Linear Regression Write Up

Predicting Board Game Complexity

Derek Call

Abstract

There are many factors that go into creating board games. What kind of game is it? What are the game mechanics? How long should it take? How many people can play? What’s the best price? These are questions that board game manufacturers need to answer to try and be successful. Another key question to answer is who the target demographic is. Board game complexity scores are a user voted metric and can be an integral part of picking out that next game you’ve wanted to purchase. Using linear regression, I have created a model that will assist in making a board game with an estimated complexity score near what we predict users will give it.

Design

I gathered manufacturer provided details for board games, along with historical data and user voted metrics. After testing many different modeling methods, I found simple linear regression to be the most predictive model and then continued to see which features specifically predicted board game complexity the most.

Data

All data was scraped from <https://boardgamegeek.com/> . I gathered data for the top 3000 games by rating, gather 16 different features per game in all. After cleaning the data and removing null values I was able to start modeling with 1372 observations. The features gathered are a mix of manufacturer provided game information, professional ratings, user voted features and historical website data on views.

Algorithms

I used the Selenium package in Python to grab the html data from each individual game’s landing page. I then parsed the html data in Python using the BeautifulSoup package.

Feature Engineering:

1. Tested polynomial terms in the linear model using SciKitLearn’s PolynomialFeatures method. This did not improve the model score.
2. Manually input some polynomial terms based on SeaBorn pairplot of data and found the square root of ‘minutes’ was an appropriate inclusion that boosted the model score.
3. I honed my focus on features that were provided by the manufacturers and removed features that did not improve the model score.

Models

I split the data into training + validation data (80%) and test data (20%). I used a K-Fold validation method to test between linear, Ridge and polynomial regression models, using the training/validation data from above. After finding the mean R2 score for each model, for each K-Fold validation, with 100 random subsets of training and validation data, a simple linear regression model with a single polynomial term added for ‘minutes’ yielded the best score.

Findings

The training data yielded an adjusted R2 of 0.749, with the K-Fold cross validation yielding a score of 0.744, so the model works well to predict about 74% of the variance in game complexity scores. When I tested this model using the test data it had not seen the test R2 was 0.705. The main metric of interest though is the Mean Absolute Error (MAE) in this final model which is 0.807. This means that on average, the model prediction will be off by 0.807. For complexity scores between 1 and 3 the MAE is much lower as the variance of the errors is also lower.

The most import features and their coefficients in the final model are: Geek Rating: 0.1069\*\*\* Strategic: 0.3136\*\*\* Minutes: -0.0055\*\*\* Minutes (Sqrt): 0.2743\*\*\*

\*\*\*p < 0.01, \*\*p < 0.05, \*p < 0.1

Tools

* Selenium and BeautifulSoup Python packages for html retrieval and parsing
* Pandas, Numpy, StatsModel and SciKitLearn Python packages for data cleaning and modeling
* Matplotlib and Seaborn Python packages for data visualization

Figures

Chart, scatter chart

Description automatically generated

Chart, scatter chart

Description automatically generated