loop. This was needed to reduce the number of unique tags shifting something like “Role play game” into “Roleplaying”

Manual substitution:
The last step was performed manually, 39 tags with very low frequency were still present after the process. These tags were leftovers of the translation process. Since a language model was used to perform this task, being a non-deterministic model, the correction of these tags had to be performed manually in order to prevent the possibility of having even more mistranslation on a second execution.
Below there is an example of these substitutions

'estrategia' → 'strategy'
'for free' → 'free to play'
'strategies' → 'strategy'
'massively multiplayer' → 'mmo'

After these steps, the number of tags was reduced to 38, a way more manageable number than the previous 498.

Dataset shape after Reduction

Before clustering, the dataset attributes are 2 continuous attributes **Peak CCU** and **Net Votes** and 38 binary attributes derived from **Genres**

Cluster Tendency

Before proceeding further, the cluster tendency of the dataset needs to be assessed, in order to verify that the dataset does not have a random structure.

Hopkins Statistic for cluster tendency was computed for the dataset, resulting in a value of

H=0.9999999994849823

This suggests a cluster tendency at more than 90% confidence (H > 0.75)

Clustering Model

Different clustering methods were tested with the dataset, in the following section will be discussed what gave satisfactory results and what was discarded along the way.

Spectral Clustering

A spectral clustering was implemented to reduce the dimensionality of the dataset by transforming each tag in a **Word2Vec** vector and applying the **Ng-Jordan-Weiss (NJW)** Algorithm.

Spectral clustering was used before splitting the tags in standalone attributes, right after the cleaning and reduction phase, the word embedding of each tag was computed and then a naive approach was used with the average of all the tags relative to a game being taken as a single value.

The idea behind it was that the average resulting vector of games with identical or almost identical tags would be the same or fairly similar. (*i.e. the vector of two games with a tag list ['casual','indie'] would be the same*)

Given the RAM needed to compute the affinity matrix for the whole dataset (68gb) the model was trained on a sample of 10000 entries.

By using K-mean for clustering, the results were good only empirically (silhouette score was way lower than other models used and varied wildly based on the sample set used), returning games quite similar but not enough to justify the use of this method or the loss of global context.
This may be due to the sample size, not being able to capture the whole structure of the dataset.

This method was discarded since the leading eigenvectors are computed on the global affinity matrix and using a sample of the dataset would lead to the loss of global context.

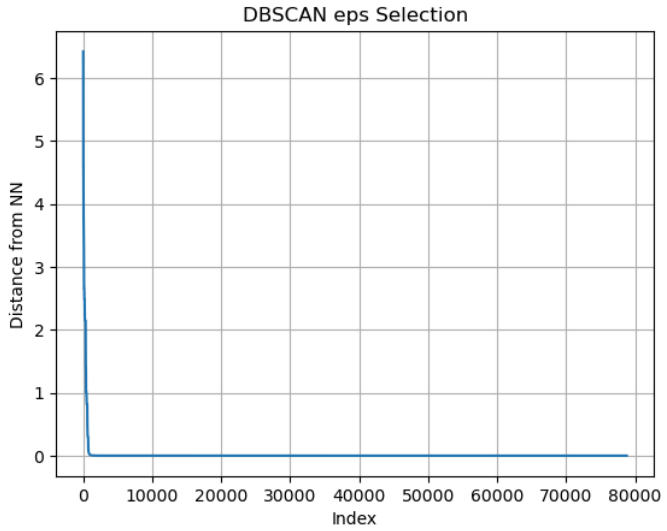
Furthermore, this approach would not be well suited in the case of an input not present in the dataset, in case of a feature implementation which allows the recommendation system to update the dataset with new non-seen entries.

DBSCAN

DBSCAN, being a density based clustering algorithm, is not well suited for high dimensionality datasets, but since the dimensionality of the dataset was greatly reduced from the initial state, there was a trial run to test how the algorithm may have performed.

The plot for the n_neighbour distance results in a fast dive and then a flat line, for different values of n_neighbour. This may be either due to the high dimensionality of the dataset or due to the nature of the dataset, where data points are very densely packed, so the

distances to the nearest neighbors is similar for most points (flat line) and very high for the few outliers of the dataset



.

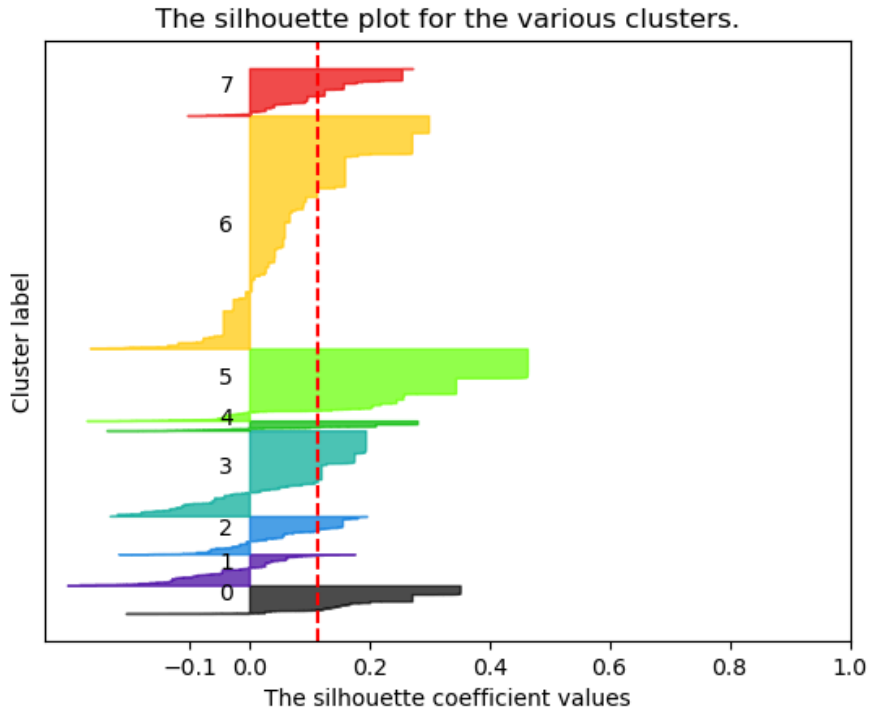
Different parameters were tested for DBSCAN, but none yielded satisfactory results.

Birch

Hierarchical models are well suited for clustering in high dimensional data, for this reason Birch was tested.

Birch with Genres, Peak CCU and Net Scores

The number of clusters was determined by computing the average silhouette score at different combinations of number of clusters and threshold. For a dataset containing Genres, Peak CCU and Net Votes as attributes the best silhouette score and balance of data was found at **N° Clusters = 8** and **Threshold = 0.7** for an average silhouette score of 0.1150



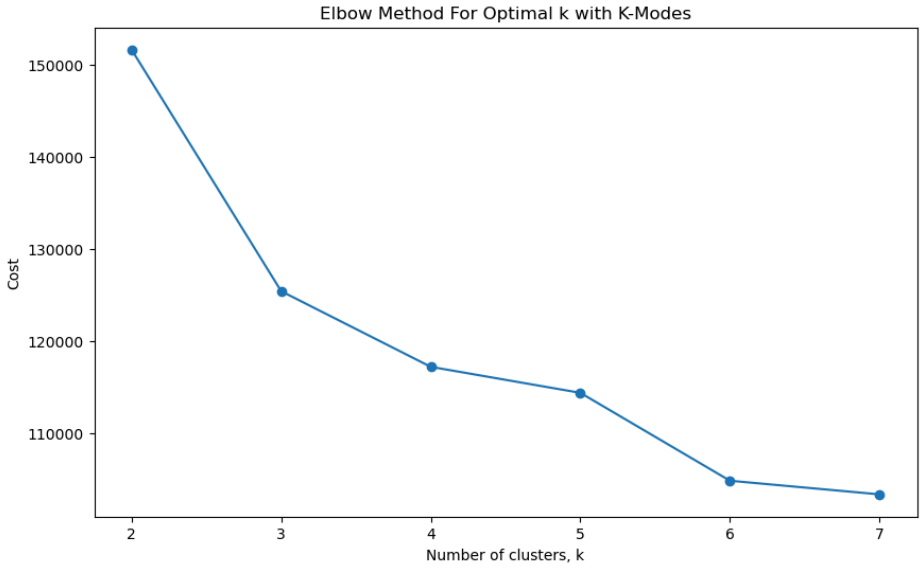
The issue with this approach is that the model is handling both binary data and continuous values, therefore there is no cohesive clustering as shown in the silhouette plot.

A silhouette score this close to 0, may indicate that some clusters are overlapping.

Models with only Genres

By removing the attributes Peak CCU and Net Votes, using only binary attributes for the clustering process, the silhouette of the clusters improved.

K-modes



For k-modes, Cost was used instead of Nearest neighbor distance to draw a plot for selecting the best k. There is not a clear elbow, there are two points where there is curvature, 4 and 6, but even at those values, the results were not satisfactory. After k=7 the line becomes flat.

K-modes was implemented on the dataset including only genres of games and tested at different k, but the best result obtained was 0.17 silhouette score.

Agglomerative clustering

Agglomerative clustering was tested on the dataset with only genres.

Single-linkage and complete-linkage achieved almost the same results as K-modes, average linkage instead generated only a single cluster.

Agglomerative clustering from sklearn does also have a “ward” linkage which could not be tested on the binary dataset since it is based on the Euclidean distance, which would not be suited for the binary nature of the data.

Spectral Clustering + K-means and Random Forest

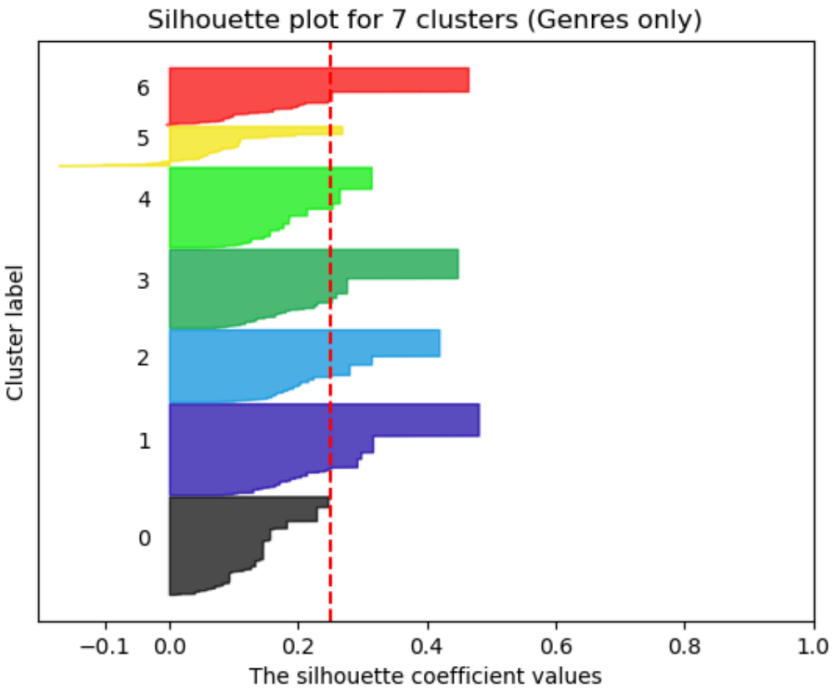
After the unsatisfactory results from the other models, the following approach was tested:

Since the approach with spectral clustering was tested on word vectors, it had yet to be tested on the binary only data, so it was selected for this test.

Since the affinity matrix can’t fit the memory if computed on the whole dataset, a sample of the size of 5% of the dataset was used instead.

An algorithm sampled the dataset multiple times and applied the clustering process, with different parameters. The best run was selected for the clustering model.

The final model used **hamming** as a distance metric, k=7 and a sample of approximately 5% of the dataset, yielding a silhouette score of 0.256.



While the silhouette score is not high, it doubles the result obtained with BIRCH and it is higher than the result obtained with k-modes.

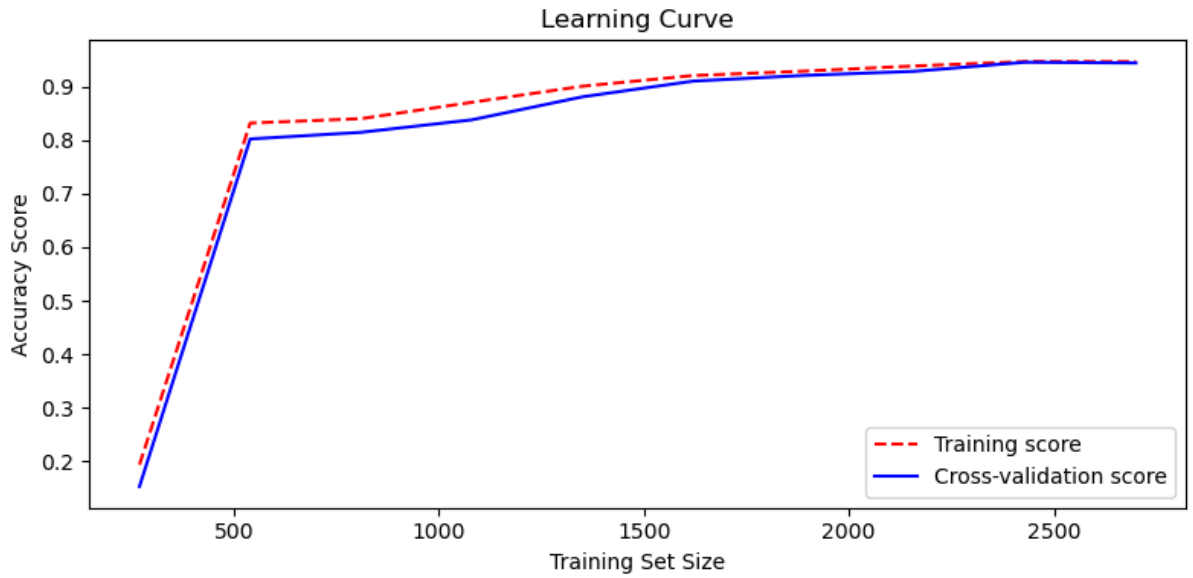
Note that, for the clustering part of this process, the attributes with frequency lower than 1000 were left out, in order to reduce the dimensionality of the dataset and focus on the most frequent and significant Genres. 14 Genres were used for the clustering process.

The clusters have the following distribution of entries [570 529 420 457 466 228 330],
Cluster 5 is the one containing the least entries and the ones with highest misclassifications, with a portion of the cluster having negative silhouette score. This may be caused by outliers in the dataset, that in the case of binary attributes such as genres could be unique combinations which the clustering could not fit in any of the other clusters.
Other clusters have multiple entries at low silhouettes, indicating possible overlapping.

The sampled dataset, now labeled with the 7 clusters and providing a ground truth, was then used to train a Random Forest Classifier, which was used to label the rest of the data with the respective cluster.
To train this model, all the attributes were used, expecting the classification algorithm to find further associations between the labels and the complete tuples.

The random forest model obtained the following performance for the best combination of parameters:
Model F Measure: 90.25%
Model Accuracy: 95.00%

The statistics for the cross validation are the following (10 fold cross validation)
CV Mean Score: 95.83%
CV Standard Deviation: 0.96%



For multiple combinations of parameters, there seems to be some degree of overfitting, this run was the one with the lowest difference. All runs presented a plateau at the end of the plot, suggesting that adding more data would not significantly improve the model, which was the main concern since a sample size of 5% was used.

The sharp increase at the beginning may be due to the nature of the data used, being real world high dimensional binary data the model may have difficulties in capturing the underlying patterns in the data, so as more entries are introduced, the model is able to understand the patterns, thus the sharp increase.

The cluster distribution is the following:

Cluster 0 →17791
Cluster 1→15038
Cluster 2→10888
Cluster 3→12125
Cluster 4→12506
Cluster 5→1402
Cluster 6→9024

Cluster 5 is the one with the fewest entries. Other clusters have a well defined set of game genres in them, for example:

Cluster 1 contains prevalently “casual” games,
Cluster 2 contains prevalently “Adventure” games,
Cluster 3 contains lots of “Simulation” and “Sport” games, with an high number of games with both “Indie” and “Action” genres together
Cluster 4 contains prevalently “Indie” games,
Cluster 6 contains prevalently “action” games

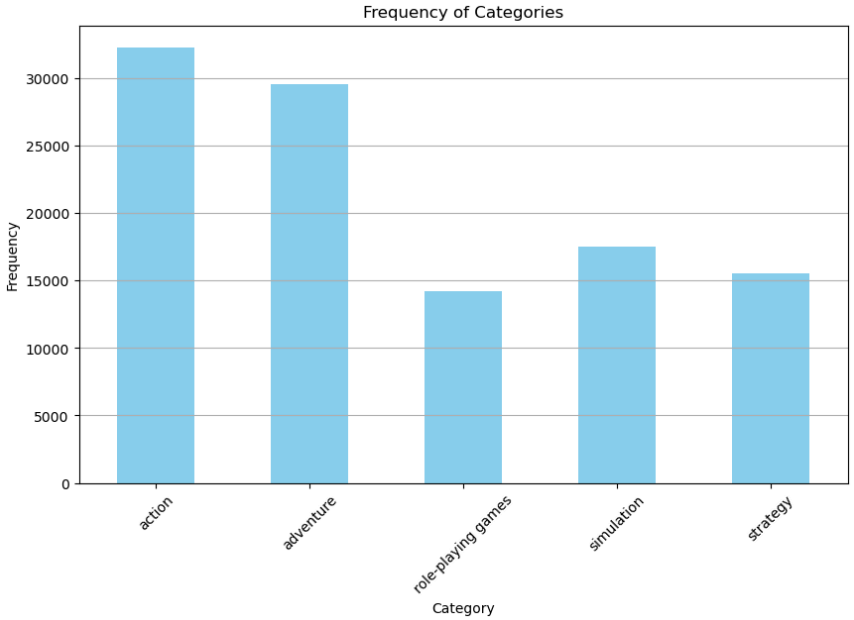
Cluster 5 instead, doesn't have a prevalent type of game at first glance. From the silhouette of the initial cluster before applying random forest, it was also the one with the lowest score and it seems almost like it ended up being the Rag Bag cluster.
To be noted is that the frequency of each cluster after the classification, was really close in proportion to the frequency of the clustering alone, below the proportion of each cluster to the dataset size (Or sample size in the case of k-mean):

Method	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
K-means	0.19	0.176	0.14	0.152	0.155	0.076	0.11
RF	0.226	0.191	0.138	0.154	0.159	0.018	0.115
Difference	0.036	0.015	0.002	0.002	0.003	0.058	0.005

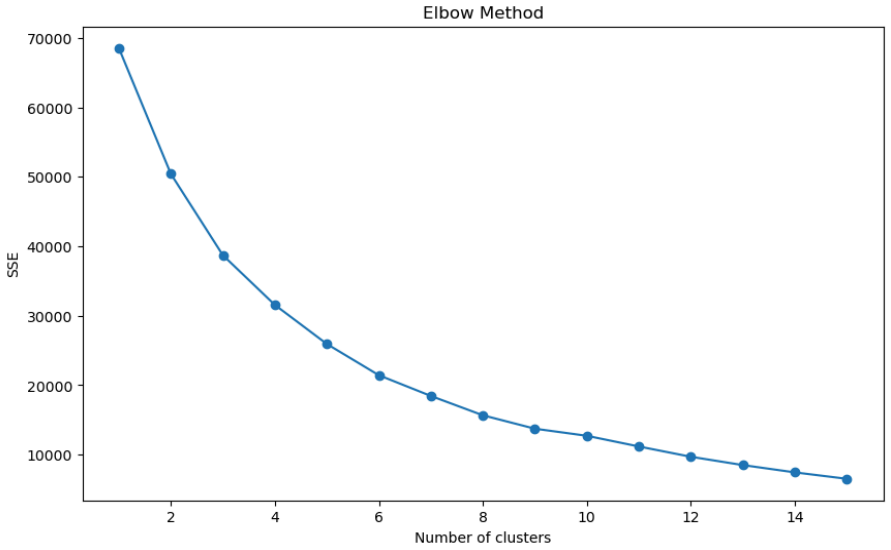
This further indicates that the sample was distinctive enough and that the classifier was able to detect patterns in the clustered sample and apply the knowledge on the entirety of the dataset effectively.

Final model - K-Means

The final model used, employs K-Means over the dataset with dimensions greatly reduced. Since most of the attempts on high dimensional dataset were not satisfactory, only the main (most frequent/common) tags were used in the clustering process, specifically: "RPG", "Action","Strategy","Simulation" and "Adventure".

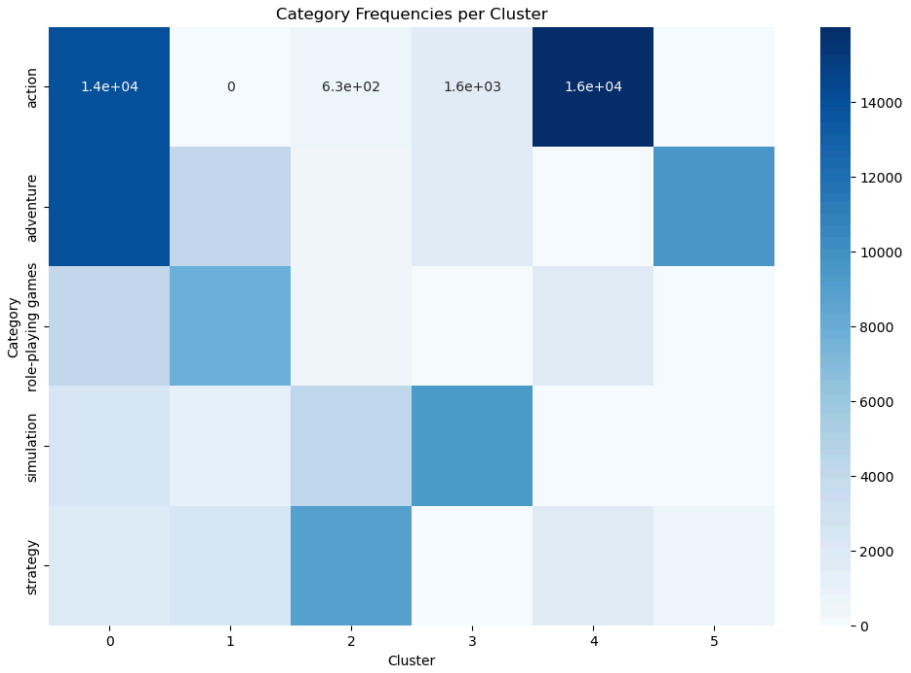


The frequency of these tags is shown in the histogram below:
After plotting the sum of squared errors up to a k value of 15, the algorithm was tested over k values between 5 and 7 included.The value k=6 was the one which produced the best result, thus the one used in the final version of the model

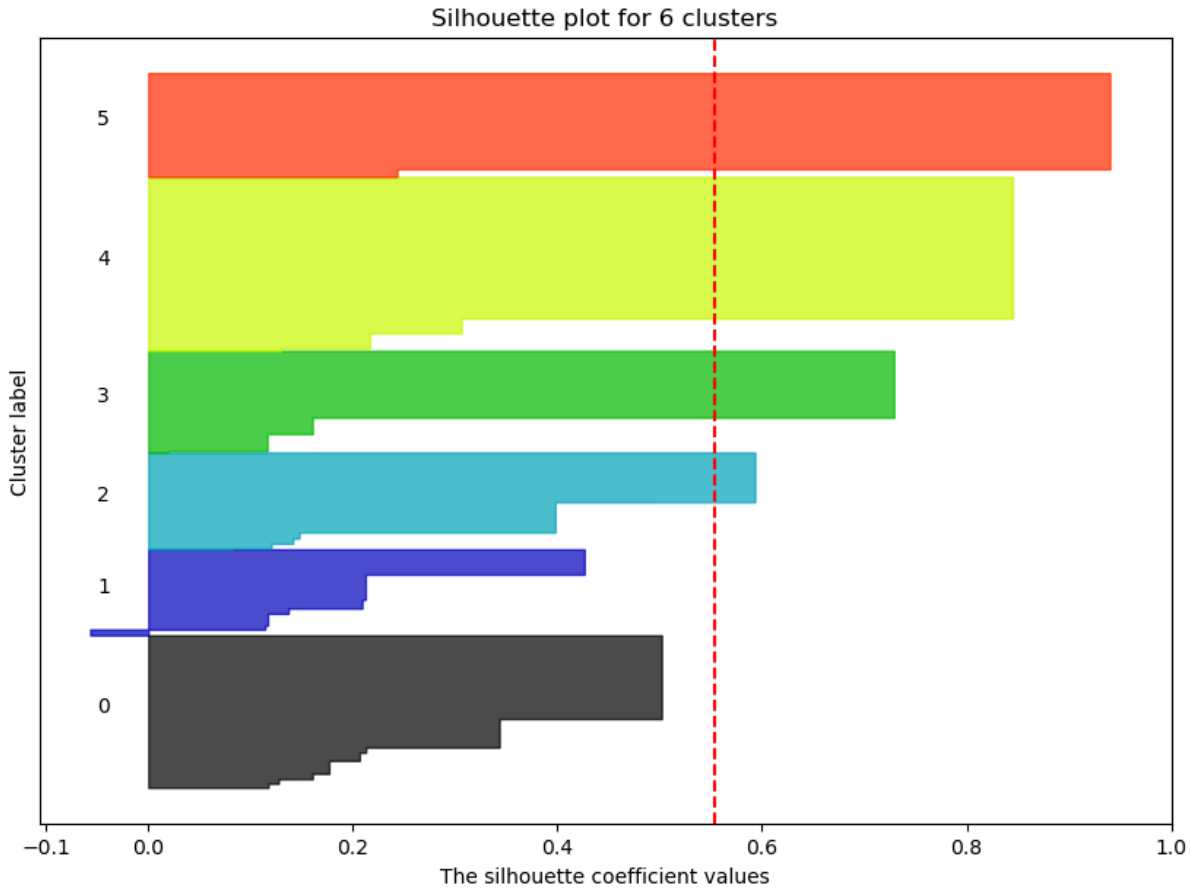


After K-means, the distribution of the elements in the produced clusters is as follows:

```
Cluster 0: 14022 elements
Cluster 1: 7907 elements
Cluster 2: 8900 elements
Cluster 3: 9357 elements
Cluster 4: 15996 elements
Cluster 5: 9542 elements
```



Looking at the heatmap, k-means managed to assign to each cluster elements predominantly of a single class, or in the case of cluster 0, elements which had both the adventure tag and action tag (This cluster makes sense since an high number of action games are adventure games as well generally)



The final silhouette score for the clusters is 0.554, which some cluster going over 0.8.
The low silhouette of cluster 1 may be due to the fact that many RPG games fall in the "adventure" genre, so the negative score of some elements could be due to the fact that those games are closer to the cluster 0 or cluster 4 than the one that was assigned to them.

Results

A GUI was developed for the application using Tkinter in python, the user interface includes a textbox for the input AppID, which is the AppID of the game the user wants recommendations of, a Recommend button, to execute the search and a listbox of results, which are double-clickable to open the Steam page of the clicked game.

For games that are already in the DB, the system checks if a cluster is already assigned to the inputted game, if not the RandomForest algorithm will assign a cluster to it and proceed with the recommendation process.

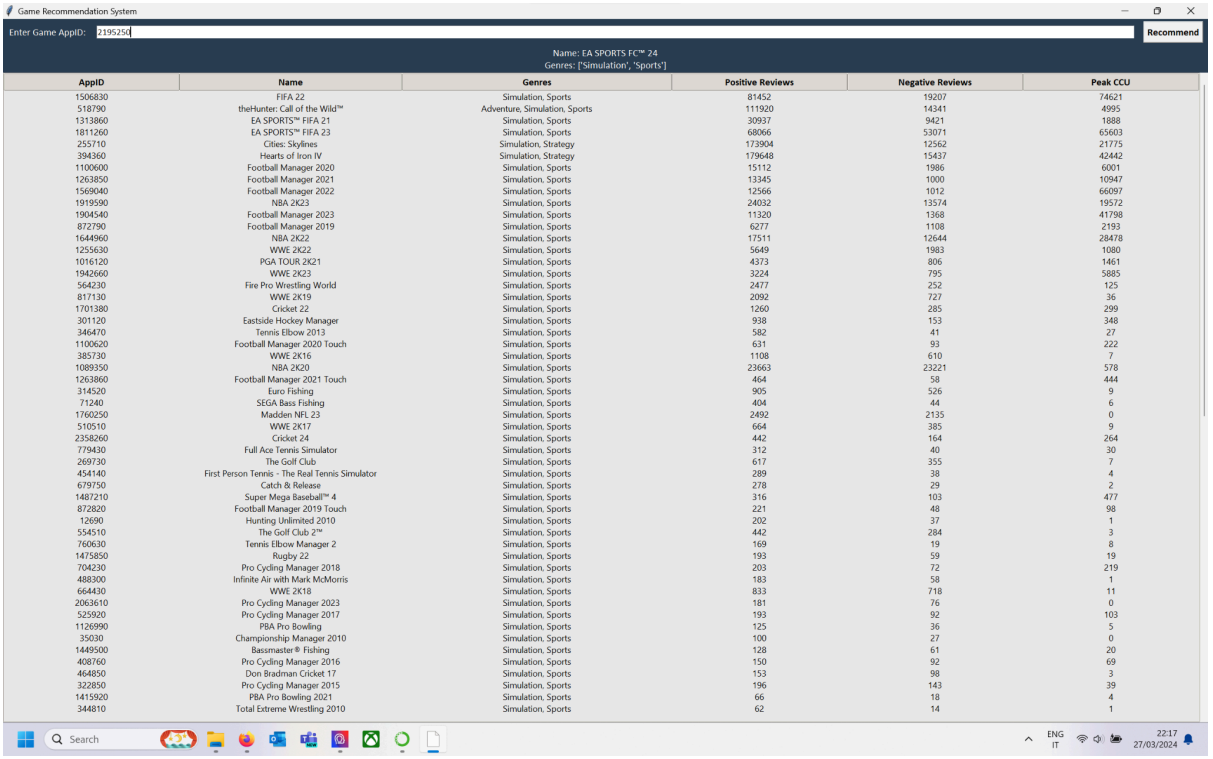
If a game is not present in the db, a popup message will prompt the user to use the scraping code to retrieve the complete game list and go on with the DB updating process.

The results, based on the jaccard similarity, are weighted with the Net Votes attribute.
The weight of the Net Votes was reduced significantly, since the goal of the recommendation system was to give less popular games the possibility to be discovered by an user, for this reason the weight associated with Net Votes is 0.08. Different degrees of this weight were

tested and over 0.1 the Net Votes had too much influence over the set of genres, giving good results still, but did not include any game with few votes.

The computation time is very low, less than a second. By having to compute distances only inside clusters and not on the entire dataset the computation time was reduced drastically, since an execution on the whole dataset was tested and required several seconds.

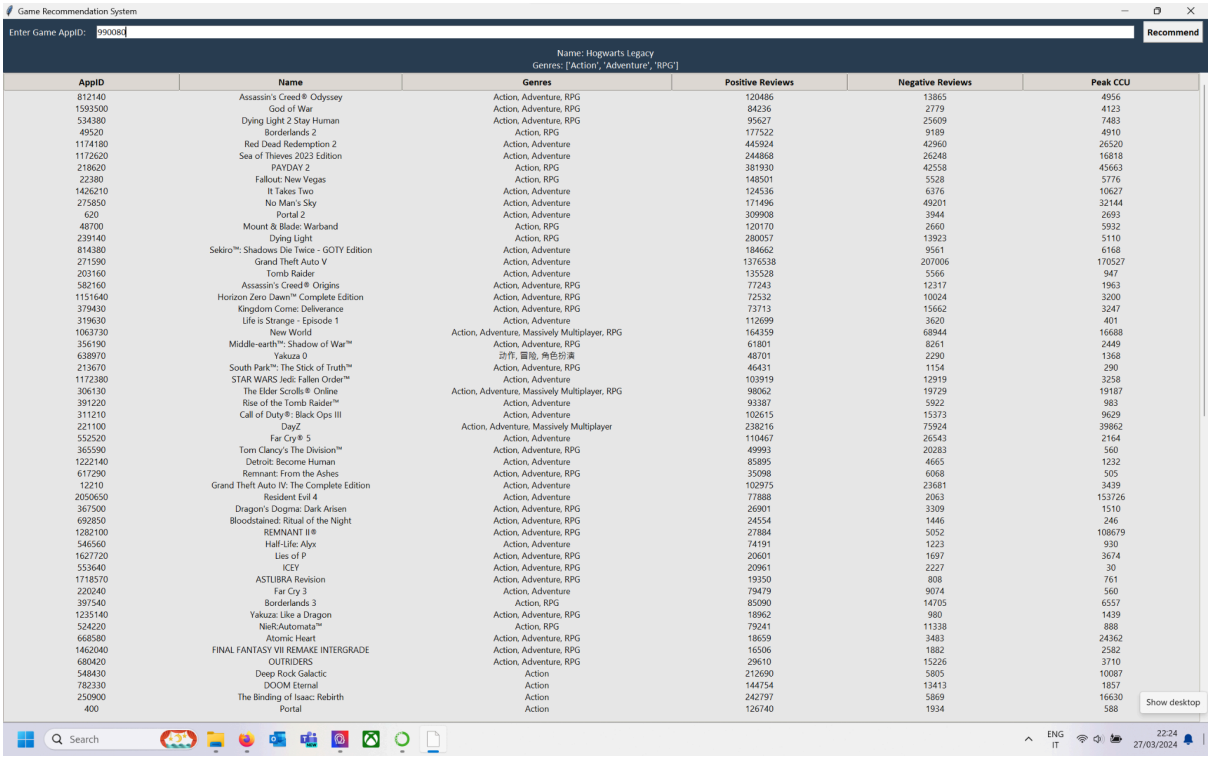
Below a screenshot of the application, the recommendation was run on the soccer game “EA SPORTS FC™ 24” , in the blue part under the textbox the name and tags of the input game are displayed.
In the list,the top 100 recommendations in descending order of Weighted Similarity are displayed.



Since the dataset used for information display is a different version from the dataset used for recommendation computation (which is a one hot encoded version of the genres), some result’s genres may be still in a different language, since this is the state of the DB just before applying the translation model.

From the previous screenshot, the majority of the suggestions are very similar to the input game, but there are two games in the top 10 which, while being simulation games, are not about sports and end up there by having a very high Net Votes value. Before the weighting most of the popular results were ranked high for this reason, but with a value of 0.08 there seems to be a good balance in the results across different games.

For games with more than 2 tags, the results are even better, giving a wide variety of results which are all related to the core concept of the tags assigned. In the screenshot below, results for the game “Hogwarts Legacy” are displayed.



Code folder structure

Name	Date modified	Type	Size
.ipynb_checkpoints	27/03/2024 23:00	File folder	
Data	13/03/2024 18:18	File folder	
Models	27/03/2024 18:24	File folder	
Data_Analysis.ipynb	14/03/2024 09:41	IPYNB File	824 KB
Data_Cleaning.ipynb	13/02/2024 17:54	IPYNB File	921 KB
Data_Model.ipynb	27/03/2024 23:02	IPYNB File	92 KB
Data_Transformation.ipynb	15/03/2024 12:37	IPYNB File	18 KB
Failed_attempts.ipynb	27/03/2024 17:10	IPYNB File	30 KB
key.txt	09/02/2024 21:31	Documento di testo	1 KB
Recommendation_system_GUI.ipynb	27/03/2024 22:42	IPYNB File	160 KB
requirements.txt	27/03/2024 18:26	Documento di testo	1 KB
Steam_api_core.ipynb	14/03/2024 09:57	IPYNB File	27 KB

Inside the Project folder there are 7 Jupyter Notebook files with the .ipynb extension.

Data_Cleaning.ipynb contains the cleaning process
Data_Transformation.ipynb contains the transformation and reduction process

Data_Analysis.ipynb contains the code for the analysis of the initial Kaggle dataset
Data_Model.ipynb contains the code for the clustering and classification process
Recommendation_system.ipynb contains the GUI code and the recommendation algorithm
Steam_api_core.ipynb contains the code used for scraping
Failed_attempts.ipynb This code is not necessary for the application, but contains some of the code about trials discussed in this report. Note that the code in this file is composed of copied and pasted sections from the original structure, with just a Markdown cell identifying what that code was used for.

The **Data** folder includes the final version of the two datasets used for the project
The **Models** folder includes the models generated from the Data_model file and other models such as the glove model for word vectorization or the translation model, which are not included in the final .zip of the report due to the file size

key.txt contains the API key for steam, which will be removed from the zip file
requirements.txt contains the list of required libraries for pip install -r execution

Conclusion and Considerations

The clustering of the dataset presented many challenges, as it is clear from the various silhouette scores and the number of methods employed, the algorithms could not find clear and distinct clusters in the data.

The dimensionality of the dataset was greatly reduced from the initial state of the data and this allowed the final method to achieve some results on the most frequent tags.

The final shapes of the clusters were surprisingly good with respect to the results obtained up to that point. The clusters are fairly balanced and each one contains a specific group of genres.

Another challenge arisen during the development was that it was not possible to compute affinity matrices on the entire dataset, requiring too much memory. An approach I wanted to tackle was to use the spectral clustering on the entire dataset to preserve the global context, even if the implication would have been that no new entry could have been added subsequently in the clusters, requiring a new computation of the affinity matrix and the projections.

If I were to start again and had the possibility to do so, I think a better approach for this recommendation system would have been to use the games actually owned and played by the users, clustering similar users and subsequently recommending games played by similar users. This approach was considered at the beginning of the project, but the API limit on user requests, the fact that there was no way to get a complete user list to iterate on and had to be scraped from the community posts comments and game reviews, and the fact that lots of users have set their profile to private, prevented the development of the project in this

direction. Still, I think that if there was the possibility to gather this information, the clustering alone would have been much more effective.

Regarding the application itself, there are some possible improvement that could be implemented from an usage point of view, the system could retrieve the game list from steam,compare the DB with the list and update it in a separate thread from the one handling the GUI system (Similarly to how workers work in C#). The possibility to search for a game using its name could be another improvement, using a similarity score between strings to return the closest named game to the input one in order to circumvent the exact string search.

Overall, the system is capable of giving relevant suggestions in a very short amount of time, meeting the initial goal of this project.

References

[1]“Steam Games Dataset.” *Kaggle*,

https://www.kaggle.com/datasets/fronkongames/steam-games-dataset. Accessed 30 January 2024.

[2]Alsop, Thomas. “Steam users operating systems used 2023.” *Statista*, 27 October 2023,

https://www.statista.com/statistics/265033/proportion-of-operating-systems-used-on-the-online-gaming-platform-steam/. Accessed 30 January 2024.

[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning.. *GloVe* https://nlp.stanford.edu/projects/glove/ 2014