Game of Thrones Predictive Model

Audrey Crockett | Jagannathan Govindan | Zinia Kamal | Rich Schiavi

Executive Summary

Game of Thrones (GoT) is a fantasy TV-series that premiered on HBO in 2011; it is an adaption of the book series written by George R.R Martin. Due to the popularity and continued success of Game of Thrones, this predictive model will attempt to predict the demographic user ratings of GoT episode based on the characters and their associated screen time in the episode.

Data Acquisition

To build the dataset and create inputs for the predictive model, we used IMBD to gather rating information for each episode for overall rating and demographic. The first step in the process was to scrap the information from IMBD using Python; data scraping allowed us to collect information from IMBD for ratings per episode. Once ratings were collected then we had to merge it with the episode data. Next challenge was collecting all the screen times for each character. Luckily we found multiple fans who had previously documented all the screen times but finding the appropriate data to import into R was the challenge (see apprendix for github link for code).

Exploratory Data Analysis

We started by examining the stats of each character using the data pulled from IMBD. The first instinct was to explore and visualize the data to identify any trends by creating different views of the data; we created scatter plots, heat maps, bar graphs among others which can be viewed below.

The first trend we noticed was that as Game of Thrones progressed as a series, a valley in the curve around the middle of season 2, a large valley during season 5, and climbing ratings since the middle of Season 5 (as seen in figure 1).

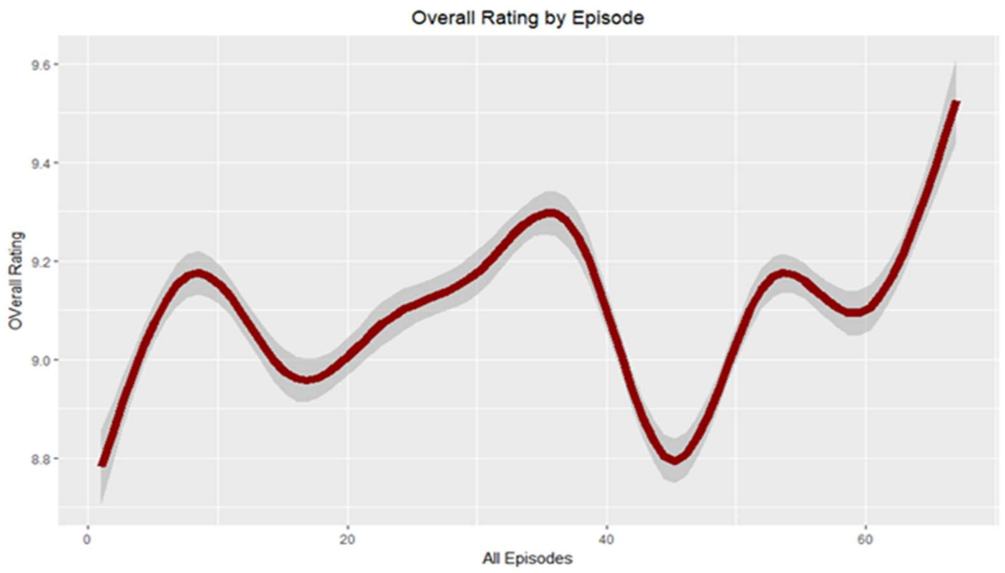


Figure 1

The next trend, in Figure 2, we noticed is episodes later in the season tended to get better ratings which makes sense since as the season progresses more and more watchers view the show due to popularity of the show.

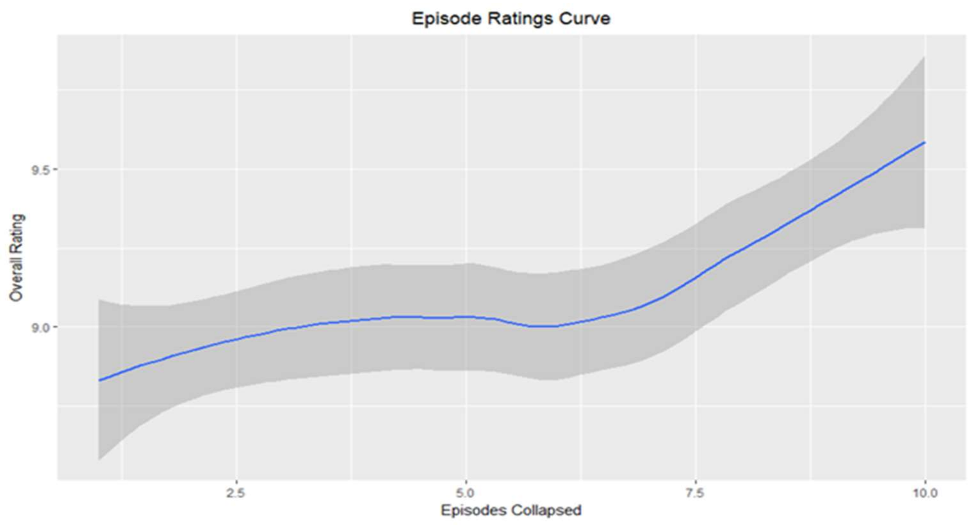


Figure 2

Figure 3 is another way of visualizing rating trend over time. The scatter plot dot size reflects the number of people who voted on the episode rating and the colors represent the seasons.

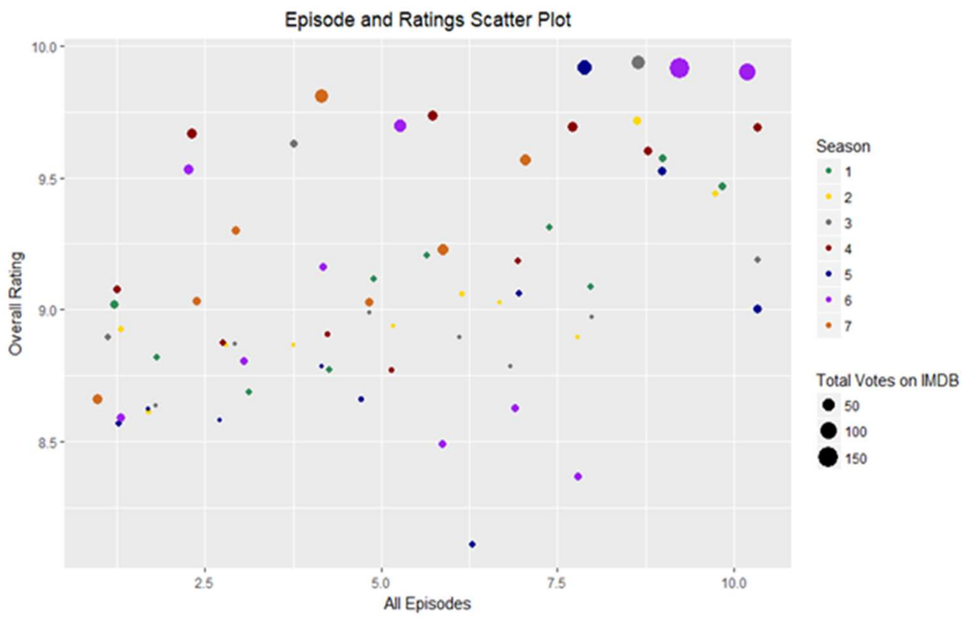


Figure 3

Figure 4 shows episodes with total votes and ratings. In this bar graph, we can see the influence of total votes on overall rating. “Battle of the Bastards” has the highest rating and the highest number of total votes. As many Game of Thrones fans confirm that this episode was one of the most epic episodes.

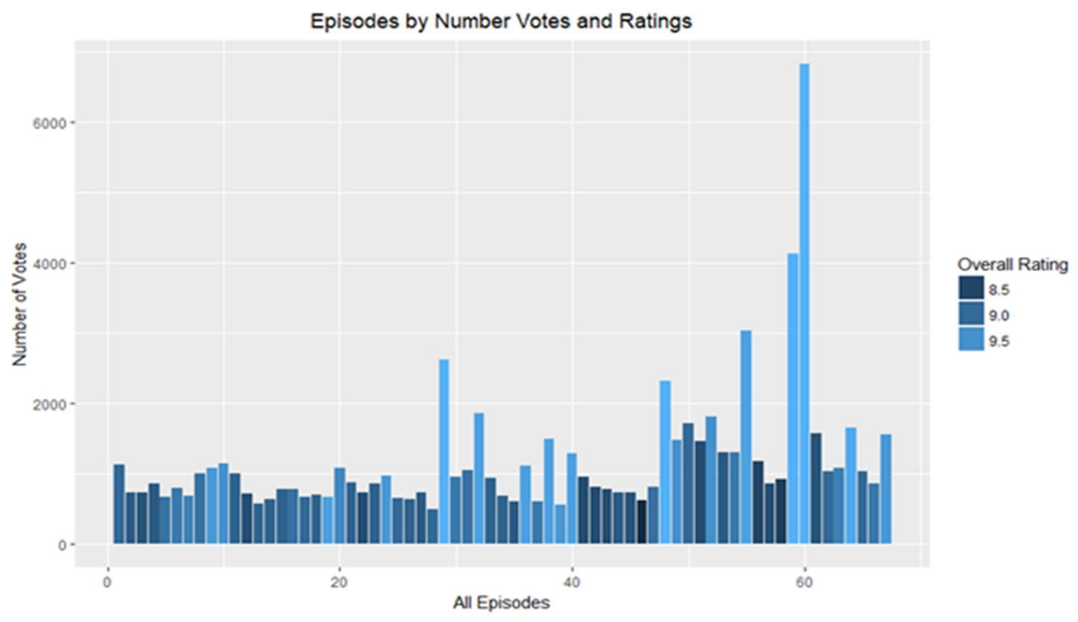


Figure 4

Something that was difficult to display with our data was characters. Our data included over 550 characters. We discussed many ways of creating a character cut-off for visualizations. Before we came up with a character cut-off we want to visualize a few characters with the episode ratings over time. Figure 5 is an example of one of these plots. Here, we have Arya Stark and the episodes that she appears in with the overall rating of that episode.

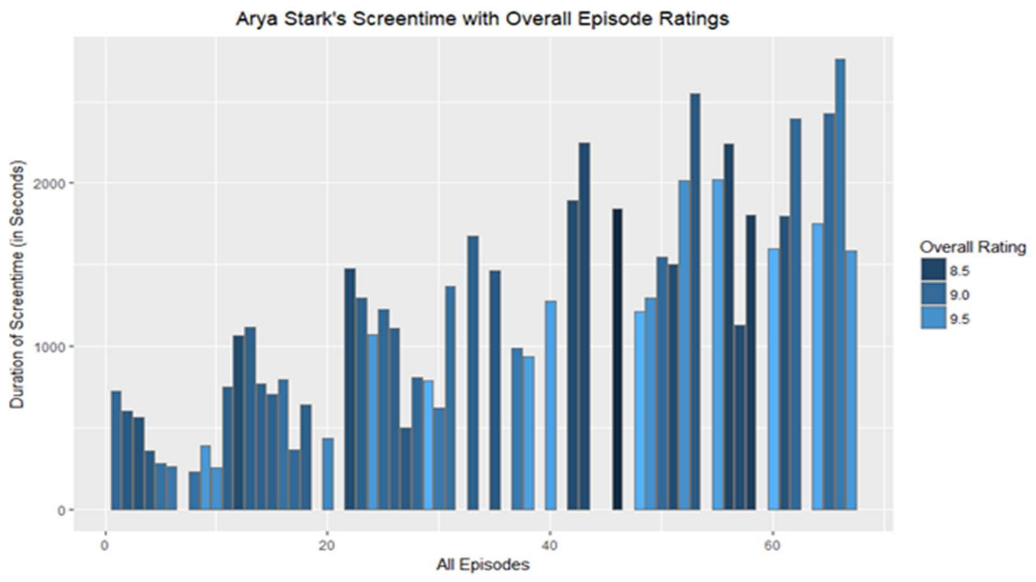


Figure 5

As we mentioned before, the difficulty of identifying cut-offs of characters for our visualization. In our next visualization, we identified our cutoff as total screen time greater than 10,000 seconds. Using this we still account for character who died early, like Ned Stark who dies in season 1. Let's examine the heatmap shown below further in Figure 6, the most obvious trend is the top few characters have a consistent high rating. However, we also noticed variations in the other stats, so we decided to add weights across the rating to normalize the data.

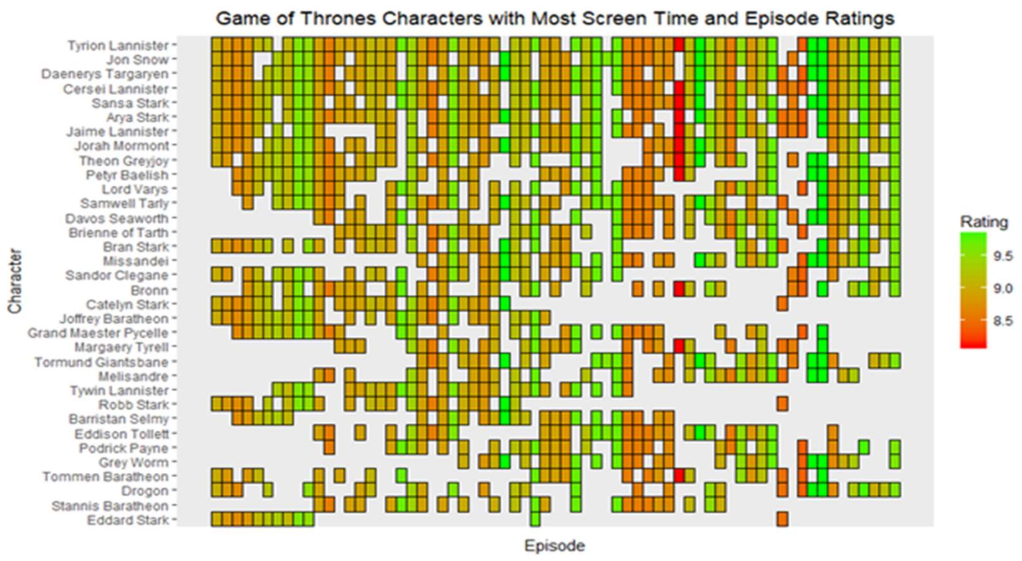


Figure 6

Figure 7 is a visualization of our weighted rating system. The characters included in the visualization are 35 highest rated characters according to our rating system. The top 3 characters according to our rating system are Tyrion Lannister, Jon Snow, and Daenarys Targayren. As fans of the show, we agree with accuracy of these top 3 characters listed, we believe that these characters make sense for the popular characters of the series with room for bias.

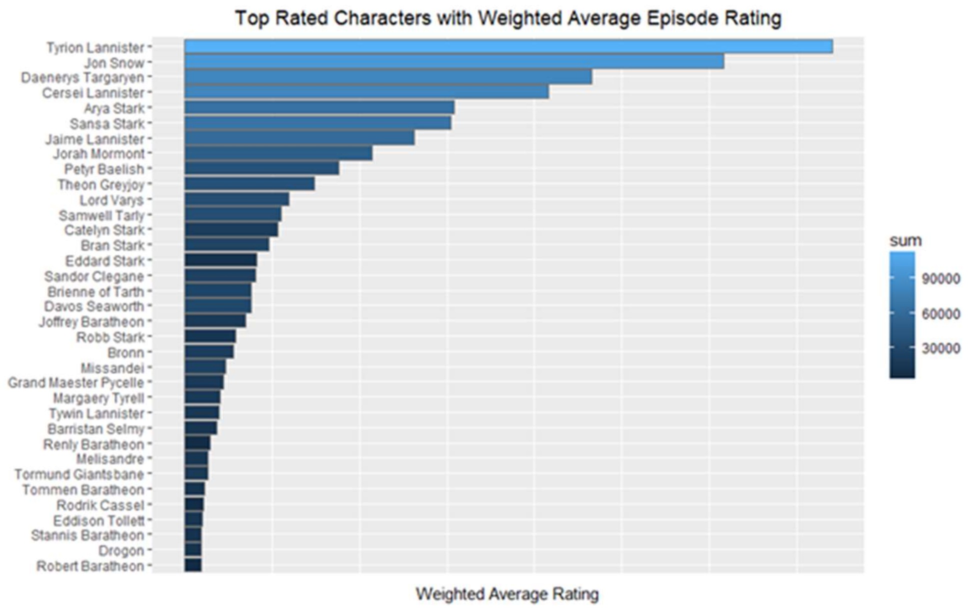


Figure 7

Predictive Model

Random Forest Model

In our project, we decided to use machine learning techniques, we constructed predictive models to determine screen time based on rating. The model-free methods we employed included random forests and neural network. Model-free method adapt to the intrinsic data characteristics with fewer assumptions.

First random forest shown in Figure 8, we used variables screen time, overall rating, weighted rating (per character and episode), character (in the form of a character id number). The model accounts for 83.3% of the change in overall rating.

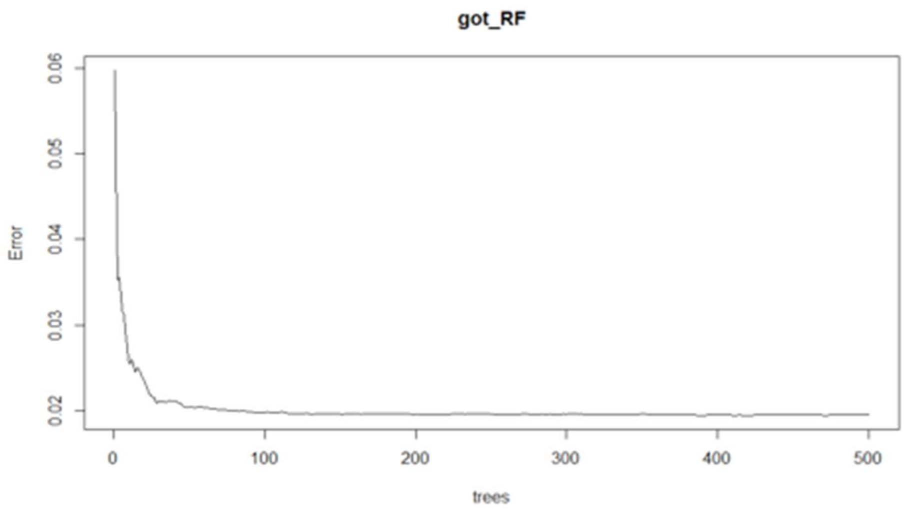


Figure 8

As a conclusion from our first random Forest model, the total votes was the most influential factor on our model. First concern we had about results were that model was not very accurate and it was predicting values that fell between 8.7 and 9.5. Second concern was that we needed to know total votes and weight rating ahead of time, which is not possible.

Second attempt at random forest shown in Figure 9 using just screen time and characters to predict rating. We started off poorly, only 2% of the variance in overall rating was explained by overall ratings, which means any results we found have 97.32% chance of being the result of randomness or an error.

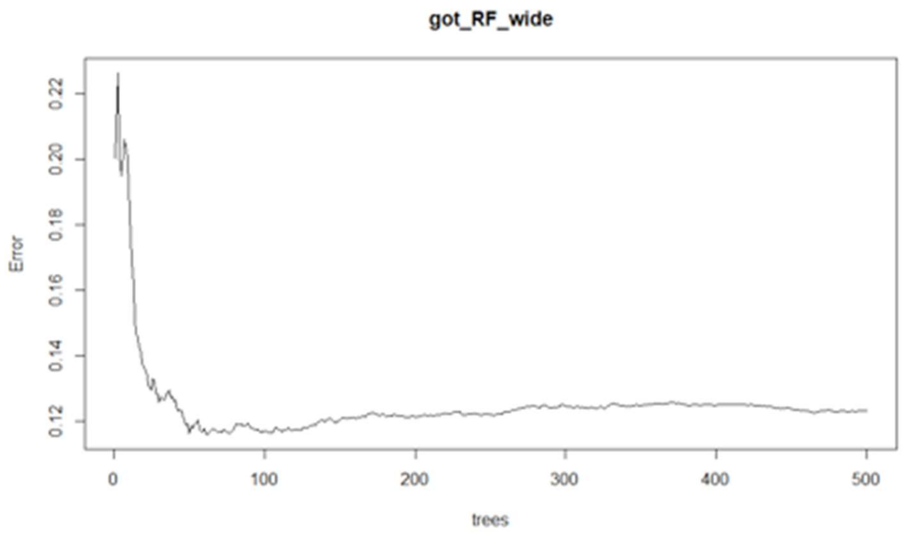
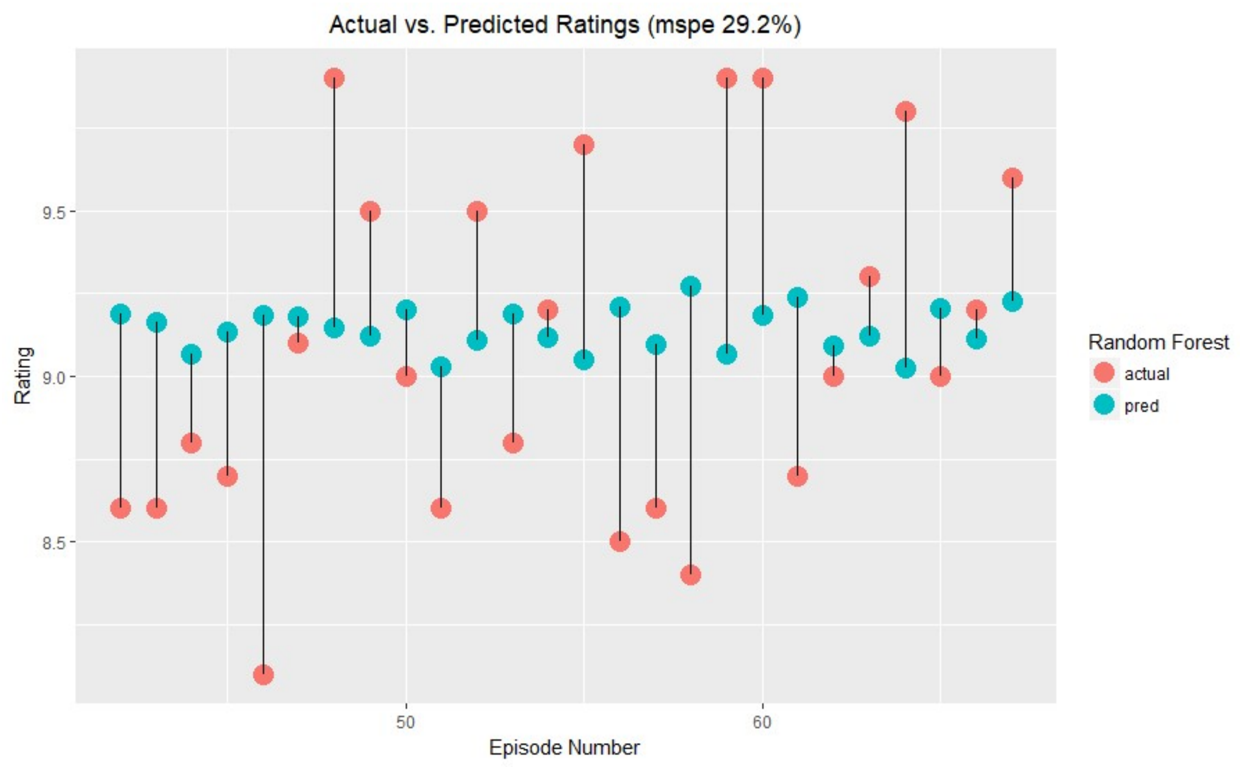


Figure 9

Error rate is up which one can see from the first error plot to the second plotted random forest prediction model, it is decently a large error rate. Model prediction tends to above 9 but below 9.5. On the plus side, with the predict function one can enter a character and screen time amount and get a rating. However, the rating is not accurate and usually falls between 9 and 9.5.



Neural Network

Once we concluded our random forest model was prone to error, we decide to try a neural network model shown in Figure 10. We identified our input variables as characters and screen time and our output variable as screen time. We used 10 hidden nodes for our neural network. A hidden node is a layer in the neural network used to find a nonlinear pattern in the data. The more hidden nodes the higher the likelihood of identifying a pattern. We chose 10 hidden nodes, instead of say 50, for optimization purposes. The time it would take to run 50 hidden nodes does not outweigh the benefit in the predictive model (i.e. 50 hidden nodes over 10 would only increase the model's ability to predict by a miniscule amount).

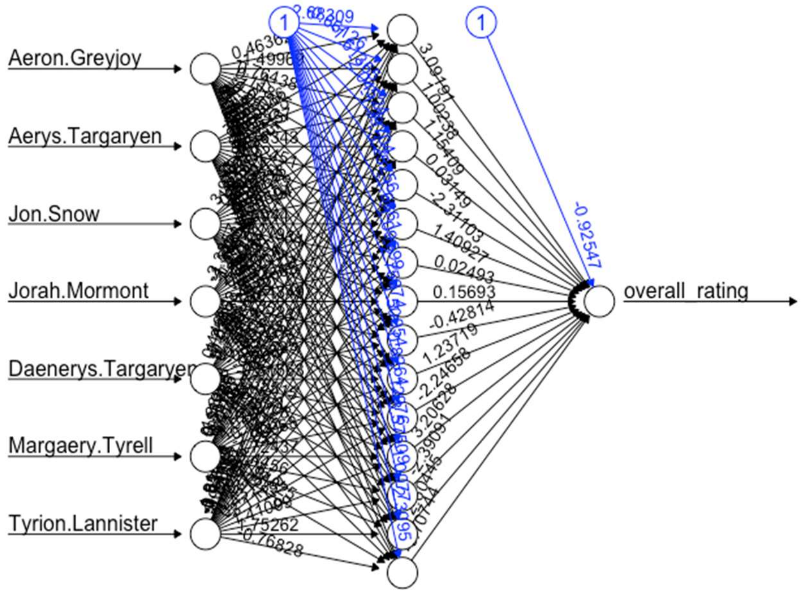


Figure 10

The first time we ran the neural network, we did not normalize the ratings. This led to high rate of error. The next time, we normalized the overall ratings from 0 to 1. Also, in this neural network set up, we specified that the neural network use back propagation.

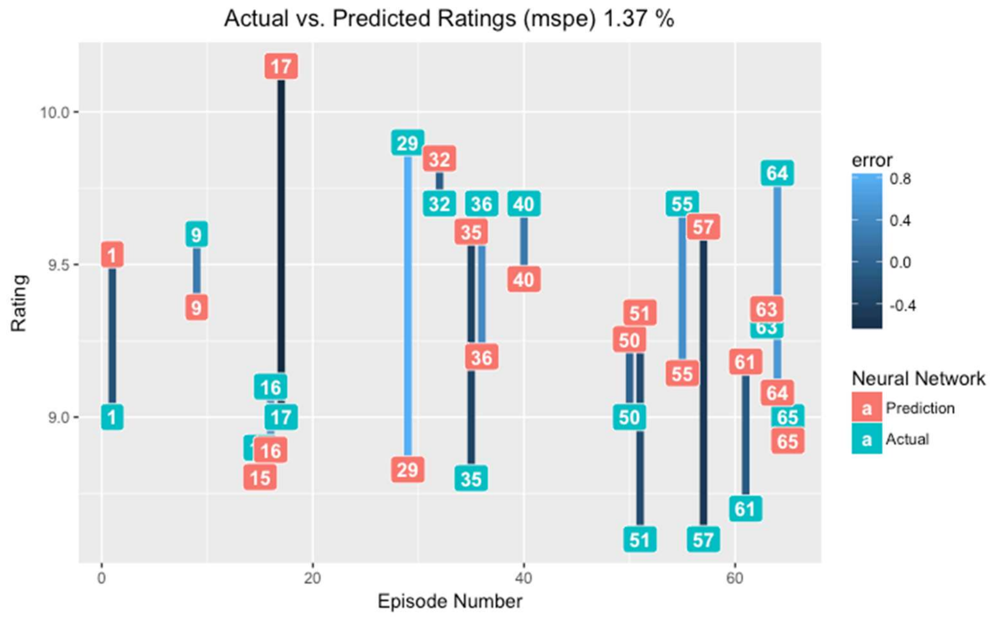


Figure 11

Conclusion

Overall, we realized that project was very ambitious especially in this short period time, however we believe that a prediction can be made based on what characters are in an episode and their screen time. While we had a great deal of characters and screen times, there were only 67 episodes and more episodes were required to enhance the models abilities to make a prediction.

Appendix

Github links:

- https://github.com/rschiavi/got_ist687
- https://github.com/amcrockett/got_random_forest

Shinyapp

- https://aloha.shinyapps.io/Got687/