**The Data Ludologists**

**Board Game Recommender**

**Capstone Project (4), Group 3**

Abi Chambers

Van Eisele

Alejandro Perez

A person in a picture frame

Description automatically generated**Introduction**

"I never go easy on kids when I play board games. The sooner they learn what the consequence of entering a competition is, the better. If they win, I punch them in the face like any adult." -Zach Braff

Okay, we would never seriously promote punching children. Quite the opposite, even! Board games, in all their endless variety, are not only a fun way to past the time but can be beneficial to child and general social development. These benefits include, but are not limited to, promoting social interaction, encouraging communication and critical thinking development, and strengthening math and problem-solving skills1.

We chose to do our capstone machine learning project on board game data as the result of sincere interest, as well as the accessibility of the data. Coming from Kaggle, one of the datasets we used had been used before in several other prediction models2, and the other was derived from a well-known online board game resource3.

**Data Information and Cleaning**

As mentioned, this project was based on two CSV files from the Kaggle website’s publicly available datasets, one titled ‘Board Games Prediction Data’2, and the other titled ‘Top 5000 Board Games at BGG’3. The Prediction dataset consisted of 20 columns, and around 80,000 rows; though we would discover a duplicate of every row, so that cut that data in half. The BGG dataset was, as named, 5,000 rows of data, by 25 columns. We did our initial data cleaning in Jupyter Notebook. We read in both of our CSVs, and immediately dropped all the duplicate values in the Prediction dataset. We then started renaming those columns in the Top 5000 dataset that corresponded to those in the Prediction dataset. In reviewing the data as it stood at that point, we realized we had several columns (‘mechanic’, ‘category’, and ‘designer’) that contained multiple entries per cell, so we proceeded to split those on the commas, using Python’s ‘lambda’ function. Additional inspection, and help from Teacher’s Assistant, Sherhone Grant, prompted us to use a regex function to clean the ‘name’ column of special characters and diacritics. Once we added a clean name column, we left-merged our two datasets, making the Top 5000 dataset our primary dataframe, and joining in all applicable data from the Prediction dataset. Then we identified columns in the Prediction dataset that were not in the Top 5000 dataset, created a sub-dataframe from just those, and re-left-merged to add those columns of data back in to a clean dataframe. We saved that clean data off to a CSV and were on our way to Tableau analysis and visualizations, and machine learning.

**Research Questions & Tableau**

Upon importing our clean and complete CSV into Tableau, it was realized that for effective and compelling visuals, as well as for the sake of functionality, several columns would be more useful binned, to reduce unique values. These included the ‘category’ column, which we elected to rename ‘genre’ to avoid confusion with our binning efforts, as well as the ‘mechanics’ column, and our ‘publisher’ column. The ‘genre’ and ‘mechanics’ data were both meticulously grouped by related qualities using our team’s domain expertise, while the ‘publisher’ data was separated into alpha-based groups. To be able to apply these groupings in Tableau without losing any data integrity, we created alphabetical groups for our ‘publisher’ column…

A screenshot of a computer

Description automatically generated

… and for the genres and mechanics, to allow for the extensive crossovers per board game, we used calculated fields with extensive IF-THEN statements.

A screenshot of a computer

Description automatically generated

We were now ready to craft our two dashboards, which we would then embed in our web app. The first was a summary of meta-data, where we were able to get a look at board games by the numbers, i.e., how many board games by each publisher, what is the most common mechanic amongst board games (which turns out to be dice rolling), and what genre is the most popular in board games (card games have the highest counts). This dashboard also allows a user to utilize any or all of five filters: Average Rating, Popularity by Number of Ratings, Mechanic, Genre, and Publisher. Selecting a filter, or a combination of filters will show a user the number of board games fitting the selected criteria, as well as the number of crossovers between the various characteristics. The second dashboard was a more analytical look at our board game data. We took a look at board game complexity (referenced in our data as ‘weight’) as it related to the various mechanics, as well as complexity as it related to the average length of playing time of board games. Our hypothesis regarding the latter was that we would find a strong positive correlation, and while the data proved a positive correlation, it was not as strong as we had suspected. This dashboard also illustrated the top 100 games in our data by their BGG ranking, and compared that to the game’s general popularity, as indicated by the number of ratings it had received. Those results were decidedly surprising, as the two pieces of information had virtually no correlation at all! This dashboard was also responsive to a set of filters: Average Rating, Popularity by Number of Ratings, and Genre. Interestingly, the most frequently rated games do not occupy any of the very top spots in ranking.

A screenshot of a computer

Description automatically generated

**KNN Recommender**

With the data cleaned and merged we were almost ready to begin machine learning with just a little data processing to do. The first thing we had to do was to decide which columns/features that were going to be important to use to for our knn recommender. Listed below are the features we decided to use.

Features:

• Rank – we are using the top 5000 boardgames.

• Id – to give us an index number for the board games.

• Min-players – to help people pick a game they would have enough players to play.

• Max-players – to help people pick a game that could include the number of players they had.

• Average rating – to let players know the average rating that users or critics gave a game to help in the decision-making process.

• Age – so people could pick a game that was acceptable for everyone.

• Total owners – to help emphasis this is a good board game because x number of people were willing to purchase it.

• Total weights – to give the players knowledge of how complex a game was going to be.

• Mechanic count – number of mechanics a board game implemented.

• Mechanic – the style of game play that the board game utilized.

• Category count – how many different categories this game fits into.

• Category – the main category per game.

• Designer count – in case people were interested in how many designers a game had.

• Name – the name of the board game.

After we picked our features, we took a closer look at the data to see if we need to/could bin any items. We discovered we had 52 unique mechanic values and 83 unique categories. After some debate we were able to bin the mechanic and category features down to 6 mechanic and 6 categories.

After binning we need to drop unnecessary columns. The next step was to look at the data again, we noticed there were 22 null values in the name column so we dropped or null values, 22 games out of 5000 games means we dropped .004% of our data.

We had to encode mechanic column and category column to make everything a numeric value.

Once all the data had been processed, we printed out the data frame to a csv file to be used with the website.

Now it was time to create the recommender, starting with selecting our features for comparison and adding creating a data frame for the recommender model.

A computer screen shot of a computer program

Description automatically generated

Once that was done it was time to create the recommender function:

A screenshot of a computer code

Description automatically generated

Now to test the model, when we initially ran the model, we included rating and the number of owners. However, we would come to find that having these two features in the comparison was throwing the model off, when we asked for recommendations for games like Catan we got a bunch of games that didn’t really fit, like Codenames, which is nothing like Catan. We discovered that the rating and owners were doing two things to our model as a feature these columns had high values which was throwing off the matching and it added a bias to the data for popular games. While Catan and Codenames are nothing alike they were equally popular and rated.

Armed with this discovery we removed ratings and owner from the features to be compared to information that would be given once a board game was given to help the user to make a decision.

Test number 2, gave back better results:

A screenshot of a computer

Description automatically generated

With some more Catan versions populating on our table. However, we thought we could do better and decided to add a scale to our data because a few of our columns had much larger numbers like max play time. With the scalar in place, we got much better results with more version of Catan filling in the table which makes sense:

A screenshot of a computer

Description automatically generated

Also, with our scalar implemented it shrank our distances, which is the main way the KNN recommender decides what to recommend. The closer in distance an output is to the selected game the more similar the game is to the inputted game:

A close up of numbers

Description automatically generated

And as you can see all these distances are below 1, which is great. We utilized Euclidean distance in our model because it is a good type of measure that had the best all-around use. Some of the other options we talked about were: Minkowsky (which is very similar to Euclidean) and Chi Square (which is used more in some of the medical recommendation models we looked at but would sometimes out preform the Euclidean and Minkowsky measurements for distance. Since we weren’t using any locations in our data that relate to locations on earth, we did not need to use the great circle distance. There is also a cosine similarity measure but based on research that didn’t hold up as well as the Euclidean distance.

With our KNN Recommender up and running it was now time to get everything ready for the website.

**Full-Stack Design**

The website was designed in hopes of having different pages showing very different information, and yet making the pages feel like the same website. The main style of the site was coded the same on each page. The same navigation bar and title is uniform across the site. The biggest hurdle in designing the site was the syntax of the code and its effect on the type of display features. There were a lot of brute force changes to get individual desired changes, but with each page that was designed it started to become more familiar. Using a Bootswatch Flatly theme, the initial code was a good starting point to get a feature/design implemented.

A group of dice on a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

The pages showing our Tableau designs were the easiest to design. The ability to embed a Tableau view with one line of code (copied from Tableau Public site), made the creation of the pages seamless.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The modelHelper Python notebook was the biggest hurdle. Since our team decided on designing a board game recommender, we had to learn how to adapt code for our unique machine learning model. Connecting the python notebook files (app.py and modelHelper.py) to each other and to our html page required course resources to fill in the gaps or our coding. With our code working the last step was to make sure that our recommender inputs were clear enough for our front-end users to understand.

A screen shot of a computer program

Description automatically generated

The largest hurdles for our team were getting the website code to read our data csv. This required us to make a design decision of only having three user inputs. This minimized the amount of parsing needed in our Post request. Another challenge was the design idea versus the ability to html code accordingly. This was a lot of trial and error, as we learned while making small individual changes.

A screen shot of a computer program

Description automatically generatedA screen shot of a computer screen

Description automatically generated

Upon completion and looking ahead, it would be nice to eventually expand on the number of user inputs. This will allow a user to get a board game recommended to them that they will likely play. Another takeaway would be to utilize a style.css file to streamline the main design across the site/webpages.

**Limitations & Bias**

Limitations included the following:

* The BGG data was current only through 2019.
* Our other data source remains unverified in terms of origin.
* Achieving the desired website recommender formatting/structure in our coding.

In terms of data bias, our machine learning efforts found bias in the average rating and number of owners data.

**Future Work**

* The team very much would have liked to make our app searchable by various game characteristics, like maximum play time, age appropriateness, and both minimum and maximum number of players, to return recommendations.
* We would also be interested in learning how to apply Bootstrap framework to make the web app mobile friendly.

**Works Cited**

1. https://www.theschoolrun.com/13-ways-playing-board-games-benefits-your-child

2. https://www.kaggle.com/datasets/centipede148/board-games-prediction-data

3. https://www.kaggle.com/datasets/mcdemarco/top-5000-board-games-at-bgg

4. https://www.w3schools.com/html/default.asp

5. https://bootswatch.com/flatly/

6. https://github.com/qazmklp/Project-4-Group-2

7. https://towardsdatascience.com/a-simple-approach-to-building-a-recommendation-system-d0f4de1a1f50

**Other Resources**

1. https://boardgamegeek.com

2. https://boardgamegeek.com/wiki/page/mechanism

3. https://boardgamegeek.com/wiki/page/Category