Anime Recommendation System

CSPB4502 Spring 2022

Matthew Fournier
Matthew.Fournier@colorado.edu

Christie Hui
chhu9206@colorado.edu

ABSTRACT

This final project paper presents various methods for creating an interactive anime series recommendation system based on existing user and user rating data obtained from the anime cataloging site, MyAnimeList.net, via the Kaggle dataset website. Some interesting questions which our project aimed to answer were :

- Which genres (of anime) contribute to ambiguous users' watching decisions?
- How much do features such as rating, episode count, and anime type influence a user's viewing choices?
- Is the popularity of an anime or similarity to other users' profiles/viewing preferences more significant in impacting a user's preferences?

The methods discussed in this report include a simple recommendation system based on a user-provided input of an anime series title, a simple predictive model using classification and a decision tree, and the K-Nearest Neighbors predictive algorithm. The results from these methods showed that some of the highest ranked features included genre and the number of ratings given for a particular anime title. Thus, these features were particularly meaningful when our algorithms/models were recommending an anime series title based on user input.

INTRODUCTION

The aim of this project is to create an anime recommendation system for users based on existing user data obtained from MyAnimeList.net, which is essentially a cataloging site which allows users to keep lists of which anime they've seen/plan to see, view user reviews of anime titles, and rate shows based on a scale from 1-10[1]. The motivation behind choosing this topic is due to a shared common interest in anime as well as prior familiarity with MyAnimeList site functionality. The knowledge and results gained from the analysis of this data set may be beneficial for future personal decisions regarding choosing an anime to watch.

As previously mentioned, based on certain features from the data set – such as anime titles, genres, and ratings, to name a few – we aim to discover and answer several interesting questions.

These questions are important because they provided a scope for our project, and by answering these questions, we were able to gain some new insight into how certain factors influence the popularity of an anime series. These questions are also helpful for viewers who may be new to watching anime series, and initially don't know where to begin looking or which metrics to use to define the popularity of a certain series they may be interested in. For example, a user who may not know where to start could potentially get an idea of which anime they might like, based on other existing users' preferences who have similar genre preferences. Lastly, while we plan to explore the questions previously mentioned, there are certainly many more interesting questions which could be investigated. In the end, we decided to keep the realm

[1] Summary taken from https://myanimelist.net/about.php

of our questions a bit smaller due to the scope of our project purposes since our recommendation system is primarily centered around the purpose of recommending just anime television series (so excluding other types such as OVAs, special episodes, etc).

RELATED WORK

Prior work on this topic can be seen on popular anime streaming platforms such as Crunchyroll[2], which has an existing recommendation system in place which suggests various titles to a user based on the users' previous watch history on the site. The MyAnimeList site, which is where the user data in our data sets is taken from, also provides a list of recommendations to users who have added anime entries to their profile. However, these recommendations differ from those of Crunchyroll's since they also incorporate other users' unique feedback to support the recommendation[3].

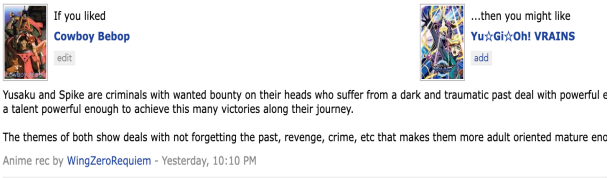

Figure 1: An example of recommendation with unique user commentary provided by MyAnimeList.net.

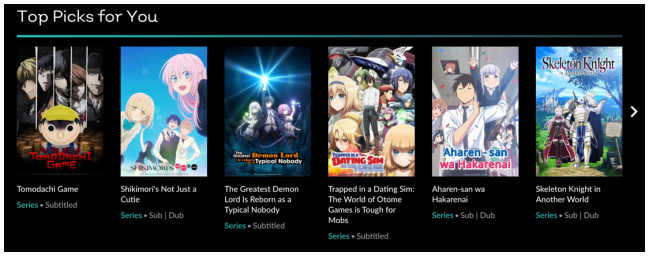

Figure 2: An example of recommendations provided by Crunchyroll, typically displayed on the user's home page.

[2] https://www.crunchyroll.com
[3] https://myanimelist.net/recommendations.php?s=recentrecs&t=anime

DATA SET

The data sets that our project primarily used came together in a file titled Anime Recommendations Database, which was obtained from Kaggle.com. The data sets can be accessed via this URL: https://www.kaggle.com/CooperUnion/anime-recommendations-database. The file is comprised of two separate CSV files, Anime.csv and Rating.csv, which contains information on user preference data from 73,516 users on 12,294 different anime titles[4]. The Anime.csv file contains the columns:

- anime_id (the unique identifier assigned to an anime title by MyAnimeList)
- name (full name of anime title)
- genre (comma separated list of genres for an anime title)
- type (whether the anime is a movie, OVA, TV episode, etc.)
- number of episodes (how many episodes for a given anime title, given a value of 1 if the type is a movie)
- rating (average rating out of 10 for an anime title)
- members (the amount of users on the site who have added a particular anime to their profile)

The Rating.csv file contains the features:

- user_id (a randomly generated user ID)
- anime_id (the unique identifier assigned to an anime that the particular user has rated)
- rating (the rating out of 1-10 that the particular user has assigned)

[4] From the description of the dataset found at https://www.kaggle.com/CooperUnion/anime-recommendations-database

TOOLS

The tools we used for this project include:

- Jupyter Notebook (for formatting and code)
- Pandas, numpy, sklearn, scikitlearn (for calculations, EDA, data processing, model creation, and data analysis)
- Matplotlib, seaborn (for visualizations)

DATA PREPROCESSING

After we initially loaded the Anime.csv and Rating.csv data sets into Pandas data frames, we concluded that the data sets obtained from were already quite clean. In addition, we decided not to incorporate supplemental data sets since we concluded that we already had an appropriate amount of data points to work with. Despite both data sets being relatively clean, we still needed to perform a slight amount of data preprocessing on them in order to make our Pandas data frames easier to work with for future analysis.

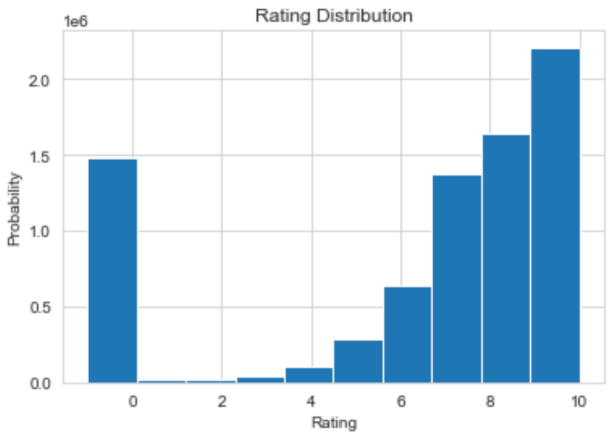

Figure 3: Unprocessed data distribution for the user ratings in the Rating.csv file.

```
# total counts for each rating

users.rating.value_counts()
```

```
 8     1646019
-1     1476496
 7     1375287
 9     1254096
 10     955715
 6      637775
 5      282806
 4      104291
 3       41453
 2       23150
 1       16649
Name: rating, dtype: int64
```

Figure 4: Total counts for each user rating in the Rating.csv file, sorted in descending order.

First, we examined the overall distribution of the ratings in the Rating.csv file. As seen from Figures 3 and 4, in the Rating.csv file, the "rating" column (which specifies the rating out of 1-10 that the user has assigned to a particular anime) contains approximately 1,476,496 counts (out of 7,813,737) of the integer value -1. Per the description of Rating.csv from Kaggle, the numeric value -1 in the "rating" column represents that a user has added the anime to their profile but did not designate a rating, whether intentionally or not. To account for these values, we simply dropped these rows, since we went ahead and assumed these values would not contribute meaningfully to our analysis and discussion portions. One can observe the change in the distribution of the ratings after processing out the -1 values in the histogram shown in Figure 5.

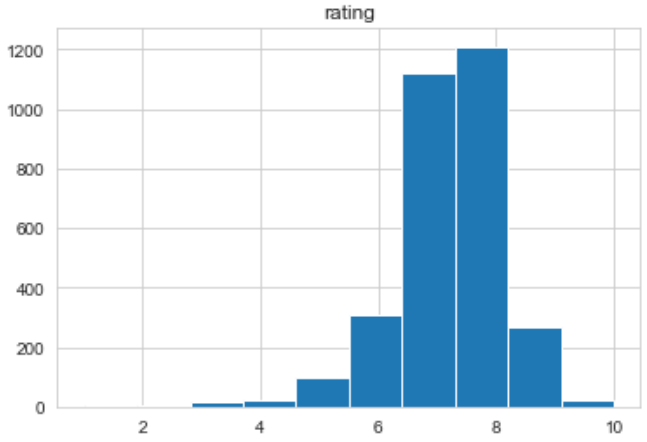

Figure 5: Processed data distribution for the user ratings after filtering out the -1 values, grouping the anime by anime_id and removing non-TV show types.

Our next step was to update our data frame containing the Anime.csv file data to include only rows with the value of "TV" in the "type" column, since our recommendation system will only be recommending television series and not other types of media (ex. OVAs, special episodes, movies, etc.).

Lastly, after dropping rows with the value of -1 in the "rating" column and rows with values other than "TV" in the "type" column, we merged our two Pandas data frames based on the "anime_id" attribute in order to get the name of the anime titles from the Anime.csv file synced up with each rating from the Rating.csv file. By merging the two data frames together, we were able to work with a more convenient data frame as well as more closely explore the relationship between attributes such as "rating" and "user_id".

```
rated_anime.groupby('name')['rating'].count().sort_values(ascending=False).head(10)
```

```
name
Death Note                          34226
Sword Art Online                    26310
Shingeki no Kyojin                  25290
Code Geass: Hangyaku no Lelouch     24126
Angel Beats!                        23565
Elfen Lied                          23528
Naruto                              22071
Fullmetal Alchemist: Brotherhood    21494
Fullmetal Alchemist                 21332
Code Geass: Hangyaku no Lelouch R2  21124
Name: rating, dtype: int64
```

Figure 6: Total "rating" count values for the anime titles in the merged data frame, sorted in descending order.

For example, in Figure 6, we explored the relationship between the "user_id" and "rating" attributes. We can observe the top 10 anime titles which contain the highest amount of ratings among the ones in the merged data frame. One question which might arise from this observation would be: Would the likelihood of an anime having a higher rating overall (in comparison to other anime titles with very few or little ratings, leading to potentially skewed ratings) be increased if the number of users that have watched/rated it is higher? We decided to do some further investigation in our Exploratory Data Analysis portion primarily centered around the "ratings" attribute in the merged data frame.

EXPLORATORY DATA ANALYSIS

The approach that was decided upon for the Exploratory Data Analysis portion of the project was to look for interesting features in the data that we could pick for future in depth exploration. The main bulk of the EDA consists of examining the data points from the Anime and Rating data sets, checking for anomalies or unwanted aspects in the data, data preprocessing, and data abstraction.

Upon examining the total counts of ratings per user, we noticed that there were users who had only assigned a rating to an anime entry once. For the purposes of our recommendation system, we decided to assume that these entries would not contribute

meaningfully to our analysis and development. A minimum threshold of at least 200 ratings given per user was imposed on the data frame, as shown in Figures 7 and 8.

```
rated_anime['user_id'].value_counts()
```

```
42635    1953
57620    1485
59643    1442
45659    1315
7345     1182
         ...
39065       1
18558       1
18373       1
33195       1
66215       1
Name: user_id, Length: 68929, dtype: int64
```

Figure 7: Total "rating" count values for each user (row) in the merged data frame, sorted in descending order.

```
counts = rated_anime['user_id'].value_counts()
rated_anime = rated_anime[rated_anime['user_id'].isin(counts[counts >= 200].index)]
rated_anime['user_id'].value_counts()
```

```
42635    1953
57620    1485
59643    1442
45659    1315
7345     1182
         ...
61438     200
16375     200
62483     200
20100     200
24074     200
Name: user_id, Length: 4687, dtype: int64
```

Figure 8: Total "rating" count values each user (row) in the merged data frame, sorted in descending order, after imposing the minimum rating amount threshold.

After imposing the minimum rating amount threshold, we also took another look at the distribution of ratings in the merged data frame after data preprocessing, as shown in the joint histogram and scatterplot in Figure 9.

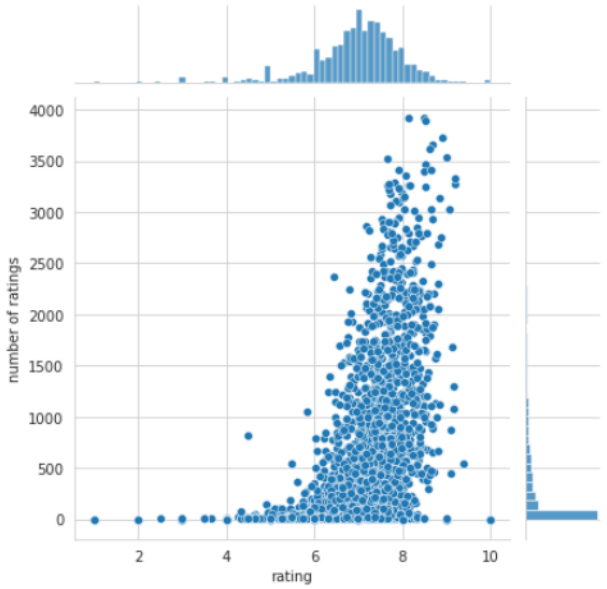

Figure 9: Scatterplot of the rating distribution in the filtered data frame, abstracted to the user level. The mode appears to be approximately within the range of 7 to 8.

Based on the scatterplot in Figure 9, we can observe that the more users there are that assign a rating for an anime title, the higher the likelihood is that it would receive a higher rating. For example, one of the highest data points in the scatterplot in the range of approximately 3,500+ ratings also appears to have one of the highest ratings in the data frame within the range of 7 to 8. This is interesting because it is not necessarily intuitive information that we can abstract from this chart, which is really the whole goal of the project. It is likely that the model will pick up on this trend when it is making predictions and will probably be one of the best features. In addition, there appears to be quite a change when we aggregate the data to this level, with a shift from the higher end of the rating scale to a heavier concentration around the 7-8 rating. This is likely due to polarized ratings on both ends balancing out with middle of the road ratings. There is also an extremely low number of shows that are at the bottom end of the spectrum which means that we will have to make sure that our models do not

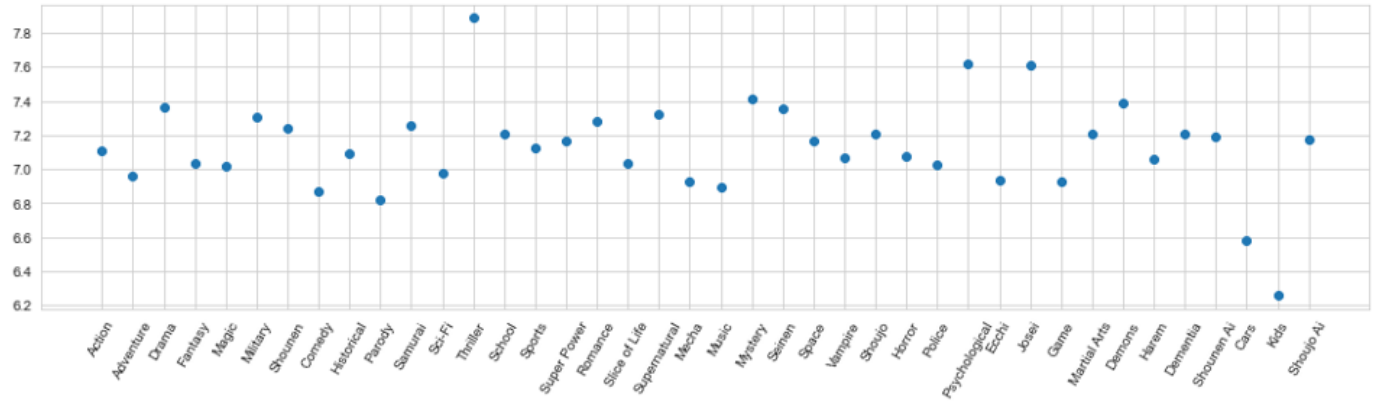

Figure 10: The graph shows the average rating for each genre represented in the Anime.csv data set.

skew to the positive side of the scale. This is an important observation because models more often than not will take the road of least resistance when predicting an outcome. Based on the distribution of the data, it seems that the distribution will be adequate to make a meaningful prediction. Moreover, by including several outliers from the data, the developed model will be able to make stronger predictions if there really is a degree of correlation.

The graph shown in Figure 10 can help shed some more light into more abstractions of the data. In order to create this visualization, an algorithm was developed that abstracts the data to the genre level. This was initially difficult to achieve because of the way the data was originally presented in the data set. The genre attribute values were formatted in a string list as values that were sepereated by commas and a space. Since many of the data points have several different values under the genre attribute in the Anime.csv data set, we also ran a loop that would capture the combined ratings for each of the genres for each individual data point. We also kept an accumulator counter variable for each of the genres as a whole by using a Python dictionary data structure. After all of the loops were complete, the totaled rating for each of the categories along with the total count for each category were used to come up with an average rating, which is shown above in Figure 10. The graph shown allows one to observe that there is arguably some level of variance in terms of average rating within each of the genres. While it initially seems that the average values for the genres are not too far apart at first glance, if observed more closely, one can see that there is a decent amount of variance, albeit not as apparent. For example, the data point for the average rating for anime titles under the “Kids” genre is significantly lower than the average rating values for the other genres; in contrast, the data point for the average rating for anime titles under the “Thriller” genre is noticeably higher than its other counterparts. Based on these observations, we presume that this will likely be the source of some of the negative predictions especially when a user has previously rated a certain genre as low (somewhere along the lines of a rating of 5 or below). It is quite likely that if the rating is consistent with the average values as shown in Figure 5 for anime titles under a particular genre, the algorithm will not suggest many, or any other shows with that same genre.

SIMPLE RECOMMENDATION SYSTEM

Our first goal after completing the initial Exploratory Data Analysis was to create a simple recommendation system that would generate an anime series

recommendation based on existing user data (i.e. ratings) and a user-provided title. The simple recommendation system utilizes a pivot table and the correlation between user ratings and anime_id. For this algorithm, we also imposed a minimum threshold of 1000 ratings for each anime to be considered as a recommendation in the system, in order to allow for a more realistic and reliable recommendation. After filtering the merged data frame with this imposed condition, we observed that the number of rows in the data frame decreased from 1,456,837 to 953,581, which is not a small decrease but still leaves plenty of data points for the algorithm to work with.

We also created a sparse matrix with the user_id attribute as the rows and the name of the anime titles as the columns, with each cell containing the rating given by the user for a particular anime. After creating the sparse matrix, we also defined a function, titled find_correlation, which accepts two parameters, a Pandas dataframe and a user-provided anime title to generate recommendations from. The results from testing our simple algorithm utilizing the sparse matrix and correlation are shown in Figures 11 and 12.

```
anime1 = 'Naruto'

# find recommendation for user input of Naruto
find_correlation(df_rec, anime1).head(10)
```

name	Correlation
Naruto	1.000000
Bleach	0.426035
Fairy Tail	0.307742
Katekyo Hitman Reborn!	0.280213
Dragon Ball Z	0.263396
D.Gray-man	0.249084
Dragon Ball	0.244696
Dragon Ball GT	0.235358
Shijou Saikyou no Deshi Kenichi	0.221060
Ao no Exorcist	0.217335

Figure 11: Example of recommendations generated by the simple recommendation system for a user provided input of "Naruto".

```
anime2 = 'Psycho-Pass'

# find recommendation for user input of Psycho-Pass
find_correlation(df_rec, anime2).head(10)
```

name	Correlation
Psycho-Pass	1.000000
Psycho-Pass 2	0.624337
Zankyou no Terror	0.383839
Death Parade	0.341469
Shinsekai yori	0.335916
Shingeki no Kyojin	0.309546
Zetsuen no Tempest	0.307261
Steins;Gate	0.295788
Kiseijuu: Sei no Kakuritsu	0.292112
Noragami	0.286894

Figure 12: Example of recommendations generated by the simple recommendation system for a user provided input of "Psycho-Pass".

MAIN TECHNIQUES APPLIED

In the beginning of the project, we had to identify what problem we were trying to solve with the project. We identified that we are trying to make a world class recommendation system. When we break down a recommendation system there are a few things that are key, namely the user assigned ratings and as many columns of features as possible. There is also the main question of: How do we measure the success of our model? Simply put, there are many ways to measure, but in this case being centred around ratings when we want "world class" as an outcome makes intuitive sense. Now, it may not be as intuitive to think of how to implement this measurement. Our approach was to take the ratings and then add a new column to the data set that indicated whether the rating was above a 7.

Admittedly after a few iterations of guess and check, 7 ended up being the best number for us to choose as the minimum line for measured satisfaction of a recommendation. This had to do with the model performance which will be further elaborated upon later. Now that we had a metric, we had to do quite a lot of preprocessing to the data to prepare it for training and eventually fitting; it entailed developing some custom hot encoding for a difficult list of strings, which added quite a lot of features to the data set. Our approach was to verify that our decision to go with the value of 7 was correct by comparing it to other models at higher ratings to see if it was possible to make it even "better", objectively. The next portion of the document will go over the main results that we had during the project and what they mean.

KEY RESULTS

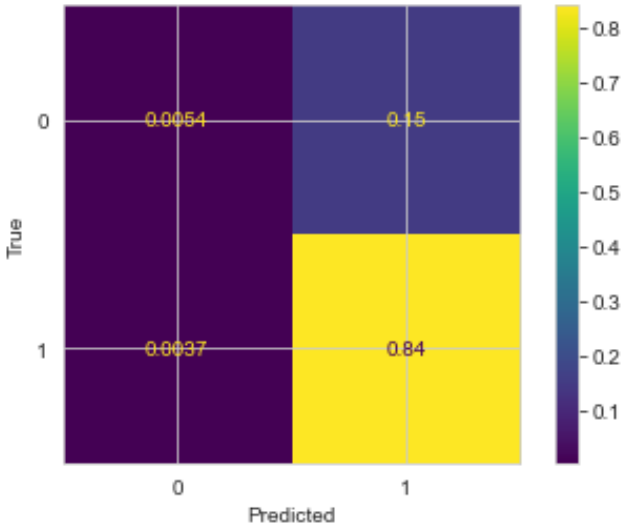

Figure 13: Model 7 Decision Tree Confusion Matrix

The first thing that we did was make sure which Classification type best fit the data. To test this, we compared the Decision Tree and the Random Forest methods against one another. Figure 13 shows the confusion matrix of the Decision tree, which gives us insight into how the model is performing and predicting. Some of the key metrics here were that it had an overall recall value of 0.85 and precision value of 0.81, along with an F1-Score of 0.78. These are really great numbers to achieve as it shows that the model is doing more than just naive selection.

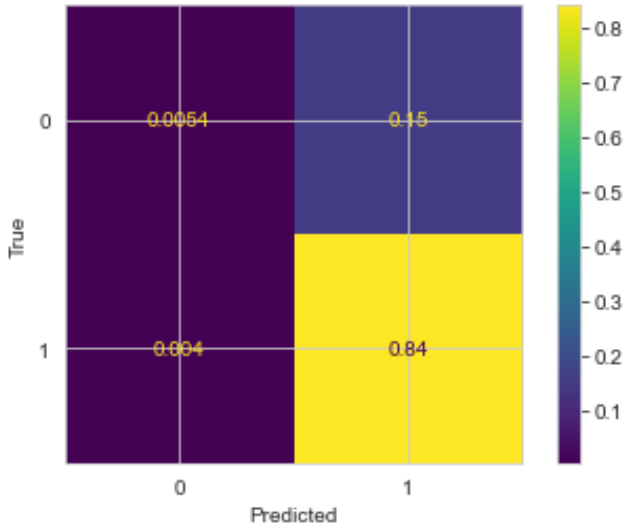

Figure 14: Model 7 Random Forest Confusion Matrix

However, just to see if there was a chance that the Random Forest algorithm could perform better than the Decision Tree algorithm, we trained a Random Forest model and did some further analysis. In Figure 14 we can see that the results were almost identical, albeit with a slightly better prediction going to the Decision Tree algorithm. At this point, we just decided to continue with one model in order to simplify the analysis process; thus, going forward the rest of the provided figures will primarily be for the Decision Tree algorithm due to its simplicity, training time, and explainability.

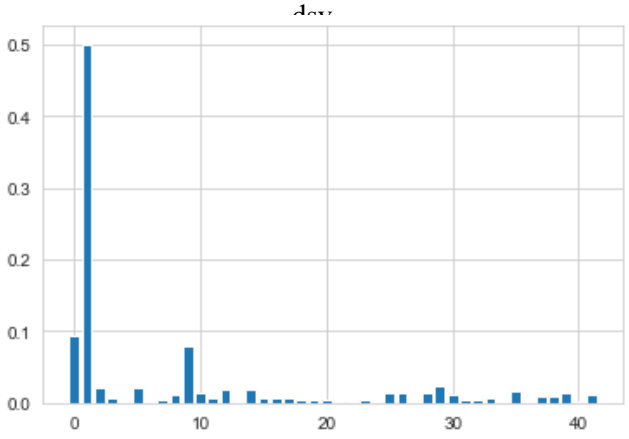

Figure 15: Model 7 Feature Importance Decision Tree

The next most important thing to tackle was to investigate why the model was predicting the way that it was. The best way to do this is to look at the feature importance, which shows what column of information was most influential in the decisions made by the algorithm. As we can see in Figure 15, the number of members/users was actually the most important feature, followed by episode count, and then a sprinkle of importance allocated to each of the genre types. This was not what we had initially expected, but nonetheless gave us a great insight into the data.

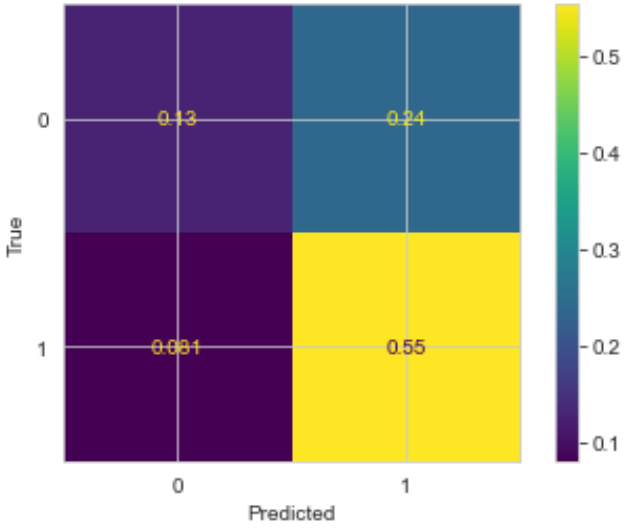

Figure 16: Model 8 Decision Tree Confusion Matrix

Now for this portion of the project we began to investigate the next best possible options of excellence that we could achieve. To achieve this, we incrementally increased from a value of 8 to a value of 10 in terms of the rating cut-off for the model. In Figure 16, we can see that there is a significant decrease in the prediction capabilities of the model. With a recall value of 0.68, precision value of 0.67, and an F1-Score of 0.65, this model performs slightly below the standards that would consider this an intelligent system.

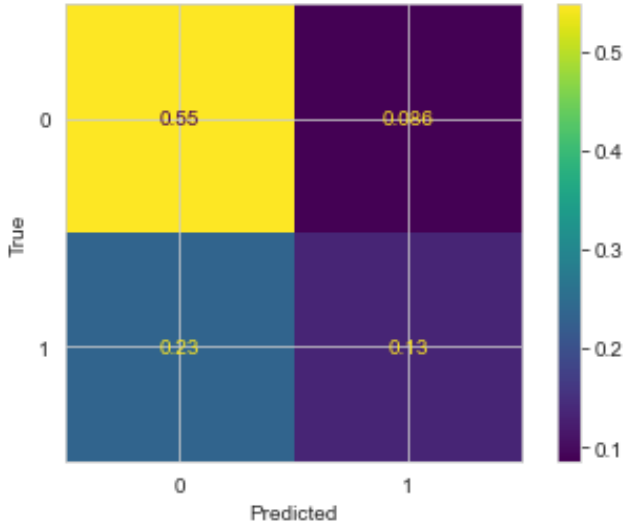

Figure 17: Model 9 Decision Tree Confusion Matrix

Moving onto the next increment with a value of 9 for the rating cut-off, we were greeted with a similar outcome, except distributed slightly differently, as shown in Figure 17. The recall value was 0.68, precision value was 0.67, and the F1-Score was 0.66. It performed slightly worse overall, but mainly the real difference is that this model trended towards identifying shows that would not fit the user, and hence shows that should not be recommended, which was not a major issue, but when we are trying to predict a good match, having a recall value of 0.36 for positive identification is considered to be very poor.

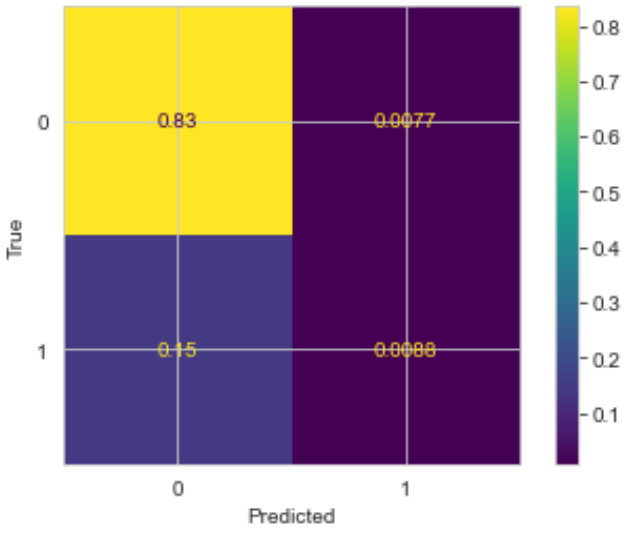

Figure 18: Model 10 Decision Tree Confusion Matrix

Finally, we examined the highest possible score of 10 as the bare minimum rating value, which would truly would be the best outcome for our recommendation system, since it would be able to provide an overall better recommendation and user experience. However, upon examining the model shown in Figure 18, we concluded that there was not enough data to support the high score threshold for the model to pick up on any trends. At first glance it would look great, because the recall value is 0.84, the precision value is 0.80, and it had an F1-Score of 0.79. If you investigate the details of where the model is getting this accuracy from though, you will see that the model is mostly blindly picking the negative matches because the data set is skewed towards lower ratings. This is what we call a lazy or naive based prediction, in which the model just goes with the most common answer because it is likely to be correct (but it doesn't actually guarantee correctness). We can verify this with the fact that it only has a recall value of 0.52 on the positive predictions.

APPLICATIONS

As seen from our recommendation algorithms and models, we were able to develop fairly appropriate recommendation systems to generate interactive anime television series recommendations for users based on the provided existing user data from the data sets. Although our predictions/systems were not perfect, we were still able to achieve a desirable level of "success" for our project and models. The knowledge gained from the development of these models and algorithms can be applied to future data mining projects involving recommendation systems, and even towards the pursuit of further enhancing our current systems, since there is arguably more analysis and investigation that could be performed past the current scope of our project.

Overall, this project contained a lot of great data insights that someone would not know if they just looked at the columnar data by themselves, such as the nature of the relationship between certain attributes such as users and user ratings. Especially when you put the models which we created to work, the data shined in ways that were not clear to the naked eye. We hope that this project can lead to even more exploration with the idea and maybe even eventually become a fully fledged integration in a recommendation platform.

REFERENCES

[1] https://myanimelist.net/about.php
[2] https://www.crunchyroll.com
[3]https://myanimelist.net/recommendations.php?s=recentrecs&t=anime
[4] https://www.kaggle.com/CooperUnion/anime-recommendations-database