A Report on the Anime Recommendation system

**Table of Content**

**Topics Pages**

1. Introduction……………………………………………………………………………………………………………………01
2. Related works…………………………………………………………………………………………………………………01
3. Data preparation…………………………………………………………………………………………………………... 03
4. Experimental Setup………………………………………………………………………………………………………...04
5. Summary of experimental result & findings……………………………………………………………….……08
6. Conclusion……………………………………………………………………………………………………………….………09
7. Reference ……………………………………………………………………………………………………………….……….10

**Introduction**

A recommendation system is an algorithm that can be used to suggest the user some relevant content. A recommendation system has become such an important part of consuming content online that we cannot imagine a life without it. A recommendation system can be used in a variety of fields but mostly it is used in the entertainment world and online shopping. Big Companies like Amazon, Alibaba, YouTube, Netflix, Spotify and other use this system to satisfy the needs of their customers. Recommender systems have also been developed to explore research articles and experts, collaborators, and financial services. On YouTube alone, 720,000 hours of content are uploaded every day. Amazon has a catalog of over 12 million products on its websites. With so many options how can one search and decide what to watch or what to buy? This ever-growing list of options if presented raw in front of the user will confuse and frustrate them eventually leading to a bad experience of the service. So, for companies like Netflix, Amazon Prime, Hulu, and many others, the business model and its success revolves around the potency of their recommendations. Netflix even offered a million dollars in 2009 to anyone who could improve its recommendation system by 10%.

Anime recommendation system this system is one of our favorites data mining project ideas. This project data set contains information on user preference data from 73,516 users on 12,294 anime movies. The data set contains rating, genre, type, episodes and different other factors about the anima. We have used two methodologies Simple recommendation system where we rate the top 50 anime based on their ratings from their fan base. The other Methodology is to **Content-based recommenders:** suggest similar items based on a particular item. For instance, we have analyzed the genre of an anime let’s call it “A” then we see how many other movies have the same genre as anime “A” after we have selected and identified those items then we can recommend that to a user that was interested by anime “A”.

**Related Works**

We couldn’t find any local works that has been done. instead, we have found two projects that are similar to our own anime recommendation system.

**YouTube Video Recommendation System: -**There are a whole lot of ways that YouTube can recommend videos but there are just two prominent methods that are followed industry-wide and have proven to work for most of the tasks.

1. **Content-based recommendation system**: - It is a very simple concept, in which the service suggests you on the basis of what you are iterating with. Like if you are watching videos on a particular topic, YouTube would recommend you videos on similar or same topics.

The system internally has a set of different features and a score related to that for all items. Now the system checks from the scores of the content or item you are interacting with and based on that it compares with feature scores of different items and the closest it finds it suggests. These kinds of models also rank feature scores based on the user details like their age, sex or location, and other personal information

1. **Collaborative filtering method: -**This works by recommending items that similar users have liked, therefore it groups similar users together and shares their interests within their group. This is not limited to a particular item but also what kind of item it was and what properties it holds. It can also be bidirectional in nature, for example, it recommends content that similar users have liked. Also, when a user likes some content, it helps in grouping them with similar users, thereby improving everyone’s suggestions. It is based on a simple assumption that if person A has similar taste for most things as person B, his taste will match user B’s in future interactions as well.

**Amazon Recommendation System: -** Along with the developing AI technology, Amazon began working on an algorithm that would be able to analyze items posted by users and determine the shopping preferences of individual customers.  
Amazon's algorithm is a recommendation engine consisting of several important parts responsible for the analysis of various data. This is possible thanks to technology based on [**artificial intelligence and machine learning**](https://recostream.com/blog/machine-learning-ai-deep-learning-diferences).

In order to provide customers with accurate product recommendations, **Amazon's algorithm must analyze huge amounts of data**. In this way, it better understands the behavior of all users and the interests of each viewer. Amazon's [**recommendation algorithm**](https://builtin.com/data-science/recommender-systems) analyzes 3 main types of dependencies and relationships for its operation. **User-Product, Product-product, User- user**

In addition to collecting information about relationships and connections Amazon's recommendation algorithm also uses different types of product and user data **User Behavior data, User Demographics data, Product Attribute Data**

**A method of filtering data by Amazon's recommendation algorithm**

**Content-based filtering on Amazonia:** - Content-based filtering is one of the simplest systems and is constantly used by modern recommendation systems.  
The main idea behind content-based filtering is that if a customer likes a certain product, chances are they will like another product with a similar specification.

**Collaborative filtering:** - Unlike content-based filtering, group filtering **uses the experience of other users to generate recommendations. Interestingly**, Amazon pioneered this approach and published the article

**Bandit and casual interference - Amazon's hybrid algorithms**

**Bandit-based algorithm: -** One of the areas of research on new recommendation algorithms is the so-called Bandit-based algorithms. The "bandits" algorithm is based on machine learning by amplification (RL) and tries to develop sales opportunities for new products using those already profitable. Bandit-based algorithms can also be used to make real-time choices between several recommender models based on how users respond to different product propositions

**Casual interference algorithm: -** Another innovative approach to recommend algorithms that Amazon has worked on is the casual interference algorithm. It mainly focuses on identifying the factors that prompted individual customers to pay attention to specific products. By developing an algorithm that combines casual interference with existing recommendation algorithms, Amazon researchers were able to generate improved recommendations by taking into account various confounders. It is also worth mentioning here that hybrid systems are becoming increasingly popular. Some of these newer approaches are not mutually exclusive and can be combined with each other

**Data Preparation**

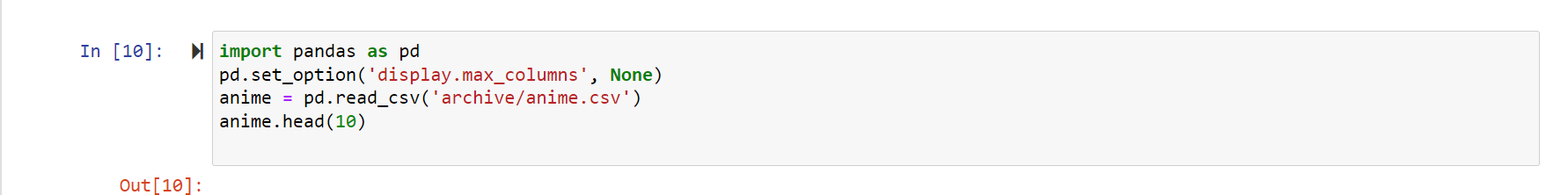
The Anima recommendation data set contains information on user preference data from 73,516 users on 12,294 anime movies. We have gotten the dataset from favtutor.com. the project has been one of the most interesting data mining projects in 2022. We were pretty excited to be a part of this project and we enjoyed it. In order to prepare the dataset, we have used python. We have taken 2 simple steps to prepare the data.

* **Data exploring**
* **Data cleaning**

**Data Exploring: -** In order for us to clean the dataset and design an algorithm we need to be much more familiar with the dataset that we are working with.

We imported the dataset on the jupyter notebook using python 3

Before we start to read the data that we are about to work on we first import panda. Pandas is a powerful, fast, flexible**open-source library** used for data analysis and manipulations of data frames/datasets. Pandas can be used to read and write data in a dataset of different formats like CSV. Now we can read the data first and we display the first 10 rows and show all rows



**Commands like**

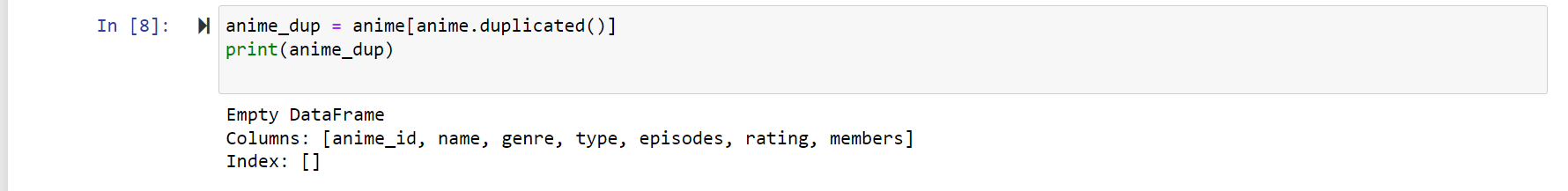
* **anime.shape // Shows as the total number of rows and columens the dataset contains**
* **anime.info () // Shows the amount of the non-null values and the datatype of each column**
* **anime.describe() // Shows the min, max, mean, std and other values of each column**

|  |
| --- |
| The Columns of the first data sets and what they represent |
| **anime\_id:** -id number for each anime title **name: -** title of the motion picture **genre: -** category **type: -** describes the anime into tv, movies, OVA, and 3 other categories **episodes: -** total number of episodes **rating: -** 1–10, lowest to highest **members: -** number of community members that are in this anime group |

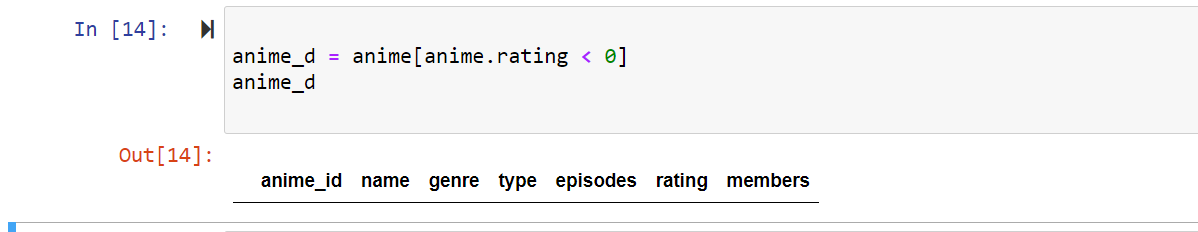
**This step is very important, as it helped us to explore the dataset and understand it better so that we can clearly move forward and design a good algorithm.**

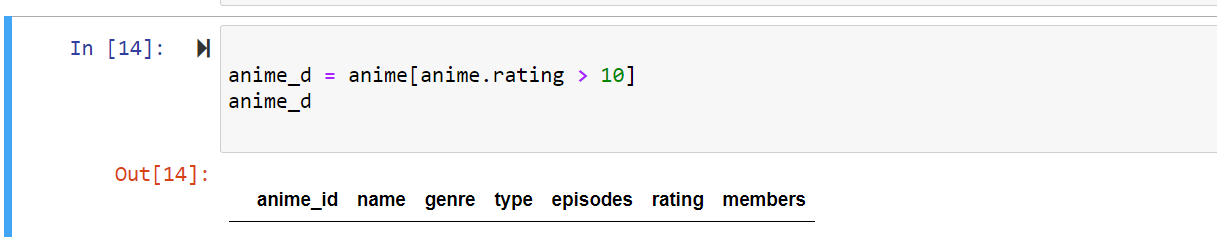
**Data cleaning: -** In order for us to get a high performing algorithm and so have an accurate recommendation system we need to clean the dataset of any outliers, noisy, duplications …

* anime.dropna(inplace=True) // Check if there are any null values and then drop them
* Check if there are any duplicates luckily, we don’t have any duplicates



* **No outliers and noisy data in the dataset**





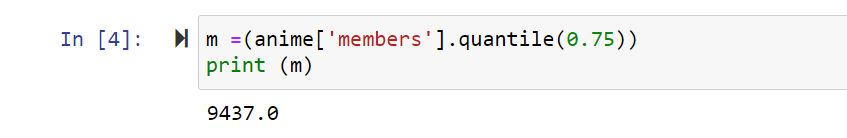
**Experimental setup**

**Simple Recommendation System:** - The basic idea behind this system is that anime that are more popular and critically acclaimed will have a higher probability of being liked by the average audience. simple recommenders are basic systems that recommend the top items based on a certain metric or score.

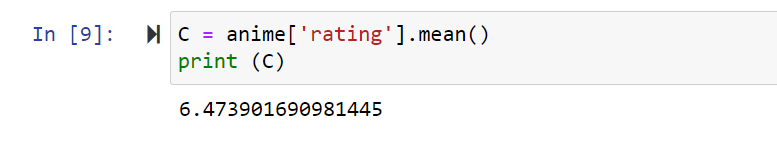
It is a pretty simple concept we just have to compare the ratings given to each of the anime but not quite. For one, it does not take into consideration the popularity of a movie. For instance, a movie with a rating of 9 from 10 voters will be considered 'better' than a movie with a rating of 8.9 from 10,000 voters. So we use what is known as Weighted rating Formula as a metric/score

* v is the number of votes for the anime
* m is the minimum votes required to be listed in the chart
* R is the average rating of the anime
* C is the mean vote across the whole report

There is no right value for m. We can consider it as a preliminary negative filter that will simply remove the movies which have a number of votes less than a certain threshold m. 75TH Percentile considering the top 25% of the movies of the number of members garnered.

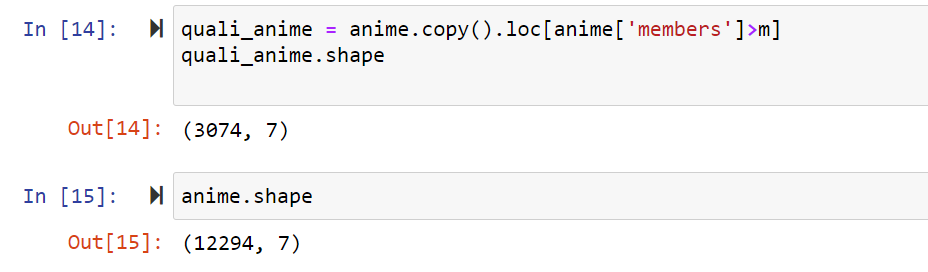


Then we need to calculate the value of C the mean rating across all anime using the pandas. mean() function:



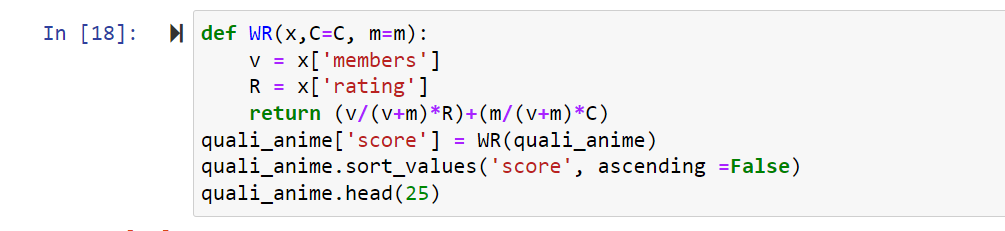
* Indicates that the average rating is 6.4

To insure that the new Data frame created is independent of your original metadata Data frame we use .copy() method. Any changes won’t affect the original data frame.



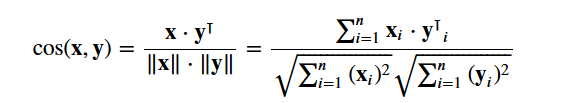
* We can see that 25% are qualified

We have already computed m and C. We have to get (v) and (R) from the data frame.



* First, we computed the weighted rating of each anime
* Then we Define a new feature ‘score’ and calculate its value with ‘weighted rating ()’
* Sorting the amine based on score value
* Lastly print the top 25 anime.

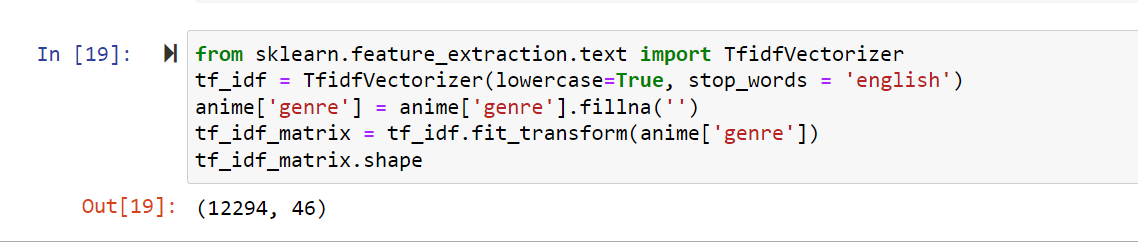
**Content-based recommenders: -** The general idea behind these recommender systems is that if a person likes a particular item, he or she will also like an item that is similar to it. And to recommend that, it will make use of the user's past item metadata. It suggests similar items based on a particular item. We use cosine similarity scores for all anime based on their genre descriptions and recommend anime based on that similarity score threshold.



**This are the necessary step we need to take in order to implement it.**

The continent we chose is the ‘gene’ of the anime. To do this, you need to compute the word vectors of each overview or document, as it will be called from now on. As the name suggests, word vectors are vectorized representation of words in a document. The vectors carry a semantic meaning with it.

We compute Term Frequency-Inverse Document Frequency (TF-IDF) vectors for each document. This will give you a matrix where each column represents a word in the overview vocabulary (all the words that appear in at least one document), and each column represents a movie, as before.



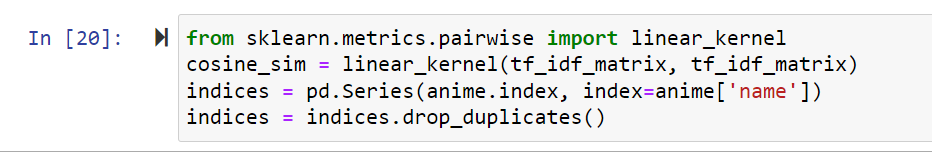
Here is an explanation to what each line do

* Line 1: - Import the Tfidf module using scikit-learn
* Line 2: - Remove stop words like “the”, “an”, etc. since they do not give any useful information about the topic
* Line 3: - Replace not-a-number (NaN) values with a empty string
* Line 4: - Finally, construct the TF-IDF matrix on the data by fitting and transforming the data
* Line 5: - Output the Shape of the tfidf\_matrix

We can see that there are 46 different words from 12,294 anime.

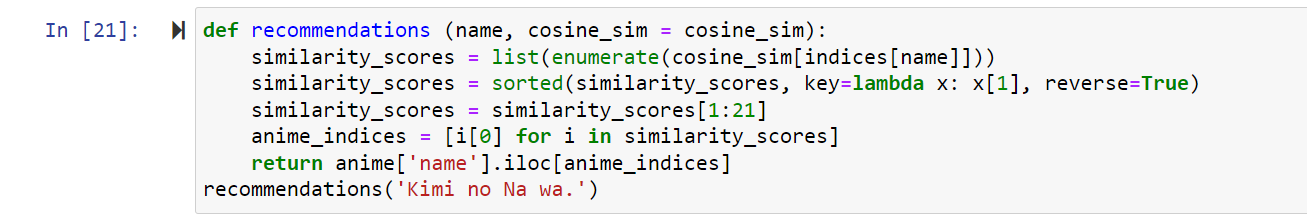
The use of the cosine similarity to calculate a numeric quantity that denotes the similarity between two anime. We used the cosine similarity score since it is independent of magnitude and is relatively easy and fast to calculate.

Since you have used the TF-IDF vectorizer, calculating the dot product between each vector will directly give the cosine similarity score. Therefore, we will use sklearn's linear\_kernel () instead of cosine\_similarities () since it is faster.



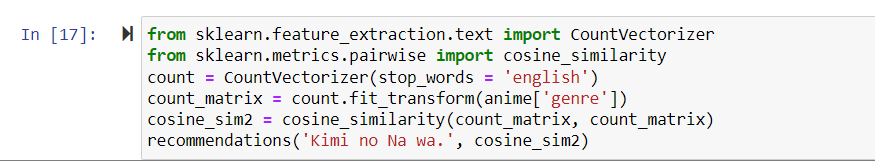
* Line 1: - import linear kernel
* Line 2: - Compute the cosine similarity matrix
* Line 3: - Construct a reverse map of indices and movie titles
* Line 4: - Drop if there are any duplicates

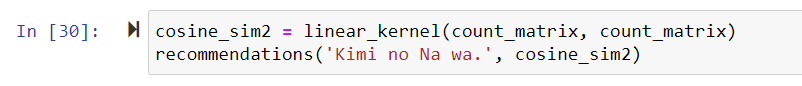
The last steps are simple we build the function that take in a movie title as input and outputs most similar anime. And here are the steps



* Line 2: - Get the index of the movie that matches the title
* Line 3: - Pairwise similarity score of all anime with the given anime
* Line 4: - Sort the anime based on similarity scores
* Line 5: - Get the score of 10 most similar movies
* Line 6: - Get the anime indices
* Line 7: - Return the top 10 similarity movies
* Line 8: - Get recommendation

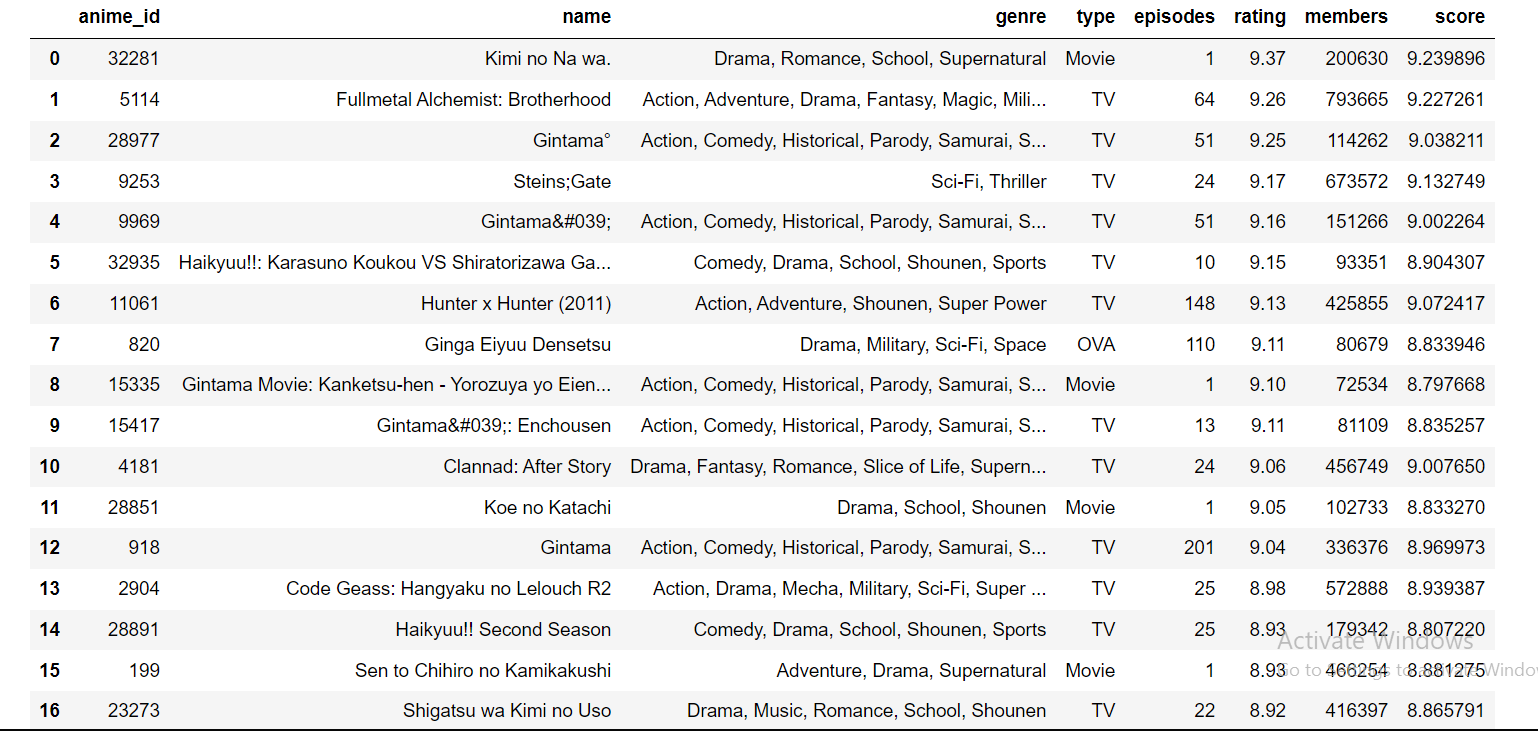
**Next, we are going to look at another model, CountVectorizer() and we are going to compare the result between cosine\_similarity and linear\_kernel.**

****

* Line 1: - import CountVectorizer
* Line 2: - compute the cosine similarity matrix based on the count\_matrix
* Line 3: - Remove stop words like “the”, “an”, etc. since they do not give any useful information about the topic
* Line 4: - Replace not-a-number (NaN) values with a empty string
* Line 5: - Compute the cosine\_similarity
* Line 6: - Get recommendation
* Line 5: - Compute the linear\_kernel

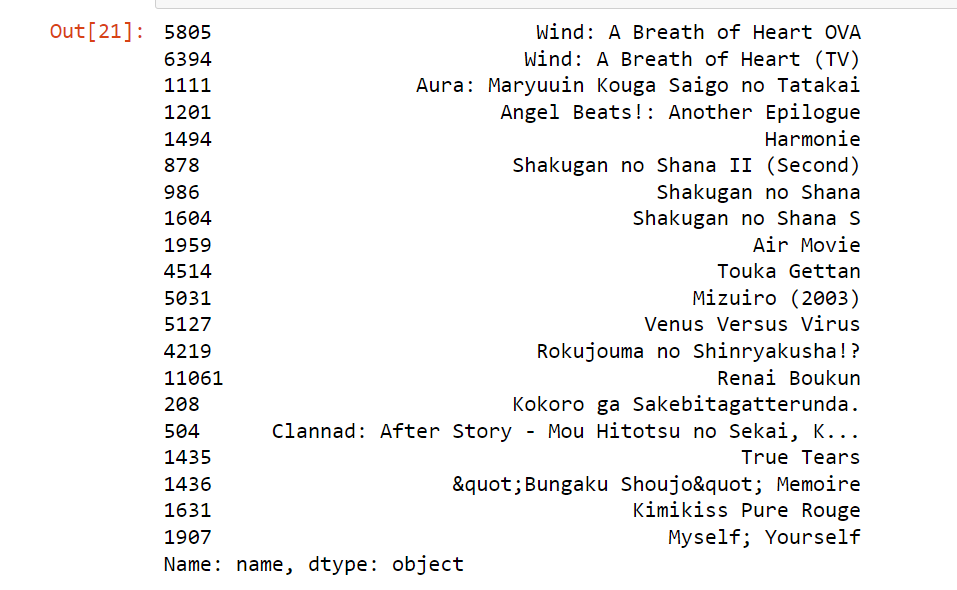
**Summary of experimental result & findings**

**The simple recommendation system we have tried to compute the top 25 anime based on the data frame. Used a weighted rating formula.**

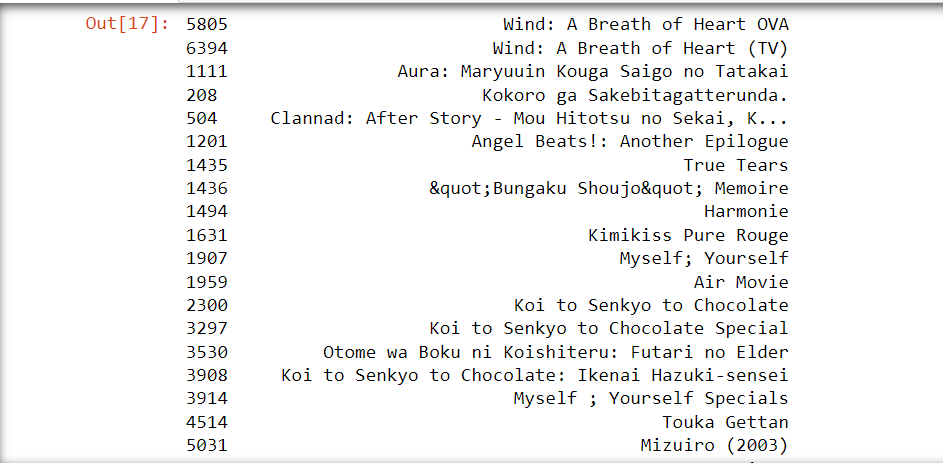


**In the content recommendation system, we have done two models using TF-IDF calculation and CountVectorizer which uses the linear\_kernel and cosine\_similarity to compute. We need to compare between the two methods in order to determine the better method.**

TF-IDF Calculation

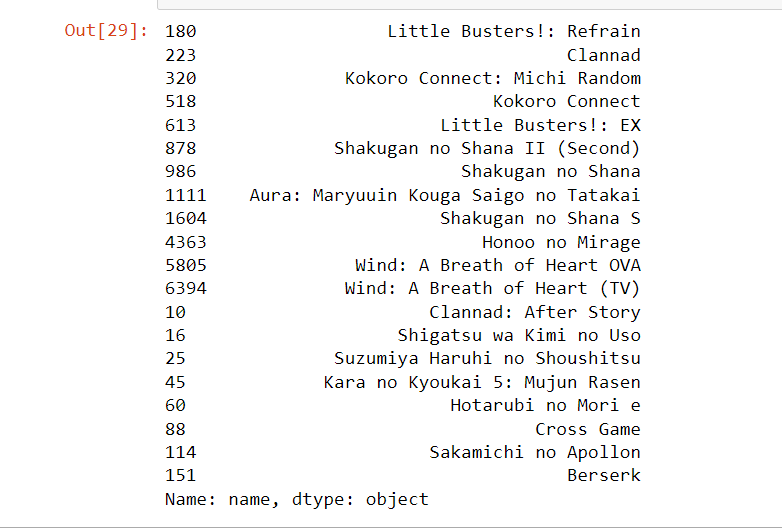
****

CountVectorizer() **cosine\_similarity**



The above two are pretty simmilar they are even the same compare two the thired one which is a bit different.

CountVectorizer() **linear\_kernel**



While comparing both linear\_kernel and cosine\_similarity produced the same result. However, linear\_kernel took a smaller amount of time to execute. When you're working with a very large amount of data and your vectors are in the tf-idf representation, it is good practice to default to linear\_kernel to improve performance.

**Conclusion**

In conclusion recommendation system is very important to refer to relevant content to users. It is also used in many fields entertainment, shopping, and researches too. These types of companies and organizations have a lot of data and those data increase in a very enormous amount by day. This growing data creates a large number of options for the users making it hard for them to choose and might lead to a bad experience. Anime recommendation system used for recommending the audience to best or recommended the anime movie from a list of thousands of anime movies using algorithms and methods. We saw different ways for a recommendation system. What we personally (during our project) did is, take the data clean it. Important thing, understand the data first. We used a tool for accomplishing this and then to clean the data of any null values and such. Then we moved on to a simple recommendation algorithm. This just calculate weights and assign it to the anime movies. Then move on to content-based recommendation, which uses cosine similarity scores for all anime based on their genre descriptions and recommend anime based on that similarity score threshold.

**Reference**

* <https://towardsdatascience.com/recommendation-systems-explained-a42fc60591ed>
* <https://recostream.com/blog/amazon-recommendation-system>
* <https://data-flair.training/blogs/youtube-video-recommendation-system-ml/>