UDACITY

Capstone Project:

An anime Recommender System Based on Content Using Tree Models

Udacity

Machine Learning Engineering nanodegree

Indy Navarro Vidal

28-02-2021

Indice

Introduction

In machine learning, there is a vast potential to build new products based on the robust algorithms that learn from data. One area in which machine learning has become strong is in the recommender system.

Recommender systems are a subclass of information filtering system that seeks to predict the rating or preference a user would give to an ítem. They are strongly related to e-commerce or streaming. We can see RS on websites like Amazon, Netflix, or Youtube.

The recommendation systems had a shallow profile two decades ago. It was not a prominent area of machine learning. Nevertheless, this changed in 2009, when Netflix launched a million-dollar prize competition, asking to build a recommendation system that could make a substantial 10% improvement over its current system. Currently, the company has one of the most solid RS in the market. This has impacted many e-commerce, technology companies, among others, to build their recommendation systems.

Additionally, there is an exciting industry exploring a recommender system, which is the anime industry. Anime refers to hand-drawn and computer animation originating from Japan. This industry has matured over time and has impacted not only in the east but also in the west, considering different sorts of art forms like drawing, films, music, and the list continues getting a considerable number of fans worldwide.

To sum up, the main idea is to help anime fans who want to continue consuming more of this art. Considering data and classic machine learning models, we will build a RS that can help users find new animes they might enjoy.

With two datasets obtained from Kaggle, we will proceed to build a single dataset that links the characteristics of the anime (based on content) with the respective score that the user gives. Hence, the objective is to try to make an accurate forecast of the valuation to give the user to the anime. Furthermore, thus to be able to recommend things that the user would enjoy.

The problem is considered to be a regression problem. Therefore, the metrics we will use for this case are the MSE, RMSE, and above all MAE, as the final comparison between models. MAE is a much more intuitive metric to explain its results to non-specialists in the area.

Finally, linear regression is used as a benchmark model, which will be described in greater detail later in the report, the use of tree-based models is proposed to be able to improve the MAE that comes from the regression, and that can lead to a better result

Exploratory data analisys

In this project, we have two datasets that are taken from Kaggle. In case you want to review the source, The datasets can obtain it from the following link: https://www.kaggle.com/CooperUnion/anime-recommendations-database
This data set contains information on user preference data from 73,516 users on 12,294 anime from myanimelist.net. Each user can add anime to their completed list and give it a rating, and this data set is a compilation of those ratings.

Anime.csv

· anime_id - myanimelist.net's unique id identifying an anime.

· name - full name of anime.

· genre - comma separated list of genres for this anime.

· type - movie, TV, OVA, etc.

· episodes - how many episodes in this show. (1 if movie).

· rating - average rating out of 10 for this anime.

· members - number of community members that are in this anime's

"group".

In general, we see that this dataset's composition has few missing values in gender and rating. For the first case, we may be able to review the anime and fill it out. The second case is not so simple and may include imputation for medians.

Rating.csv
· user_id - non-identifiable randomly generated user id.

· anime_id - the anime that this user has rated.

· rating - rating out of 10 this user has assigned (-1 if the user watched it but did not give a rating).

We can observe some of the most relevant statistics of each dataset below. First, let's look at the statistics of the anime dataset

	anime_id	rating	members
count	12294.000000	12064.000000	1.229400e+04
mean	14058.221653	6.473902	1.807134e+04
std	11455.294701	1.026746	5.482068e+04
min	1.000000	1.670000	5.000000e+00
25%	3484.250000	5.880000	2.250000e+02
50%	10260.500000	6.570000	1.550000e+03
75%	24794.500000	7.180000	9.437000e+03
max	34527.000000	10.000000	1.013917e+06

fig 1-. Description anime dataset

In this table, we can see our three quantitative variables. However, only two are relevant (since the id does not have a quantitative value per se). As shown in the rating, its mean is around 6,47 with a 1,02 standard deviation. Its minimum is 1,67 and maximum 10. In the case of members, its average is a little more than 18.071. The standard deviation is 54.820, and its minimum is 5 people and a maximum of 1.013.917. This last variable does not have a behavior similar to a normal one. The difference between the mean and the median is huge. The presence of outliers may be observed in the graph.

If we do the same exercise in rating dataset, we can see this table:

	user_id	anime_id	rating
count	7.813737e+06	7.813737e+06	7.813737e+06
mean	3.672796e+04	8.909072e+03	6.144030e+00
std	2.099795e+04	8.883950e+03	3.727800e+00
min	1.000000e+00	1.000000e+00	-1.000000e+00
25%	1.897400e+04	1.240000e+03	6.000000e+00
50%	3.679100e+04	6.213000e+03	7.000000e+00
75%	5.475700e+04	1.409300e+04	9.000000e+00
max	7.351600e+04	3.451900e+04	1.000000e+01

fig 2-. Description rating dataset

To start the exploratory data analysis, we will begin with the anime.csv dataset. In this, we will observe a histogram of the average rating that is considered to all anime. The image below demonstrates a skewed distribution, concentrating much of its values in the buoyant sector of a theoretical normal distribution. The median is 6.55, and the average is 6.47, which is not a big

difference and makes a lot of sense knowing that the rankings cannot present outliers since they are limited (in this case, from 1 to 10).

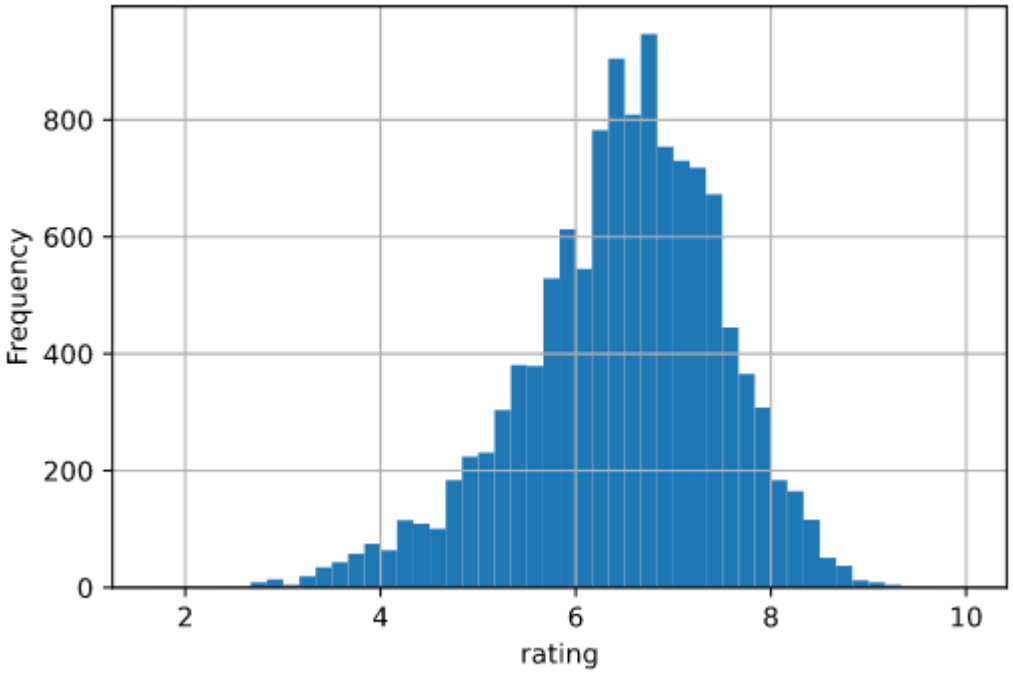

Fig 3-. Histogram rating in anime dataset

Similarly to the previous one, we consider the variable "members" to see if there is any known distribution. Still, in this case, we can see the long tail effect, where it is shown that most animes have a smaller amount of followers, and very few manage to reach the other levels, that is, to have more members.

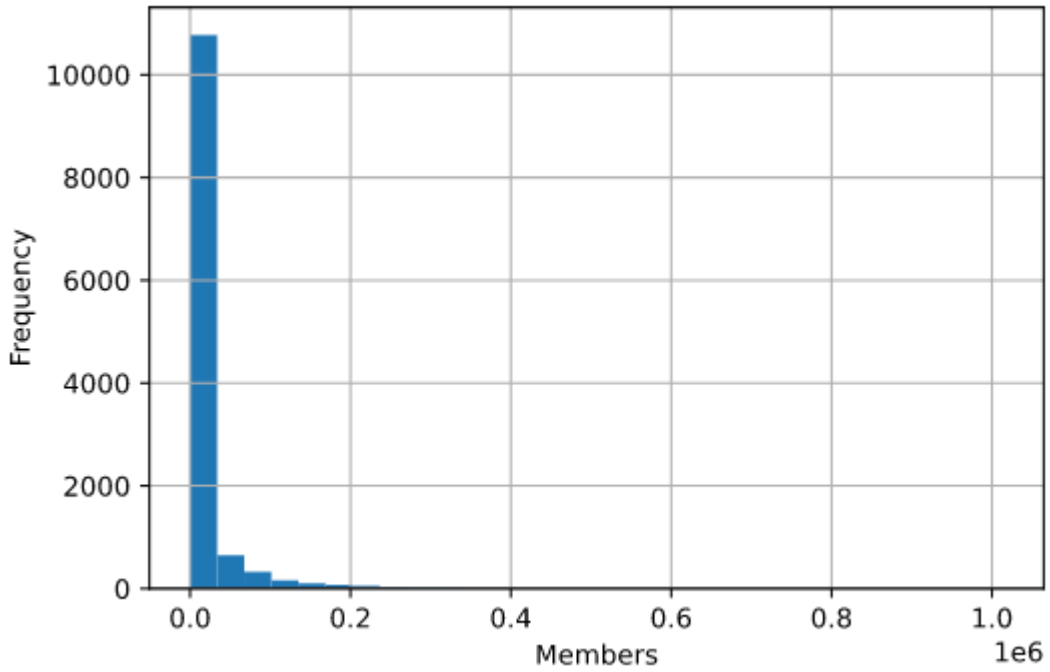

Fig 4-. Histogram members in anime dataset

Then we decided to study the frequency for each number of episodes. Anime's episodes have a considerable interval of possibilities, some reaching more than a thousand episodes. However, in this case, we have a large concentration of anime programs with a single chapter. Is that possible? If we consider that there are several types of programs: TV, movies, OVAs, etc. Many of them will have only one episode.

On the other hand, TV shows are renewed every season (3 months). Many of them reach 12, 13, 24, 25, or 26 episodes, so the results are more than acceptable. Note also that there are a significant number of series that are "unknown" to be considered.

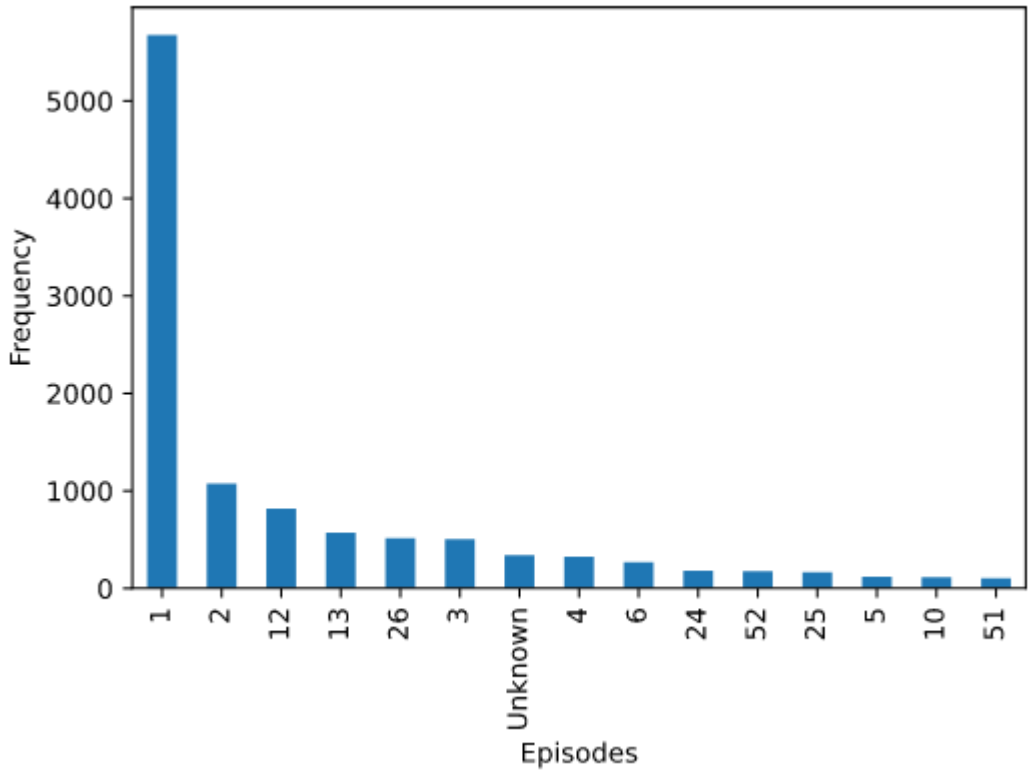

Fig 5-. Histogram episodes in anime dataset

Now we turn to consider the types of anime shows that appear on the base. In general, we will see that the majority are TV, followed by OVAs and Movies. Considering that the amount of TV is not that great, it would rightly explain that most animes have only one episode.

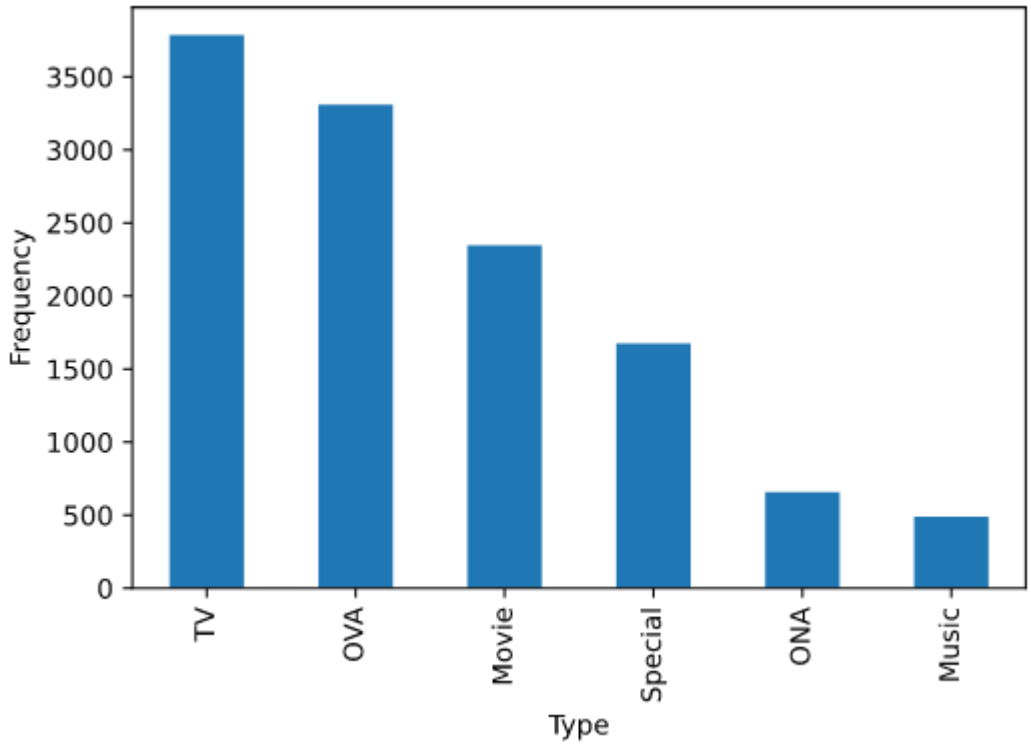

Fig 6-. Histogram type in anime dataset

Then we seek to observe the top 10 of the series with the highest rating, this variable that we can observe can be misleading since when judging these ten anime, we can notice that there are some of them which are not popular (such as the first three), but also we can see in the list animes like Kimi no Na wa, Gintama or Steins Gate, which are more or less popular and which could be in a ranking of this type. It's believed that a possible defect of this feature is that it is an average of the people who have voted. Maybe the people who participated were a minority, causing an overvaluation of anime.

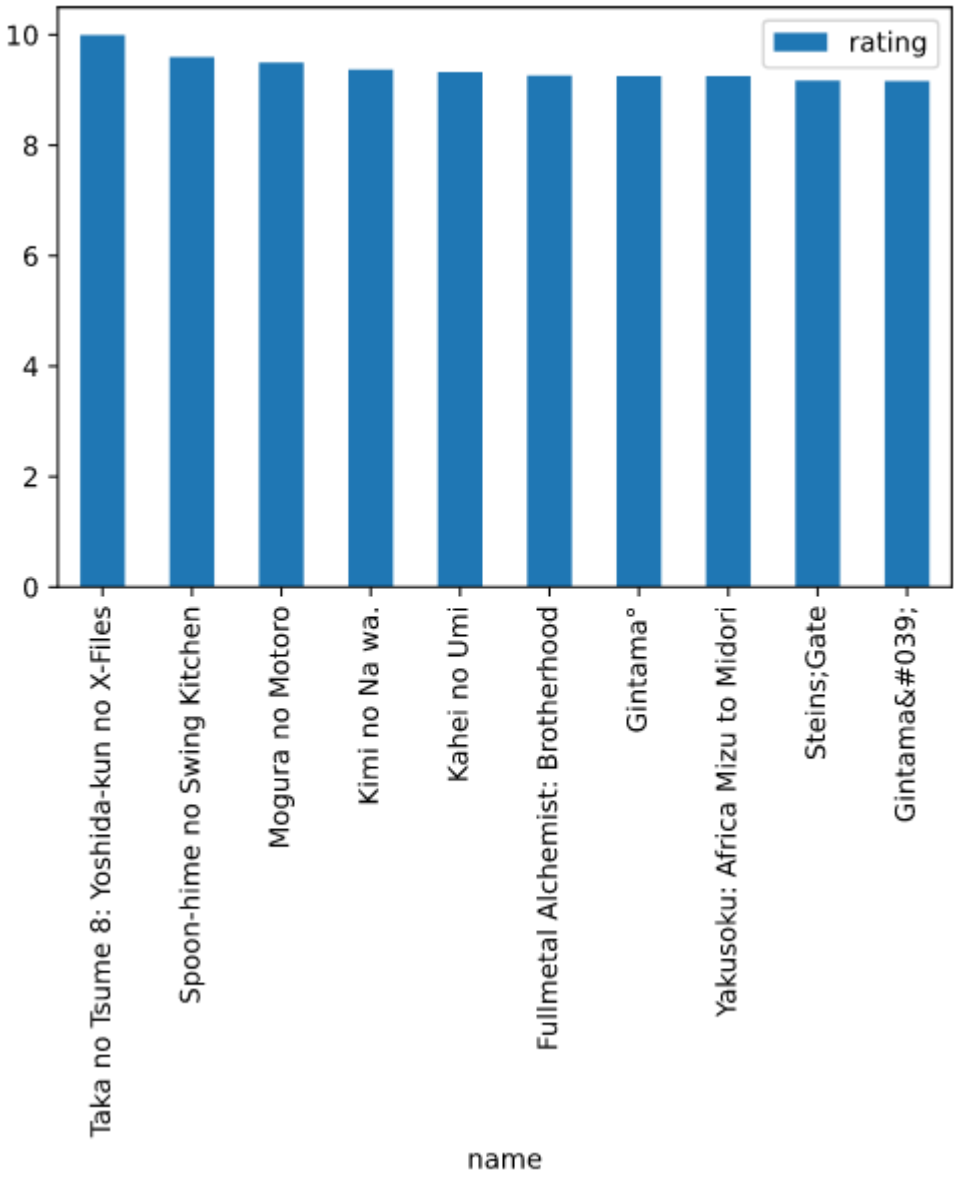

Fig 7-. Top 10 anime based on rating

The following graphic presents another top 10, but this one is based on community. This variable makes more sense to check because all the series in this ranking are top-rated in their launch season and later ones. So it can mean that it is a more significant variable to think about recommendations.

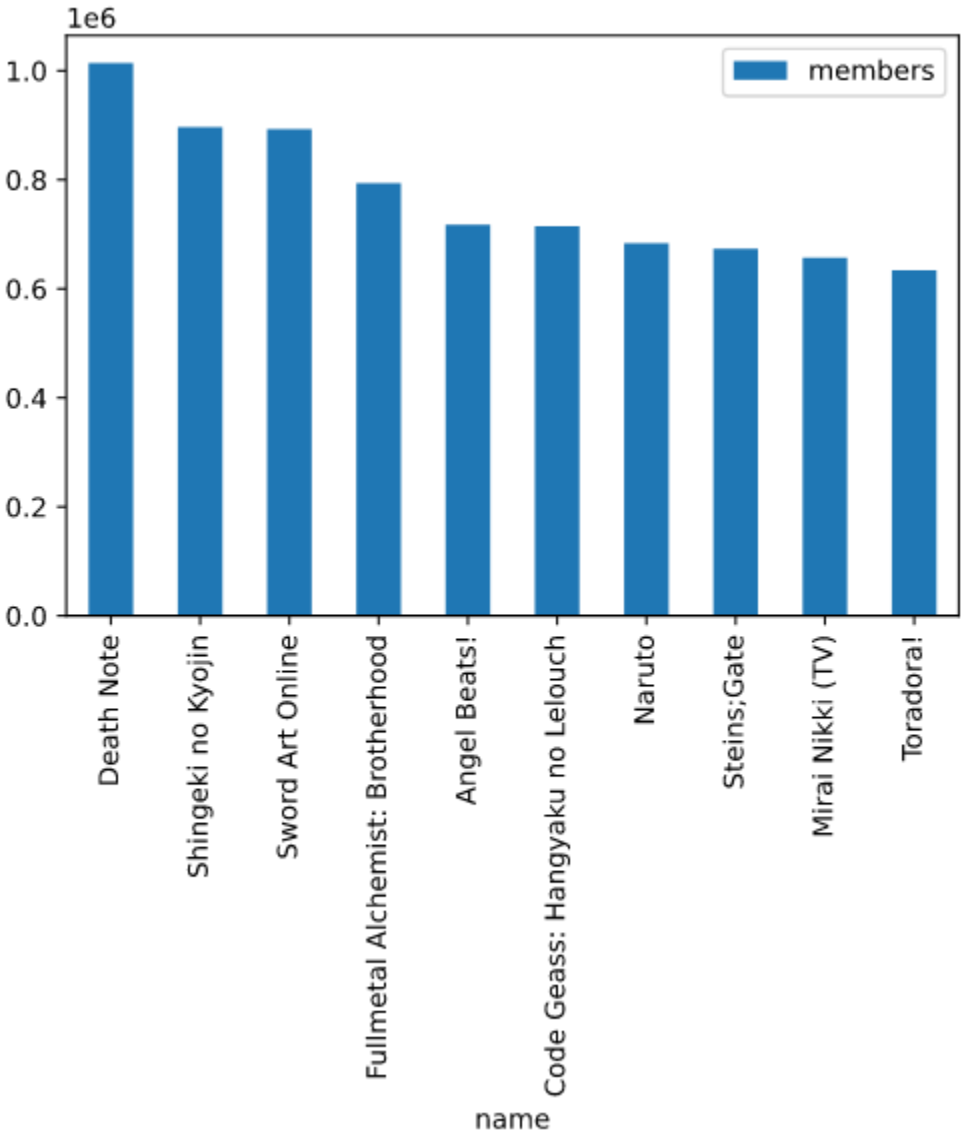

Fig 8-. Top 10 anime based on members

The following graph shows a relationship between the number of members and the rating that we can observe in each of them. This shows an intuitive relationship, but it is still impressive. In general, with few members, an anime can have a good or bad ranking, even observing the best ratings came from small communities (these series may be niche and only really draw attention to a smaller group of people). However, on the other hand On the other hand, as the number of members increases, the rating tends to be positive. Of course, if the community is large, one would expect the anime to be at least decent, right?

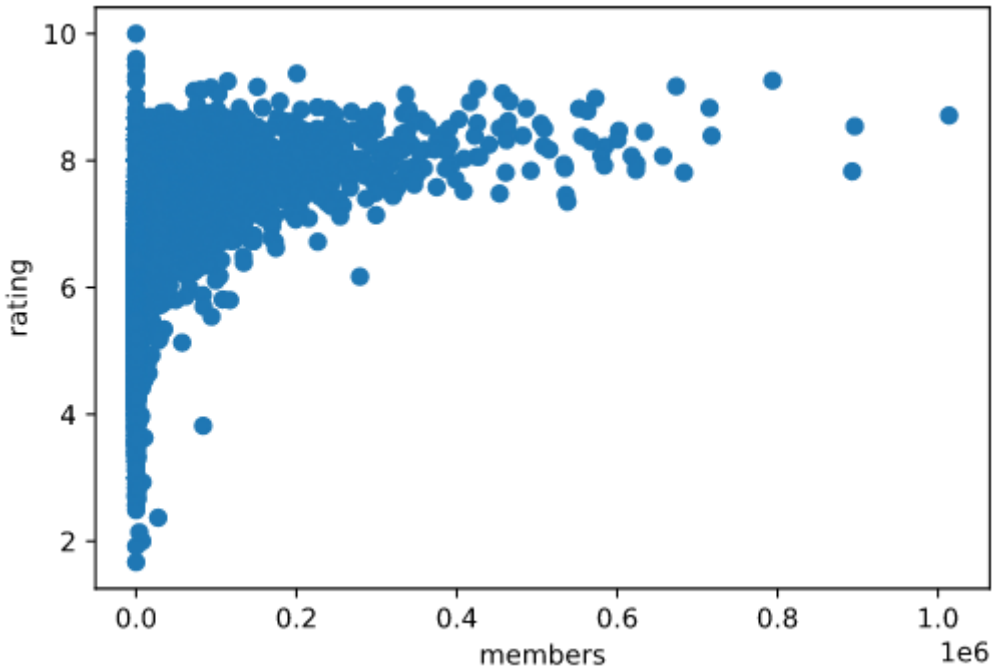

Fig 9-. Scatter plot members-rating

Similarly, we can review the relationship between the episodes and the rating obtained from each anime. We can also see that with few episodes can get the best evaluations and the worst ones (and all the options of the interval). However, the more episodes the series has, the more it can score well. Series like Naruto, One Piece, or Dragon Ball may have followers who will significantly value the series. Of course, no one would waste their time watching such a long series if they didn't like it.

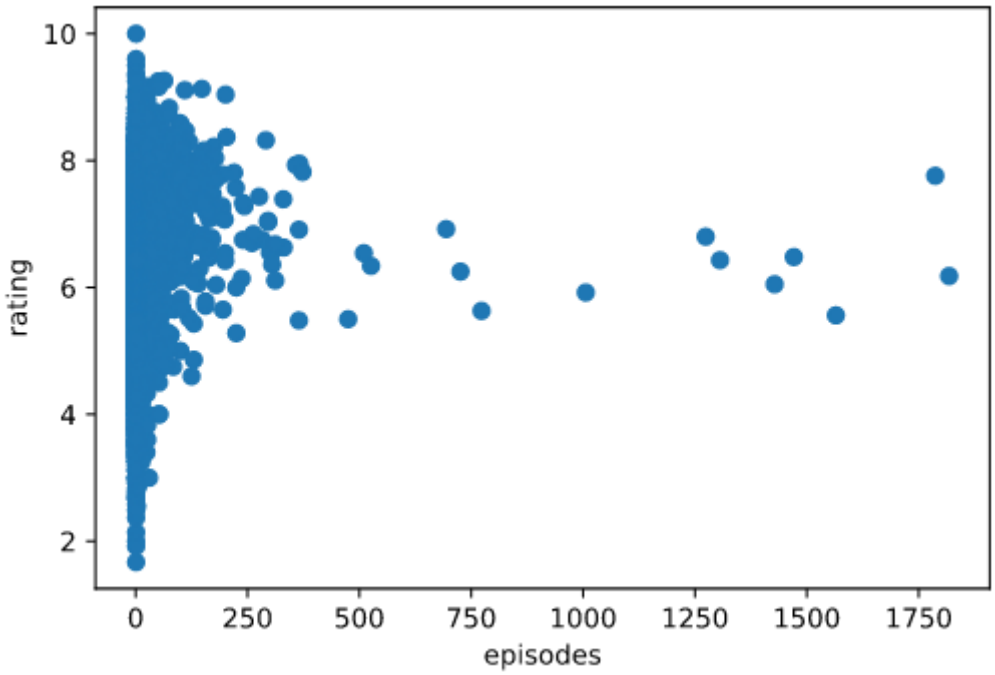

Fig 10 -. Scatter plot episodes-rating

We will now analyze the rating.csv dataset, which will be much simpler since only three variables. The first graph below shows us the rating we have for each of the variables. A specific imbalance can be observed in the labels. The majority prefer to vote 10 points and progressively with the decrease in this evaluation. It also decreases the frequencies of this. This can be due to several factors, such as the motivation to participate or the real motivation to leave negative feedback. There may be a tendency to participate more when the evaluation is positive. On the other hand, we also have many people who did not vote for the animes. The truth is that there are some explanations such as the impossibility of voting since the person did not have an account in My Anime List, a group of people who are not interested in leave feedback, or series that were not really positive for the user to the point that they declined to leave a rating or even animes that they stopped following and that they only registered and did not evaluate (since they have not finished it yet). For this project's scope and to simplify it, we decided not to make an imputation of this data and only delete it from the dataset.

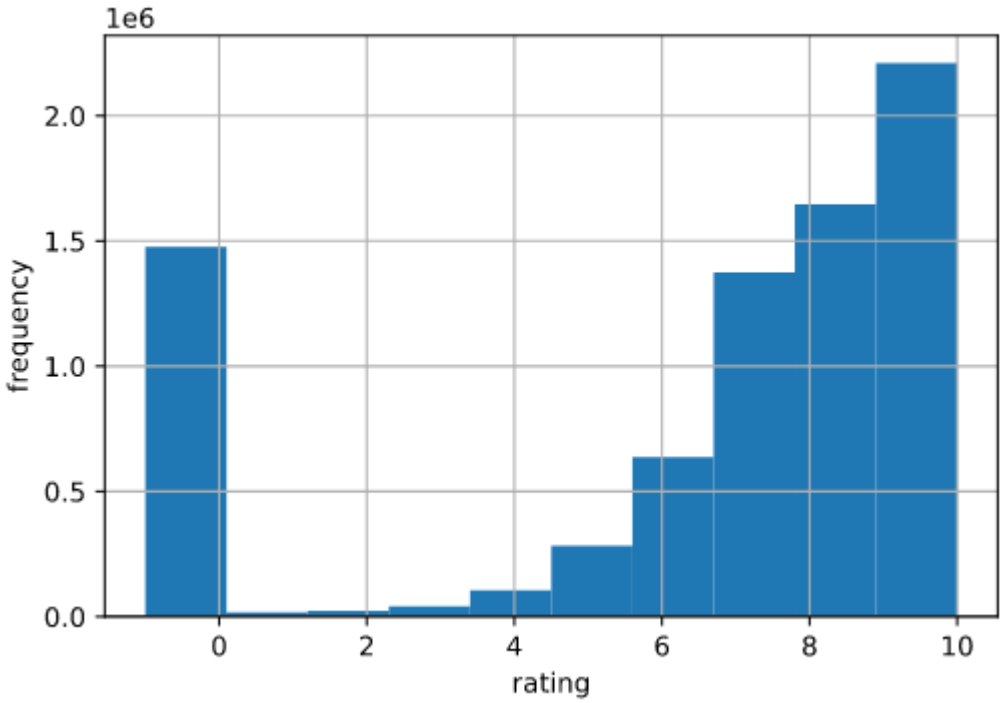

Fig 11 -. Histogram plot on rating in rating dataset

Then, we can observe the participation of the users in the dataset. Here we can observe an unusual case. A particular user has a much higher number of records than the rest of the users and then drops considerably by the second user with more records. It is a somewhat unusual long-tail since a user has such a quantity that it is considerably different from the rest.

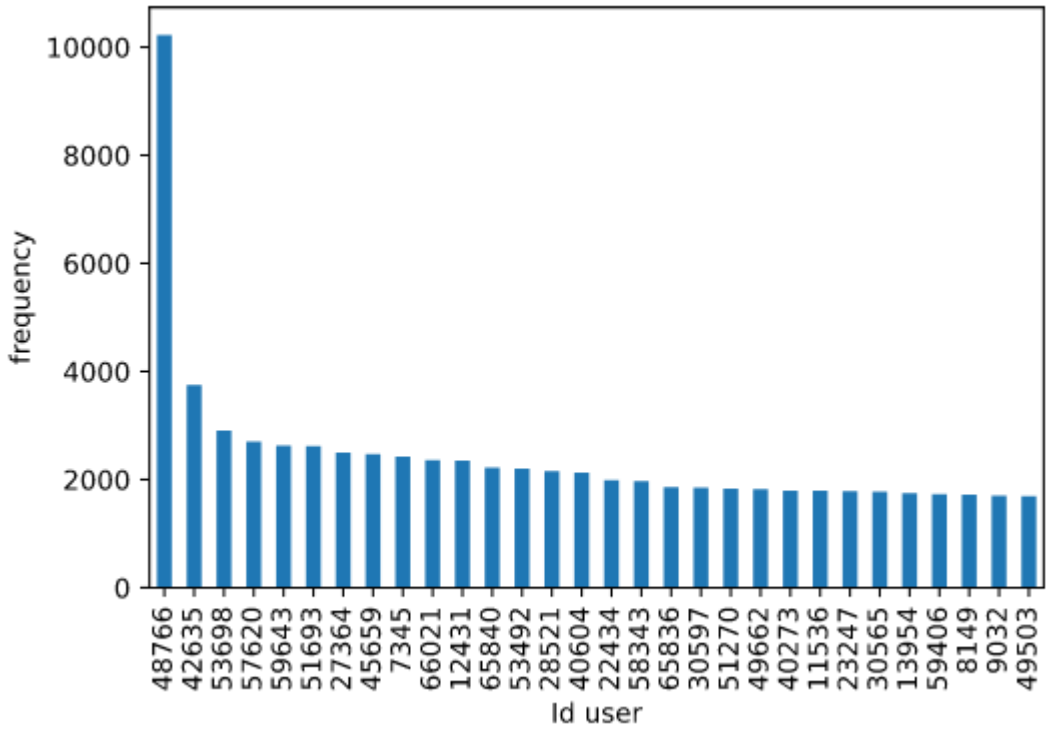

Fig 12 -. Histogram plot on Id user in rating dataset (top 30)

For the graph of the series, we can see that the frequency ordered from highest to lowest shows us a better balance, where the series does not have a high demand concerning the others.

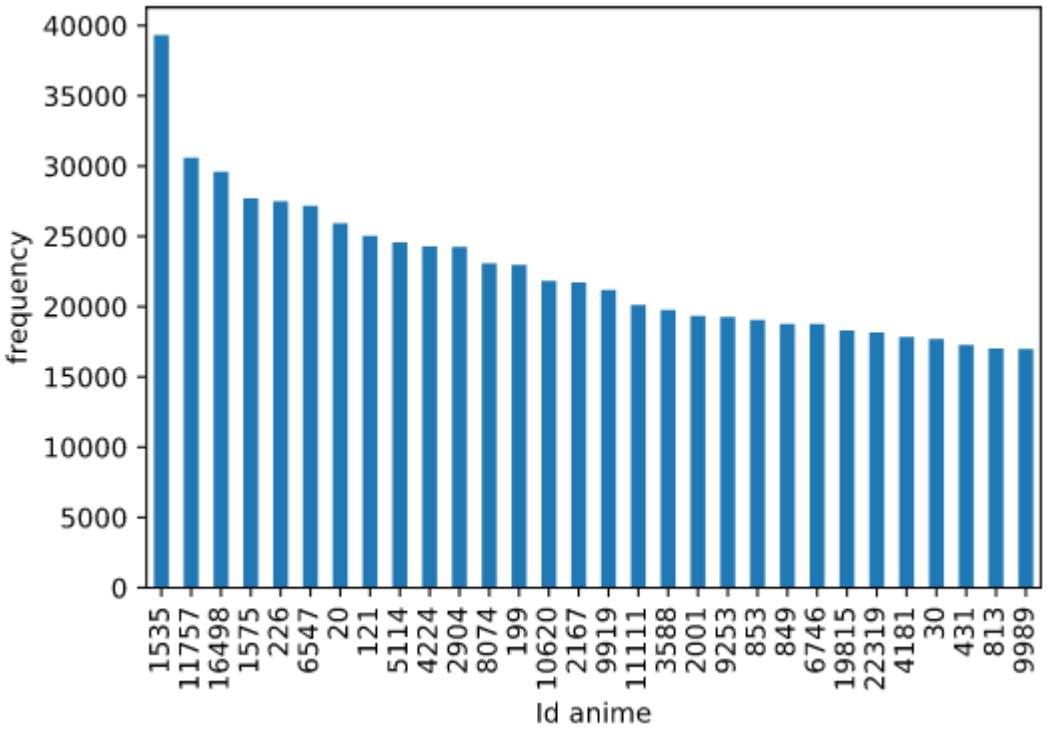

Fig 13 -. Histogram plot on Id anime in rating dataset (top 30)

Algorithms

This section will present the model used as a benchmark and the proposed models to evaluate possible improvements to the score, these are models based on trees and ensembles.

Linear Regression

The proposed benchmark model was linear regression, and it is one of the simplest models in machine learning; the choice of a model of this type is because, in addition to its simplicity, it can be understood by business owners. On the other hand, it provides a lower bound on the performance of a model and is one of the fastest models and where it does not require the configuration of hyperparameters.

Given a dataset, a linear regression model assumes that the relationship the the relationship between the dependent variable *y* and the *p*-vector of regressors **x** is linear. Thus the model takes the form:

$$f_{w,b}(x) = wx + b$$

Where we have a collection of labeled examples $\{(x_i, y_i)\}_{i=1}^{N}$ and N is the size pf the collection, x_i is the D-dimensional feature vector of example i= 1,…,N, y_i is a real-valued target and every feature $x_i^{(j)}, j = 1, \dots, D$ is also a real number.

To get this latter requirement satisfied, the optimization procedure which we use to find the optimal values for w and b tries to minimize the following loss function:

$$\frac{1}{N}\sum_{i=1..N}(f_{w,b}(x_i) - y_i)^2$$

Decision Tree

In the literature, tree-based models are the most used models in recommendation systems, they have had a tremendous impact in the world of machine learning and have even been able to defeat deep learning models in some cases, there are different variations and types, all of them more complicated than linear regression.

A decision tree is a predictive modelling apptoaches, to go from observations about an ítem (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the label can take a discrete set of values are called classification trees otherwise are called regression trees. There are various formulations of the decision tree learning, to describe one we present the equation for a ID3. The optimization criterion, in this case, is the average log-likelihood:

$$\frac{1}{N}\sum_{i=1}^{N} y_i ln f_{ID3}(x_i) + (1 - y_i)\ln(1 - y_i)\ln(1 - f_{ID3}(x_i))$$

Where f_{ID3} is a decision tree. The goodnes of a split is stimated by using the criterion called entropy. Entropy is a measure of uncertainty about a random variable. It reaches its mazimum when all values of the random variables are equiprobable. Entropy reaches its minimum when the random variable can have only one value. The entropy of a set of example S is given by:

$$H(S) = -f_{ID3}^{S} ln f_{ID3}^{S} - \left(1 - f_{ID3}^{S}\right)\ln(1 - f_{ID3}^{S})$$

When we split a set of examples by a certain feature j and a threshold t, the entropy of a split, H(S-.S+), is simply a weighted sum of two entropies:

$$H(S_-, S_+) = \frac{|S_-|}{|S_+|}H(S_-) + \frac{|S_+|}{|S|}H(S_+)$$

Random Forest

Random forest is considered a ensemble learning method for classification and regression, it creates a multitude of decision trees at training time independently from each other and outputting the class that is the mode of the clases (i fis classification) in case of regression it uses mean (average) of the individual trees, this characteristics correct the overfitting problem to their training set. This algorithm applies the general technique of bootstrap aggregating, or bagging to tree learners.

Methodology

At its beginning in the anime dataset, we have to observe that the episode variable should be an integer, but due to the presence of the word "unknown" in order to transform it into a numeric variable, we convert the components that contain "unknown" by 1 (which is the representation of mode).

When we are going to look at the missing values in this dataset, we can see that three variables have missing values, which are minimal compared to the dimension of the dataset.

feature	Number of nulls
genre	62
type	25
rating	230

Table 1-. Missing values

To impute these values, We use mode for gender and type, while for the rating, the values are imputed with the median. Since the rating is a skewed distribution, the best way to do it is with this indicator instead of the average.

Then, looking to include gender and type variables, we try to separate it into different columns with the one hot encoding technique.

No significant disturbances are observed in the rating dataset, considering only the users who did not mark the animes they saw, then it only remains to remove the rows where the rating has a value of -1. With this, we proceed to merge the datasets through the anime_id key. The following image shows the resulting dataset. Finally, the dataset dimension is 6.337.239 X 55

	anime_id	Action	Adventure	Cars	Comedy	Dementia	Demons	Drama	Ecchi	Fantasy	...	type_Music	type_ONA	type_OVA	type_Special	type_TV	rating_x	members	episodes	user_id	rating_y
0	32281	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	9.37	200630	1	99	5.0
1	32281	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	9.37	200630	1	152	10.0
2	32281	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	9.37	200630	1	244	10.0
3	32281	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	9.37	200630	1	271	10.0
4	32281	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	9.37	200630	1	322	10.0

Fig 14 -. Header of the merged datasets

Finally, to make the first test of the models, in this part, we only want to see how the three models indicated perform in general terms, the linear regression (benchmark model), decision tree, and random forest. This section also wants to review the relative importance of the features, which we can determine with random forest function.

After defining this, we want to split the final dataset. In this case, we split the data in a 9: 1 ratio since the dataset's size is large enough. Then we scale the data, this is considered a good practice and necessary step because our benchmark model is affected by this process.

Finally, the results obtained in the test set can be shown in the following table:

Metrics	**lr**	**dt**	**rf**
MSE	2.051956	2.04999	2.04739
RMSE	1.43246	1.43177	1.43087
MAE	1.10964	1.10903	1.10846
Training time (seconds)	28	134	379

Table 2-. First results with benchmark model, decision tree and random forest

As could be seen, the decision tree represents an improvement in performance in the training dataset, and the random forest also takes a better performance on it. Nevertheless, if we observe the MAE metric difference, the decision tree and the random forest show an improvement of 0.05% and 0.11%, respectively. So we proceed to review if there is a way to improve the current result.

We analyze the feature importance of each variable in the dataset using the integrated function of random forest, the next table show the relative importance.

Feature	**importance**
rating_x	0,9806
user_id	0,012
members	0,0022
anime_id	0,0015
episodes	0,0013
Action	0,0001
Adventure	0,0001
Comedy	0,0001
Dementia	0,0001
Demons	0,0001
Drama	0,0001
Ecchi	0,0001
Fantasy	0,0001
Game	0,0001
Harem	0,0001

Hentai	0,0001
Horror	0,0001
Kids	0,0001
Mecha	0,0001
Military	0,0001
Parody	0,0001
Romance	0,0001
School	0,0001
Sci-Fi	0,0001
Shounen	0,0001
Supernatural	0,0001
type_Movie	0,0001
type_OVA	0,0001
type_Special	0,0001
type_TV	0,0001
Cars	0
Historical	0
Josei	0
Magic	0
MartialArts	0
Music	0
Mystery	0
Police	0
Psychological	0
Samurai	0
Seinen	0
Shoujo	0
ShoujoAi	0
ShounenAi	0
SliceofLife	0
Space	0
Sports	0
SuperPower	0
Thriller	0
Vampire	0
Yaoi	0
Yuri	0
type_Music	0
type_ONA	0

Table 3-. Relative importance with random forest

In the table above, we could see that the truth is that only a few variables are essential and that practically a large part of the variables referring to gender or type of series have too little or no influence. On the other hand, rating_x, which refers to the average ranking given to the community, seems to be the most influential predictor variable, making much sense, more than the genre or the type of anime. Our decisions are very much based on the community's opinion. For example, even if we are not fans of romantic anime, if we see that the community evaluates a romantic anime very well, we would probably give it a try.

Another critical issue that we have not yet considered and which could negatively affect, as we mentioned before, is that we have unbalanced data with a smaller sample when we have a negative evaluation. After we merge the two datasets, the count for each of these variables is as follows:

Rating	Train sample
1	14.968
2	20.883
3	37.271
4	93.895
5	254.537
6	573.818
7	1.237.440
8	1.481.647
9	1.128.834
10	860.222

Table 4-. Number of samples per label.

In order to solve this problem, we used different techniques to deal with imbalanced data: oversampling techniques (random oversampling and SMOTE), undersampling (random under-sampling) were used, a hybrid of the techniques using SMOTE and TOMELinks techniques was also considered. Unfortunately, none of the proposed techniques generated an improvement. On the contrary, it generated an increase in error, causing the MAE to exceed 1.9 for all the models.

Finally, given the relative importance of the dataset's features, it was considered to keep only the five most influential variables of the random forest, leaving out all the variables related to the gender and type of series.

	anime_id	rating_x	members	episodes	user_id	rating_y
0	437	8.23	117565	1	28669	9.0
1	4181	9.06	456749	24	52535	9.0
2	10080	8.12	194300	12	48664	8.0
3	9936	6.69	104182	12	32860	9.0
4	329	8.38	105044	26	46146	8.0

Fig 15 -. Header of datasets merged with most relevant columns

To search for possible models that could allow us to find better solutions, we decided to build a script to divide the training dataset using the kfold cross-validation technique (create_folds.py). We divided the dataset and allowed ourselves to iterate on different models in model_dispatcher.py and compare them with MAE and the training time. This has a significant advantage since it allows us to iterate on different models using fewer resources, which is not possible using GridSearchCV. Although a regression model is an appropriate solution, we can find a possible solution in classification models, like this case that we use decision tree classifier trees, and that can give a better solution than a decision tree regressor. Considering that the rating takes discrete values, it is possible to use this type of model but keeping the same metric.

After finding the best hyperparameters for the decision tree and random forest, the following results are observed for the test dataset.

Metrics	**lr**	**dt_classifier**	**rf_classifier**
MSE	2.05269	2.30109	2.27031
RMSE	1.43272	1.51693	1.50675
MAE	1.11005	1.10753	1.10092
Training time (seconds)	2	69	578

Table 5-. results with new dataset

As we can see in the previous table, reducing the MAE is achieved with a decision tree classifier and random forest classifier. However, we have a substantial increase in MSE and RMSE, so it can be understood that in absolute terms, we managed to reduce the error. However, there is a more significant presence of large differences, which increase these metrics, which can be considered as the precariousness of the prediction. As the proposed objective is to reduce the MAE, these models would fulfill this purpose, but in another case, the application of this model could be ruled out.

kfold	lr	dt_r	dt_c	rf_r	rf_c
0	1,110231	1,10928	1,106478	1,108655	1,10087
1	1,110602	1,109782	1,105591	1,109107	1,101428
2	1,111274	1,1104615	1,108596	1,109699	1,101321
3	1,111266	1,1104567	1,105722	1,109645	1,101639
4	1,111304	1,1103048	1,109298	1,109591	1,101746
mean	**1,1109354**	**1,110057**	**1,107137**	**1,109339**	**1,101401**
std	**0,000491718**	**0,000515786**	**0,001704789**	**0,00045**	**0,000341**

Table 5-. K-fold cross validation technique

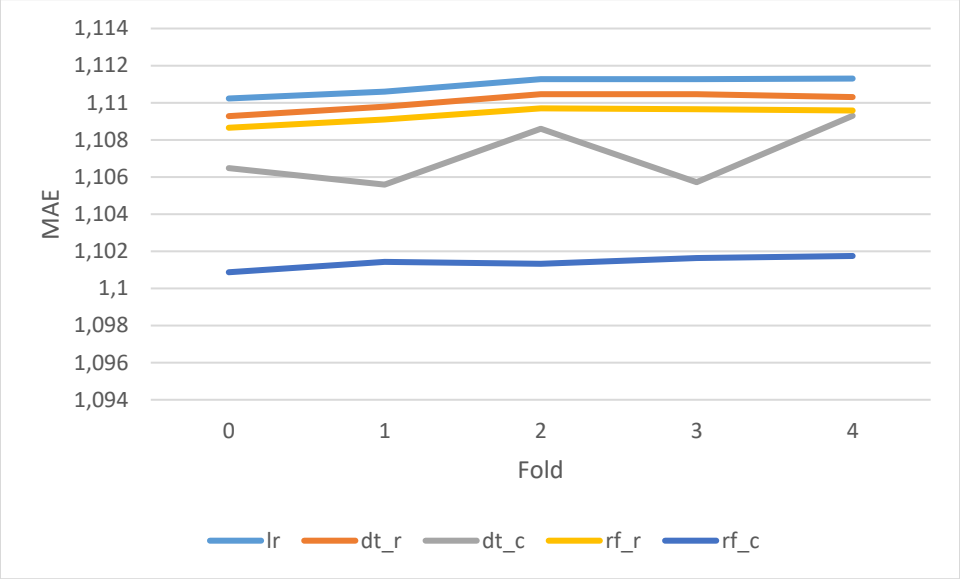

Fig 16-. K-fold cross validation technique

Each fold can be run with the terminal using the train.py script and calling the models from the model_dispatcher.py, as shown in the following image.

```
PS C:\Users\inavarro\Desktop\Scripts\RecSys\project\src> python train.py --fold 0 --model lr
Fold = 0, Accuracy=1.1102318619575975, Training time = 1
```

Fig 17-. Run a specific fold with terminal

Finally, we submit the models to a kfold cross-validation technique, we separate the training data into five equal parts, and we study the four models plus the benchmark model. We can see that the random forest classifier is the one that yields the lowest MAE with an average of 1,101400, followed by the decision tree classifier with an average of 1,107137. The last one has a higher standard deviation than random forest. This is something expected of a decision tree algorithm,

which, unlike a random forest, has a set of weak learners who manage to maintain a more homogeneous result using mode or average.

Conclusion

In this project, we explore the recommendation systems, creating shallow models based on trees that could predict the possible ranking that a user would give to a given anime. This system would help us to find potential animes that the user could enjoy.

In the first instance, we consider all the possible variables in the dataset, considering gender and type that expanded the dataset's dimensions. Once the relative importance was analyzed, we found that these variables had a minimal influence and could be deleted to create another dataset of smaller dimensions.

Dealing with imbalanced data, in this case, failed to generate an improvement in the performance of our models. On the contrary, it ended up increasing the mean absolute error. This phenomenon may be because more noise-generating data has been generated and when we subtract data, we remove relevant vectors for the model to learn. Considering that we have user IDs and animes, it is somewhat challenging to deal with more complex algorithms,

We can also observe that classification algorithms can be a great alternative to improve the MAE; they also give us faster training times than their counterpart in regression. However, they do not necessarily generate a better MSE and RMSE. In some cases, it could be entirely counterproductive, given that there is a more significant presence of large differences.

In future work, incorporate other models such as XGBoost, or a custom ensemble, and a deep learning model to see possible improvements in the dataset.

References

1. Burkov, Andriy. The hundred-page machine learning book. Quebec City, Can.: Andriy Burkov, 2019
2. Burkov, Andriy. Machine Learning Engineering. Quebec City, Can.: Andriy Burkov, 2020
3. Portugal et al. The use of Machine Learning Algorithms in Recommender System: A Systematic Review. Expert Systems with Applications, 2018
4. Thakur Abhishek, Approaching (almost) any machine learning problem, 2020