Allan Tan - A15403850

Jason Sheu - A15536122

Ronaldo Romano - A15489060

Richard Yang - A15459517

CSE 158 Assignment 2 (Anime Rating Prediction)

1. Dataset and Exploratory Data Analysis

For our assignment, we chose to pull from a Kaggle set which took data from the popular anime review site MyAnimeList in order to make a prediction on how users would rate an anime based on their written review.

We have 3 different datasets that all correlate to each other. We have one dataset that contains all the information about a specific anime, such as synopsis, genre, when it aired, and how many episodes. Before doing any cleaning this dataset contained 19,311 rows and 12 columns. Some of these rows had missing values in them so we decided to drop them as it didn't shrink our data too much, leaving us with 15,875 rows.

Our second dataset contained information about the users (profiles) which contained data such as their birthday, gender, or favorite anime. Again before cleaning it, it had 81,727 rows and 5 columns. The dataset had missing values in the birthday and gender column which is to be expected as some users are not willing to give out that information or are simply too lazy to input them.

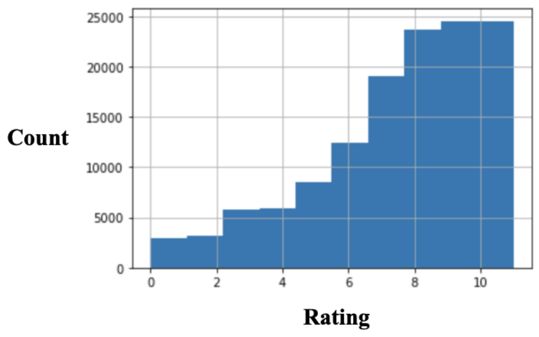
Our last and largest dataset was the one which contained all of the reviews. It had the profile/user UID as well as the anime UID which would allow us to merge the datasets if needed for later on. It also had the review itself along with the given score. This dataset contained 192,112 rows and 7 columns. This dataset did not have any missing values, most likely as users would have to login to write a review on the specific anime they have watched.

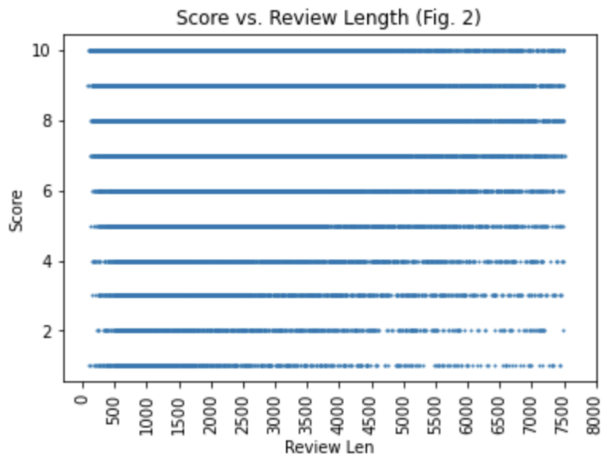
We then went on to clean each of the datasets. For the first dataset, we decided to drop the following columns: img_url, link, synopsis, and aired. We did not believe these were necessary in any of our prediction models we were thinking of attempting. We still had missing values in the episodes, ranked, and score columns but went ahead and dropped those, leaving us with 14336 rows. Looking at the profiles dataset, we decided that we did not need the gender, birthday, and link columns. Once we dropped those we did not have any more missing values. The only thing we did to clean the reviews dataset was drop the link column as it was not important to our model we were trying to build.

After having all 3 datasets cleaned, we wanted to merge the two smaller datasets to the reviews dataset but quickly ran into a problem. We were merging to the "left" but noticed we were getting more rows than what we started with, so we examined the merged dataset a little more and found that there are duplicates in the datasets which were promptly then removed. After removing duplicates from all the datasets, we were left with 14,336 rows for the anime dataset, 47,885 rows for the profiles dataset, and 130,519 rows for the reviews dataset which shrunk the datasets quite a bit but still was enough data. We once again tried to merge the anime dataset to the reviews dataset, this time using an inner join which brings us to a total of 12,6641 rows. After looking more closely at the datasets and figuring out what our predictive task would be, we realized that the profiles dataset were not useful for us and our model so we discarded it.

We then wanted to get a few statistics to better understand our data and also plot data that seemed useful. We first plotted the score values using a bar chart **(Fig. 1)** and from this we see that there is a bias (skew); there are a lot more higher scores than lower ones. We then go on to look at the average value of the score and see that it comes out to be **7.329**. We then computed the average score of the top 10,000, 1,000, and 100 anime and they are very close to the average we had already computed: **7.354**, **7.586**, and **7.964** respectively. When taking a closer look at the score column, there was 1 anime with a score of 0 and another with a score of 11 which we decided to drop.

Fig. 1





We wanted to see if there was a relationship between the length of the review and score it received. After plotting them (Fig. 2), we learned that there is not any correlation between both variables, that is why we decided to not use review length for our final model. For the plot, we sampled only 30% of the dataset, so about 36,000 data points. We did this because the graph would just be straight lines all across, so we did this to better visualize it. At the end, we decided to just go with the reviews dataset as we did not use any of the other features the other datasets had. The length of reviews was not helpful at the end of the day because there was no correlation between them.

2. Predictive Task

With the 3 datasets, we could have approached it multiple ways. One way was taking a look at a user's favorite anime and then recommending them similar anime. Another approach which is the one we went with was seeing if we could predict the rating of an anime based on the text of the review and possibly other features as well. Due to the nature of the datasets and the reviews dataset being the largest, we settled on having the predictive task be predicting a user's rating given the text of the review.

There are two different approaches to such a task: classification and regression. In other words, we can use classification and treat each rating (1-10) as its own class and classify the review into one of ten clusters. The regression-based approach would mean that an exact number would be assigned to each review, most likely in decimal form. Naturally, it seemed better to use classification because the ratings would only be in integer format, not decimal. We decided to go with a mix of both approaches, however, to see which approach would net a better result.

In the case of classification models, we would evaluate the performance using the accuracy and mean squared error of the rating prediction. Because of our biased data, we would also be examining the F-scores, calculated from precision and recall, for each of the classes (1-10) to ensure that the model is not simply predicting high ratings to improve accuracy. In terms of regression models, the evaluation criteria would also consist of mean squared error. Additionally, we could round the regression-based predictions to the nearest integer and obtain accuracy as a way to compare its performance with the classification models.

For the models, 70% of the data was used as training and the other 30% was used for testing. To avoid overfitting to the test data, we used Sklearn's cross-validation library in order to split the training data into k-folds, allowing us to tune hyperparameters and evaluate performance across models when determining the best model. We decided to start with the naive baseline of predicting the average rating regardless of review text. Throughout the model design and testing process, we establish several other baselines based on material covered in class, such as logistic regression on the length of the review.

In terms of features, we plan to primarily look at the text of each of the reviews. From the text, we can extract features such as the most popular words and create a frequency distribution. In order to accomplish this, the data would first need to be preprocessed according to conventional NLP standards, such as stemming, removing stopwords, removing punctuation, etc. In total, we were able to remove stopwords, punctuation as well as lowercase all of the text in the reviews. With this done, we could obtain a list of the most commonly used words and use its frequency distribution as a feature vector. Another feature set that we could use is the TF-IDF technique on the review text to extract particularly important words or phrases. This would be done using Sklearn's tfidfVectorizer and the output would then be used as input into another one of Sklearn's classification or regression models. An additional feature we could include in the length of the review text. As seen in the EDA portion, there could be a possible relationship between review length and rating, and using it as a feature could give us additional insight with our model.

3. Model

As mentioned in the predictive task section, we initially had a model that would predict the average rating irrespective of the contents of the review text. This model had particularly low accuracy and a high MSE, so there was plenty of room for improvement. To surpass this, we first established another baseline model that uses logistic regression to predict the rating the user would give an anime based on the length of the review (the number of characters in it). With a slightly higher accuracy and lower MSE, this would be the new baseline that we would evaluate the newer experimental models against.

Because our predictive task involved using classification and regression-based models, we experimented with various models from each section of Sklearn. Our approach was similar to that of the shotgun approach, where we chose random models from each section of Sklearn, such as SVM, tree, ensemble, linear_model, etc. Specifically, for classification, we chose: RandomForestClassifier, DecisionTreeClassifier, ComplementNB, and LinearSVC. For regression, we went with LogisticRegression and Ridge. The reason we went with more models on the classification side is because we initially wanted to do only classification but realized that regression could pose a valuable point of evaluation/comparison. Each model will be further explained in detail in the following.

First we attempted to switch from logistic regression to a more regularized linear model using Sklearn's Ridge function. With this, we established a regularization pipeline in which we tested different lambda values, ranging from 0.001 to 10000. Unfortunately, we were unable to make any significant improvement in MSE nor accuracy, regardless of the lambda value used.

Because Ridge regression was not netting noticeable results, we decided to switch back to logistic regression and instead experiment on features that could be extracted from the data. In our experimentation, we found that the n-gram model did not work very well, and in fact lowered our accuracy or would not go much higher than the baseline accuracy. Specifically, we tried a combination of unigrams, bigrams, trigrams, and a combination of each. While we were unable to obtain favorable results, we learned that unigrams were likely the best n-gram to use moving forward in experimentation.

Building off this, we attempted a logistic regression model using the bag-of-words approach and strictly used unigrams. The model netted a higher accuracy than our baseline and was the highest so far. The only parameter set in the regression function was fit_intercept to be false, since we already include a bias term during feature extraction. We tried different parameters but they netted in also the same if not insignificant changes in the accuracy and MSE. Next, we attempted to add the additional feature of review length as well as stemming and removing stop words. Again, there was no significant improvement in MSE nor accuracy, and instead actually lowered our results when including review length, so a decision was made to exclude the feature from future experiments.

Improvement finally came in the form of TF-IDF as a feature. Using Sklearn's tfidfVectorizer, we were able to extract the frequencies from the review text and use them as features. By plugging in the output of the vectorizer as input into the logistic regression function, we were able to achieve a better MSE and accuracy. Additionally, the parameters of the regression function were changed to the newton-cg solver and a balanced class weight. The reasoning for this is because the newton-cg solver trains in a faster time than the default solver and a balanced class weight was necessary due to our skewed data.

After logistic regression, we tried several other classifiers: RandomForestClassifier, DecisonTreeClassifier, ComplementNB, and LinearSVC. Similarly, the output of the TF-IDF vectorizer was used as input into these functions. Random forest, decision tree, and naive bayes all had insignificant performance. Regardless of the functions' parameters, there was little to no improvement in the MSE and accuracy. The reasoning for this is likely because of the bias inherent in the data. It was for this reason specifically, that ComplementNB was chosen instead of MultinomialNB. According to Sklearn, ComplementNB performs better on unbalanced datasets, and while it yielded a small improvement when compared to MultinomialNB, it was not significant in the broader scope. The last classifier, LinearSVC, surprisingly resulted in an almost equal accuracy. This could possibly be the case because the data is more suited for linear classification as opposed to being divided up into trees.

In conclusion, the 3 best performing models were LinearSVC with TF-IDF, LogisticRegression with TF-IDF, and LogisticRegression with bag-of-words, with LogisticRegression TF-IDF, slightly edging out the others. While the accuracies were fairly close among these models, the ultimate strength of the TF-IDF LogisticRegression was its ability to produce higher F-scores. This means that it's able to better generalize over real world reviews as opposed to the others. A weakness of this model compared to the other models is the extra time it takes to vectorize and train. Features could be extracted quicker using the bag-of-words approach and LinearSVC was able to train quicker than LogisticRegression. Regardless, the training time was not as extreme as the RandomForestClassifer and DecisionTreeClassifier, making the higher F-score - setup time tradeoff worth it.

4. Literature

As previously mentioned, our dataset came from Kaggle which helped provide information featuring different animes, profiles and reviews. Similar datasets on Kaggle, based on anime reviews, were used in a variety of ways but most prominently were used for creating recommenders; specifically for recommending different types of anime to users.

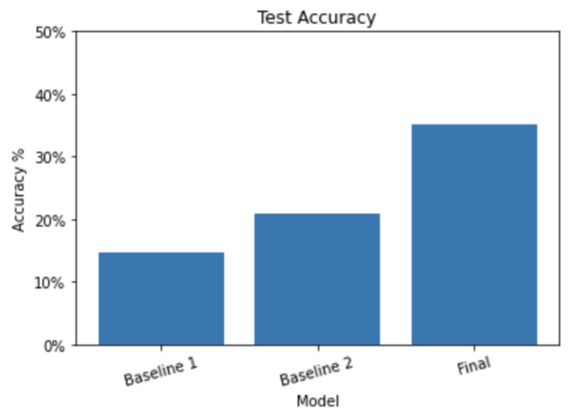
An example of this could come from a user who created a recommender system that recommended different anime to users based on their user anime rating history [1]. The user used average ratings of users to find out what generally would be considered them "liking" an anime or not based on their already previous average ratings and then pulled animes to recommend from characteristics of clusters that were created.

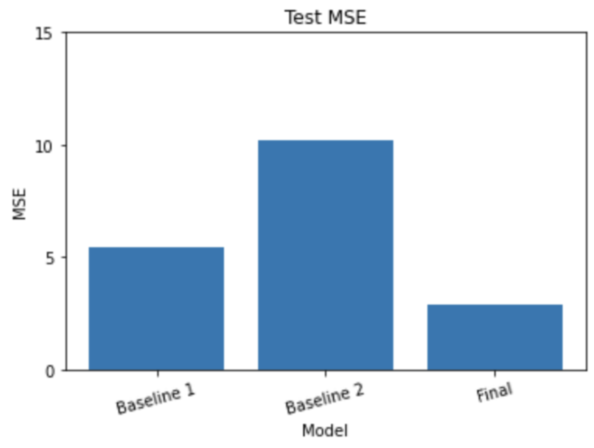
When it comes to studies based on our research of trying to predict a score based on a review, there were many to choose from. A great example of this comes from Jason Jong, from Stanford, who also created a predictive rating system based on Yelp reviews [3]. Jong created a Natives Baye Classifier for reviews to train its words for an either positive or negative rating. Then Jong builds a bag-of-words model for regression. Similar to this research by Jong, we also tried and used a Bag of Words model for our regression analysis to see if it would help and improve our results and it resulted in being one of the better models we tested.

A more state of the art example could come from a 2020 article written by Ankit Taparia and Tanmay Bagla [4]. They too were trying to find the best way to predict a rating based on Amazon reviews. Similar to our project they tested out various techniques in order to find the best accuracy for predicting the rating. They used multiple different classifiers such as Multinomial Bayes, Logistic Regression and Linear SVV and found that Logistic Regression gave them the best accuracy. This correlates to our dataset as we set our baseline based on a logistic regression model. They also used TF-IDF vectorization to help give significance and weight to words that appear in the reviews to help organize their model better. Similar to their model, we also used TF-IDF and found the greatest success as our accuracy was at its highest.

Overall, from the many related works we see that many of the features and functions that helped their model succeed likewise helped ours too. Many of the core functionality in their models helped provide great aid into achieving our best model overall.

5. Results

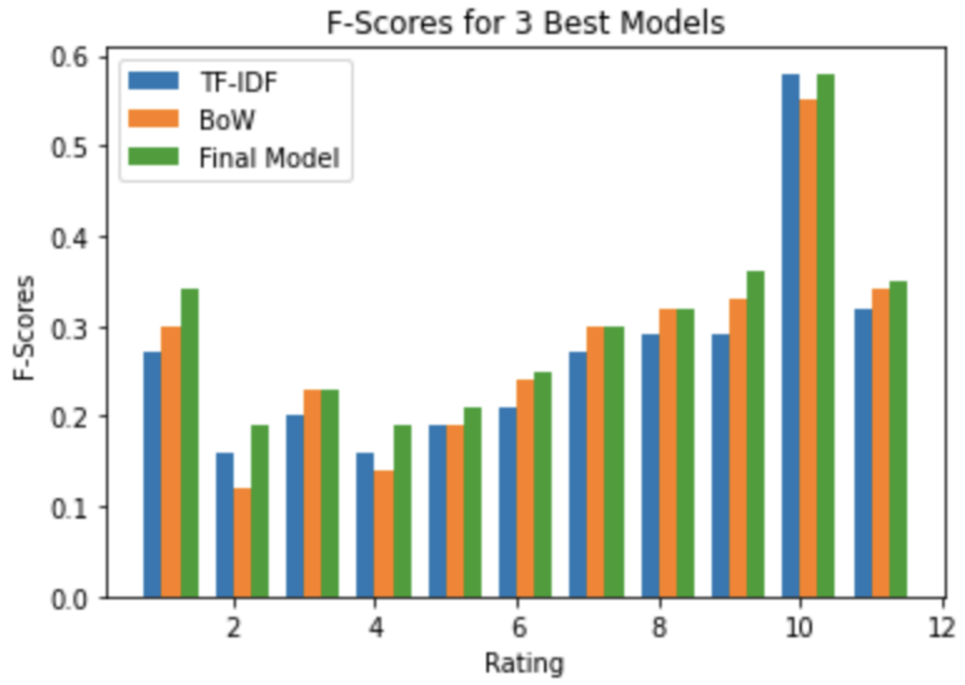




Overall, we attempted several different models: a basic logistic regression, n-gram models, tf-idf, random forests and decision trees to see which one performed the best on our predictive task. Our naive baseline predictor, which simply output the average rating irrespective of the data, had an accuracy of 0.146 and a mean squared error 5.44 on the test set. Our second baseline, logistic regression with the review length as a feature, netted a test accuracy of 0.2 and test MSE of 10.2. The final model we decided to use was a logistic regression model utilizing a TF-IDF vectorizer for feature extraction, which ended up having a test set accuracy of 0.35 and mean squared error of 2.872. The comparison of our final model (in terms of test MSE and test accuracy) can be seen in the graphs above. Our other alternative models, such as our random forest, decision tree, or n-gram models, did not perform as well as our TF-IDF model did. In our testing, we found that the best feature representation was converting the text into tf-idf representations, along with converting the text into a bag of words. Stemming, removing stop words, and using review length was not very effective in lowering the mean squared error or improving the accuracy of our model.

Method	**Feature**	**Test Accuracy**	**Test MSE**	**Average F1 Score**
RandomForestClassifier	TF-IDF	0.27	6.76	0.173
DecisionTreeClassifier	TF-IDF	0.21	7.11	0.16
ComplementNB	TF-IDF	0.28	6.14	0.176
LinearSVC	TF-IDF	0.32	4.68	0.275
LogReg	TF-IDF	0.35	3.31	0.30
LogReg	Bag-of-Words	0.34	3.37	0.27
Ridge	Bag-of-Words	0.24	3.3	-

The performance of our best model can be compared to our other models using the table above. Additionally, as mentioned in the Model section, the best model was chosen not only for its higher accuracy and lower MSE but also for its better F-Scores. These can be visualized in the graph below. A better F-Score allows for the mode to better generalize for real world reviews.It is important to note that while our “best” model had a higher accuracy and F-score than most of the others, it does not necessarily mean it is a good model. In terms of the significance of its results, an abysmally low accuracy of 0.35 is most likely attributed to the inherent bias in the data. Unfortunately, because of this all the accuracies across all models were particularly low. A solution to this would be to resample the data from the dataset, but the lower ratings (1, 2, 3) did not have enough samples. As a result, we were unable to use a balanced dataset when training these models, ultimately causing them to fail. This was a good learning experience, however, as we now know the importance of picking a proper dataset.



Regarding successful and unsuccessful feature representations, TF-IDF most closely captures the most important words in a review, so with many negative words like bad, terrible, mediocre, the rating is likely to be worse. Similarly, our bag-of-words approach is able to identify particular words that have a positive or negative connotation. These features together summarize effective forms of feature representation for our project. For unsuccessful feature representations, we found that including review length as a feature did not help at all. This is likely due to the highly distributed nature of review lengths, with many low and high rated reviews having similar review lengths.

Works Cited:

[1] LastNight, “User Clustering for Anime Recommendation” *Kaggle*, (2018).

https://www.kaggle.com/tanetboss/user-clustering-for-anime-recommendation/notebook

[2] Fatih Bilgin, “Story of Anime” *Kaggle*, (2019).

https://www.kaggle.com/fatihbilgin/story-of-anime

[3] Jason Jong, “Predicting Rating with Sentiment Analysis” Stanford (December 16, 2011).

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.375.2409&rep=rep1&type=pdf

[4] Taparia, Ankit and Bagla, Tanmay, “Sentiment Analysis: Predicting Product Reviews’ Ratings using Online Customer Reviews” (May 15, 2020).

https://ssrn.com/abstract=3655308 or http://dx.doi.org/10.2139/ssrn.3655308