# AniML : Automatic scoring prediction for Japanese Animation

**Valentin KAO**
Department of Computer Science[*]
Tsinghua University
Beijing, China
`valentin.kao@epitech.eu`

**Emile JONAS**
Department of Computer Science
Tsinghua University
Beijing, China
`emile.jonas@epitech.eu`

## Abstract

Being able to predict the future smash hits is of great importance in decision making specially in the marketing. In spite of the fact that there are many factors that influence the success of a product. It is not generally clear how they influence, this project endeavors to determine these factors using machine learning strategies by predicting a score out of 10 for a project based on its data provided.

## 1 Introduction

The Japanese anime industry is a major worldwide success thanks to the interest of other Asian countries. It is also, of intense interest to both Japanese economists and the public, who became more and more international, because of its high profits and entertainment nature. In 2017, the industry pulled in a record $17.7 billion last year, and boosted by the famous hit, Your Name, growing exports and revenues (1). According to the annual report from The Association of Japanese Animations (AJA), the part of this growth, over the past 4 years, is principally due to expansion in the Chinese market.

Most importantly, an anime tends to run on tight budget. As a comparison, a medium budget American animation project like *Avatar: The Last Airbender* cost 1 million dollars per episode to produce while a classic anime cost between $125,000 and $300,000 to produce. It is a highly competitive environment (50 shows per seasons), with low margin. In this condition, each yen must be spent wisely. A machine learning algorithm giving the opportunity to judge the potential success of a project before it enters production would be an useful tool for production committee in the same way machine learning help bank in the decision process for a loan

## 2 Data

If some dataset about anime existed, most of the time, they are built to determine what an user could potentially like depending on what he have already seen. As such, there was not any dataset for our purpose. We had to build one from scratch.

### 2.1 Data collection

We wanted to build a dataset with the data you could expect to have before entering production : the genre (Action, Romance, Comedy, etc.), the format (1 cour, 2 cours, movie, OVA, etc.), the

[*]`http://ac.cs.tsinghua.edu.cn/`

Table 1: Dataset content

| Key | Description |
|---|---|
| mal_id | ID associated with MyAnimeList |
| title | Item's title |
| type | Item's type (example: TV, Movie, etc.) |
| source | Item's source (example: Manga, Original, etc.) |
| episodes | Item's episode count |
| aired | Associative keys of "from" and "to" in ISO8601 format |
| duration | Item duration per episode |
| genre | Item's genres numerically indexed with array values (example: Action, etc.) |
| studio | Item's studios numerically indexed with array values |
| licensor | Item's licensors numerically indexed with array values |
| related | Item's related item numerically indexed with array values |
| favorites | Item's favorites on MyAnimeList |
| rating | Item age rating (example: PG-13, R+, etc.) |
| rank | Item rank on MyAnimeList |
| score | Item's score on MyAnimeList up to 2 decimal places |
| voice_actor | Item's voice actors numerically indexed with array values (see detail below) |
| staff | Item's staff members numerically indexed with array values (see detail below) |
| voice_actor | detail |
| id | ID associated with MyAnimeList |
| name | Item's name |
| favorites | Item favorites on MyAnimeList |
| staff | detail |
| id | ID associated with MyAnimeList |
| name | Item's name |
| favorites | item favorites on MyAnimeList |
| position | item position in staff in a show |

producer, the studio, the voice actors, the staff, etc. To get those data we looked over the web for a comprehensive anime database, websites like aniDB or MyAnimeList. We decided to go for MyAnimeList for a few reasons. It is a social cataloging application, which means it has both detailed information about anime and a corresponding user aggregated score, and it is the world's most active online anime community and database.

Our main metrics to decide if a project is a success is a user meta score, in our case, the user' score on the website MyAnimeList. Notice that it could have been interesting to use Amazon sales which are often used to determine an anime success, but that kind of data are hard to collect in English in a reliable fashion. Plus, a lot of work has already been made in this area, for example, on Amazon Ranking Stalker(2), there are information about the sales of an anime.

To build our dataset we built a PHP scrapper on top of JikanAPI, an unofficial PHP REST API for MyAnimeList. Data scrapped this way was dumped in a main file called anime.csv and5 sub-file called producer.csv, licensor.csv, studio.csv, staff.csv, voice_actor.csv as some of those information might have several value (multiple studio, staff members, etc.) with the anime_id kept as a "foreign key" to identify the anime related. To not have our scrapper banned for flooding the API with requests, we had to make a request every five seconds. This is made to prevent bots degrading the quality of service for normal user. The whole process would take around 50 hours, hence the scrapper was deployed on a server so the dataset could be built fast enough, and if any problem arise, we could notice it faster. In the end, we managed to build a dataset of a little less than 13,000 anime. Therefore, this dataset should be pretty complete as aniDB list 14,185 anime as of July 2018.

## 2.2 Data cleaning and improvement

We then built a python parser to read our files and create our dataset from it. As expected from raw input, many data were incomplete, either old entries with missing information or new entries where the data has not existed yet. Other entries were removed because they were out of the scope of

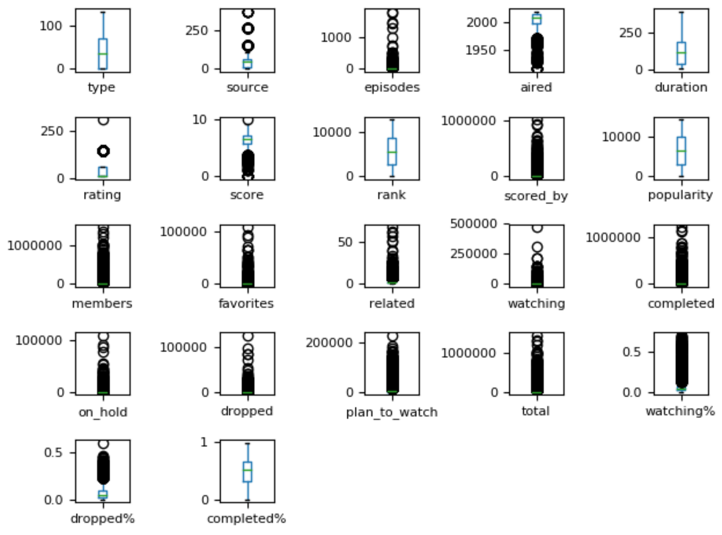

Figure 1: Distribution of values in the early dataset.

our project (18+ for example). Another task for the parser was to replace non-numerical data with weighted IDs, so it can be computed later. This weighted ID are basically integer that represent a string value. For example, "Adventure" genre can't be understood by an algorithm, so it needs to be replaced in a consistent way and with a value that adequately represents it. Finally, the parser writes down the data so it can be read by the CSV parser of the library we're using. It uses the same CSV library to write it down.

After that step, numerical value aren't ready to be feed as it is. There can be huge variation between values 1 and this variation need to be tone down with the standardization and normalization of our data to better fit machine learning estimators requirement. They might behave badly if the individual feature do not more or less look like standard normally distributed data. We used a standard scaler, standardizing features by removing the mean and scaling to unit variance.

## 3 Method

Our work is made from scratch with our own-built dataset, first, we need to choose the best features and eventually, create new ones. Then, with those features, the selection of the most appropriate algorithm is primordial to predict a score close to the real one.

### 3.1 Features selection

We have basically two types of features: "raw" features and "complex" features. Raw features are basically elements from the database like the season of an anime or its studio. Complex features are multiple data combined to give an index. We produced some score to use as a feature, like a producer score, a studio score, etc, but also a voice actor score or a staff score. Studio score are pretty simple as it is built from the number of occurrences of the said studio (How many anime it worked on) and the mean score of the shows produced. It gets a little more complex with voice actor as it takes in account the number of user that consider this voice actor its "favorite". It also takes in account the number of lead actors in a show when it builds the mean score of the shows. Lastly, staff score also include the position as a weight (if this staff member is a director, his work as a director will count more heavily). All voice actor and staff member score are then summed to get the final voice actor/staff member score of an anime. We also used the date to determine on which season the anime was airing

To select features, we tried to determined the correlation of features. It was essential to remove features that were not correlated or worse, negatively correlated. To do that we built a correlation graph to quickly and visually determine which feature was not well correlated. We had to remove the licensors for example that were negatively correlated.

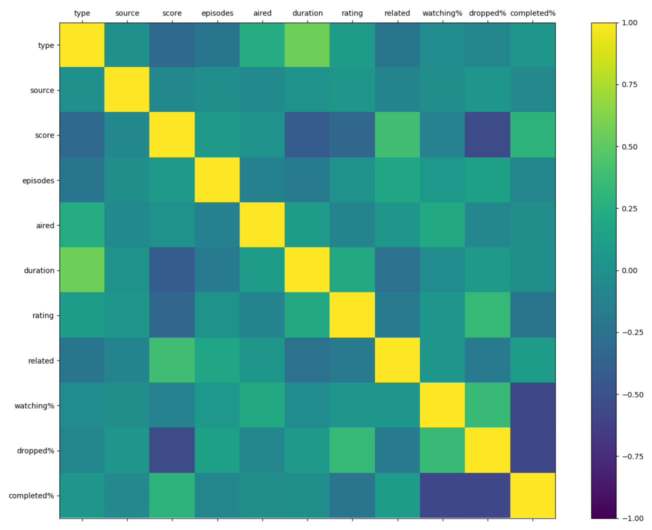

Figure 2: Correlation matrix

After that, we tried different composition of features to see which one worked the best. To get some hint about a proper composition, we first "brute-forced" different combination to see which one was the most efficient. To do so we ranked features by recursively eliminate features. Each feature is weighted using a support vector regression (the coefficients of a linear model). We then selected features by recursively considering smaller and smaller sets of features. From that we derive the importance of each feature. The least important features are removed from the set. This whole process being done recursively. From that we can derive which attributes (and combination) contribute the most to predicting the target attribute.

Another concerns in the feature selection was the number of features we could use as the dataset was incomplete, and the more features you included, the smaller the dataset was. In the end, we stayed flexible, so it's possible to use more features for a more precise result, but also to use less features to have a broader training dataset. Depending on the number of features you want to include, the training set will go from 7,000 entries to 10,000 entries

### 3.2 Evaluation

To know which algorithm will do well for this regression problem, we use 10-fold cross-validation, a technique to evaluate predictive models by partitioning the original sample into a training set to train the model since our dataset' size is good. We decided to train our model on 90% of our dataset and the 10% remaining are used for testing. We will evaluate algorithms using the Mean Squared Error metric, aka MSE, defined by the following formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

with $\hat{\text{Y}}$ is a vector of n predictions, and Y is the vector of observed values of the variable being predicted.

### 3.3 Algorithm Selection

#### 3.3.1 Linear and nonlinear Regression Algorithms

Several algorithms capable of working on this regression problem were selected based on Xueshi Hou and Mladen Marovic et al researches (3) (4)

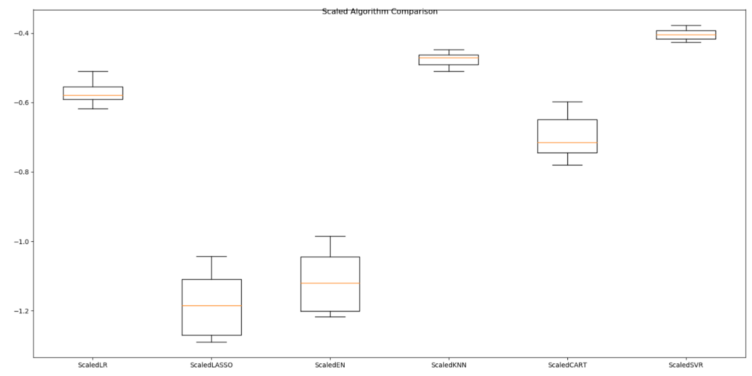

Figure 3: Difference between Linear and Nonlinear algorithms.

Table 2: SVR Settings

| Parameter | Value |
|---|---|
| C | 5.0 |
| Gamma | ’0.1’ |
| Kernel | ’rbf’ |
| Epsilon | 0.1 |
| Tol | 1e-3 |

1. Linear Regression (LR)
2. Lasso Regression (LASSO)
3. ElasticNet (EN)
4. Classification and Regression Trees (CART)
5. Support Vector Regression (SVR)
6. k-Nearest Neighbors (KNN)

The algorithms were performed without any tuning parameters to determine which one would be the most efficient. In the Figure 3, we present the actual scores of different algorithms. The results show that the Support Vector Regressor has the lowest MSE followed closely by the k-Nearest Neighbors algorithm. To make sure that those MSE metrics are significant, the distribution of scores across all cross-validation folds by algorithm would be helpful.

### 3.3.2 Tune regression algorithm

The SVR algorithm achieves good results on our dataset, but it was performed with default values of SVR. Tuning parameters value for machine learning algorithms will definitely improve the performance and the result. In SVR, three parameters have a high impact on the model performance: the kernel, the gamma value, and the epsilon value. Since the library used by our program is scikit-learn, the grid search is the most adequate approach to parameter tuning. Indeed, it will methodically build and evaluate a model for each combination of parameters that we specified.

The Figure 4 shows that SVR coupled with the parameters from Table 3.3.2 are the best for our problem.

### 3.3.3 Ensemble Algorithms

However, due to the willing of improve the result obtained by the SVR method, we decided to use Bagging and Boosting Ensembles. As the way for the linear and nonlinear algorithms, we selected four ensemble methods :

1. AdaBoost (AB)

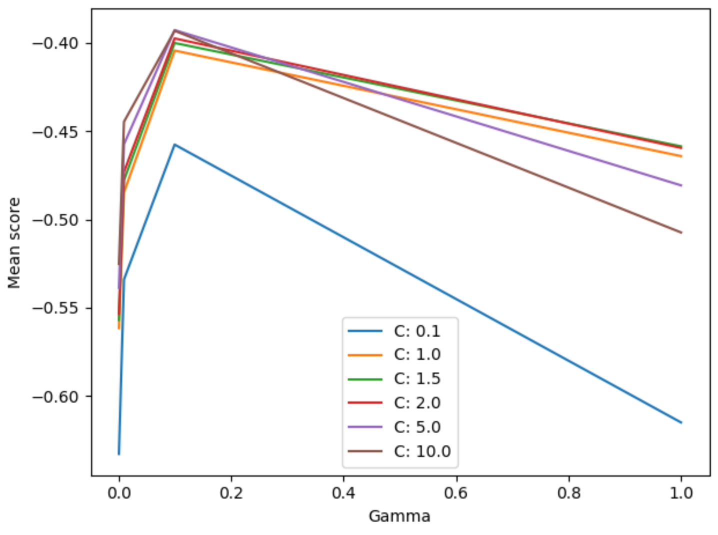

Figure 4: SVR Tuning

Table 3: Evaluation of Ensemble Algorithm

| Algorithm | MSE | Derivation |
|---|---|---|
| AB | -0.719602 | 0.067095 |
| GBM | -0.381320 | 0.019947 |
| RF | -0.384529 | 0.016558 |
| ET | -0.379321 | 0.024441 |

2. Gradient Boosting (GBM)
3. Random Forest (RF)
4. Extra Trees (ET)

We also use the same 10-fold cross-validation on these four methods and for each method, we calculate the mean squared error using the default parameters. The Table 3.3.3 shows the mean and the standard derivation of MSE for each algorithm. Indeed, except for AdaBoost's result, the three other results are significantly better than the results from the regression linear and nonlinear algorithms. Extra Tree has the lowest MSE followed closely by Gradient Boosting, in addition, the Extra Tree has a larger distribution than Gradient Boosting.

### 3.3.4 Tune Ensemble Methods

Since it used default parameters and their score are pretty tight, we performed parameters tuning on two algorithms, Extra Tree and Gradient Boosting. To determine which combination of parameters is the most appropriate, we also use the grid search method provided by scikit-learn. The table 3.3.4 is the result of the grid search on the two algorithms, giving the best result possible with the algorithm.

Table 4: Ensemble Algorithm Settings

| GBM | | ET | |
|---|---|---|---|
| Parameter | Value | Parameter | Value |
| min samples leaf | 1 | learning rate | 0.1 |
| min samples split | 6 | max depth | 7.0 |
| n estimators | 725 | n estimators | 600 |

## 4 Results

Several algorithms were performed on our self-made dataset. Each time we tried to find the most efficient algorithm, for any type of algorithm, linear, nonlinear, boosting or bagging. At the end of experiments, three algorithms were particularly efficient compared to the other algorithms tested.

Using RBF kernel, tuned parameters, and SVR model, we can achieve a MSE as **0.42** on the whole dataset, which is a quite good result. It means that on average, the score predicted is **0.648** different from the real score out of 10. In comparison, the ensemble methods reach better results. The Extra Tree method with tuned parameters can reach a MSE of **0.38938** i.e. **0.6240** of difference compared to the real score, against **0.5820** of difference with the Gradient Boosting algorithm tuned.

This final result is the best we obtained the current configuration of our program and its dataset. This result is closed to the one obtained by Xueshi Hou in his work of predicting score of IMDB movie (3) (4), which was **0.538**. The difference can be explained by the quality of our self-made dataset compared to the official IMDB dataset, but also, by the number of works usable in the datasets.

## 5 Conclusion

This project imagined by ourselves is a regression predictive modeling machine learning problem. By computing multiple linear, nonlinear and ensemble algorithms on a self-made dataset built from scratch, the program finally predicts with a margin error of 5.7% the score of an anime. However, the combination of several models as it was made in the search of Mladen Marovic et al. by combining kNN and SVD methods (4), could give us more efficient and more accurate results.

### Acknowledgments

We are grateful to the Machine Learning Department of Tsinghua University for providing us the material for this project, to Prof. Jun Zhu and Prof. Jie Tang for their guidance and all teaching assistants for their valuable advice.

## References

[1] *Rafael Antonio Pineda (2017, October). "Anime Industry Takes in Record 2.0 Trillion Yen in 2016" retrieved 4th june of 2018 from* `https://www.animenewsnetwork.com/news/2017-10-25/anime-industry-takes-in-record-2.0-trillion-yen-in-2016/.123164`

[2] *Amazon Ranking Stalker website* `http://www27392u.sakura.ne.jp/index.cgi`

[3] *Xueshi Hou, University of California, San Diego* "Prediction of Score on IMDB Movie Dataset"

[4] *Mladen Marovic, Marko Mihokovic, Mladen Miksa, Sinisa Pribil, and Alan Tus* "Automatic movie ratings prediction using machine learning"