Matt Cronin

Econ 9000 – Machine Learning – PS1 – Board Game Geek Exercise

Data Description

I chose to scrape the data on game names, ratings, number of votes, and prices on boardgamegeek.com. In addition to this I scraped the games Board Game Geek webpage URL extension as well. This website contains a table of the game ratings and rankings at the following URL: https://boardgamegeek.com/browse/boardgame/page/1

The first 30 rows of my data frame are here:

	avg_rating	file_name	geek_rating	link	...	key	price_type	game_index	price
0	8.91	html_files\boardgame_1.html	8.613	/boardgame/174430/gloomhaven	...	1	List	0	140.00
1	8.91	html_files\boardgame_1.html	8.613	/boardgame/174430/gloomhaven	...	1	L_Amazon	0	87.81
2	8.91	html_files\boardgame_1.html	8.613	/boardgame/174430/gloomhaven	...	1	N_Amazon	0	104.99
3	8.91	html_files\boardgame_1.html	8.613	/boardgame/174430/gloomhaven	...	1	App	0	NaN
4	8.91	html_files\boardgame_1.html	8.613	/boardgame/174430/gloomhaven	...	1	Used	0	NaN
5	8.91	html_files\boardgame_1.html	8.613	/boardgame/174430/gloomhaven	...	1	Amazon	0	NaN
6	8.65	html_files\boardgame_1.html	8.492	/boardgame/161936/pandemic-legacy-season-1	...	1	List	1	69.99
7	8.65	html_files\boardgame_1.html	8.492	/boardgame/161936/pandemic-legacy-season-1	...	1	L_Amazon	1	NaN
8	8.65	html_files\boardgame_1.html	8.492	/boardgame/161936/pandemic-legacy-season-1	...	1	N_Amazon	1	NaN
9	8.65	html_files\boardgame_1.html	8.492	/boardgame/161936/pandemic-legacy-season-1	...	1	App	1	NaN
10	8.65	html_files\boardgame_1.html	8.492	/boardgame/161936/pandemic-legacy-season-1	...	1	Used	1	NaN
11	8.65	html_files\boardgame_1.html	8.492	/boardgame/161936/pandemic-legacy-season-1	...	1	Amazon	1	NaN
12	8.54	html_files\boardgame_1.html	8.263	/boardgame/182028/through-ages-new-story-civil...	...	1	List	2	69.95
13	8.54	html_files\boardgame_1.html	8.263	/boardgame/182028/through-ages-new-story-civil...	...	1	L_Amazon	2	NaN
14	8.54	html_files\boardgame_1.html	8.263	/boardgame/182028/through-ages-new-story-civil...	...	1	N_Amazon	2	63.26
15	8.54	html_files\boardgame_1.html	8.263	/boardgame/182028/through-ages-new-story-civil...	...	1	App	2	9.99
16	8.54	html_files\boardgame_1.html	8.263	/boardgame/182028/through-ages-new-story-civil...	...	1	Used	2	NaN
17	8.54	html_files\boardgame_1.html	8.263	/boardgame/182028/through-ages-new-story-civil...	...	1	Amazon	2	NaN
18	8.40	html_files\boardgame_1.html	8.237	/boardgame/167791/terraforming-mars	...	1	List	3	69.95
19	8.40	html_files\boardgame_1.html	8.237	/boardgame/167791/terraforming-mars	...	1	L_Amazon	3	51.12
20	8.40	html_files\boardgame_1.html	8.237	/boardgame/167791/terraforming-mars	...	1	N_Amazon	3	54.97
21	8.40	html_files\boardgame_1.html	8.237	/boardgame/167791/terraforming-mars	...	1	App	3	NaN
22	8.40	html_files\boardgame_1.html	8.237	/boardgame/167791/terraforming-mars	...	1	Used	3	NaN
23	8.40	html_files\boardgame_1.html	8.237	/boardgame/167791/terraforming-mars	...	1	Amazon	3	NaN
24	8.33	html_files\boardgame_1.html	8.176	/boardgame/12333/twilight-struggle	...	1	List	4	64.99
25	8.33	html_files\boardgame_1.html	8.176	/boardgame/12333/twilight-struggle	...	1	L_Amazon	4	31.42
26	8.33	html_files\boardgame_1.html	8.176	/boardgame/12333/twilight-struggle	...	1	N_Amazon	4	32.97
27	8.33	html_files\boardgame_1.html	8.176	/boardgame/12333/twilight-struggle	...	1	App	4	6.99
28	8.33	html_files\boardgame_1.html	8.176	/boardgame/12333/twilight-struggle	...	1	Used	4	NaN
29	8.33	html_files\boardgame_1.html	8.176	/boardgame/12333/twilight-struggle	...	1	Amazon	4	NaN

The dataset's numeric values are described below:

	avg_rating	geek_rating	rank	votes	year	key	game_index	price
count	471306.000000	136308.000000	101238.000000	471306.000000	580356.000000	634506.0	634506.000000	11835.000000
mean	6.119505	5.714732	8437.000000	192.704740	1999.786448	1.0	52875.000000	42.049641
std	1.518667	0.373110	4870.839594	1458.192927	67.874621	0.0	30527.708213	68.386091
min	1.000000	3.463000	1.000000	1.000000	-3500.000000	1.0	0.000000	0.000000
25%	5.250000	5.524000	4219.000000	2.000000	1996.000000	1.0	26437.000000	16.490000
50%	6.250000	5.575000	8437.000000	8.000000	2009.000000	1.0	52875.000000	29.950000
75%	7.110000	5.732000	12655.000000	40.000000	2015.000000	1.0	79313.000000	49.975000
max	10.000000	8.613000	16873.000000	83832.000000	2022.000000	1.0	105750.000000	5538.640000

Supervised Learning Exercises

I performed two supervised learning exercises. They were missing price data imputations and board game geek rating predictions.

Missing Price Data Imputations:

During my data scraping exercise, I noticed that not all games had every price type listed. For example, a game could have a list price, but no Amazon prices, or a game could have a new amazon price listed, but no lowest amazon price and no list price. Looking at the first 6 rows of my data frame in the screenshot above, shows that the game "Gloomhaven" has list, new Amazon, lowest amazon, but no iOS app, used, or general amazon price (obviously there could be no app version of this game). Going back to board game geek I noticed that the price data they report for some games changes. At some points in time the website will report different price data for the games

and new price data becomes available. A researcher looking to perform some analysis on boardgames could go back to the website and re-scrape the data with the hopes that they are lucky and get a more complete draw of prices at that time they decide to re-scrape. However, that process is time consuming and the outcome is uncertain in that your new set of available prices may not be a larger set than the prices you have already pulled. In addition to this you could re-scrape scrape prices when only a subset of the price data, that you already have, are available. Because only some price observations were missing for certain games, I wanted to perform a supervised learning exercise to impute the missing price data. More specifically I would use the sklearn linear model to estimate 6 regressions and predict each of the 6 price types for each game where that price type price was missing.

To be specific the regression equation I estimated, for a price type *i*, was as follows:

$$Price_i = B_0 + B_1 AvgRating_i + B_2 GeekRating_i + B_3 Rank_i + B_4 GameAge_i + B_5 VoteCount + \varepsilon_i$$

This specification assumes that given a certain average rating, geek rating, rank, and game age, a price type *i* can be predicted. After the linear model is training by estimating the above equation using available data, prices can be predicted for each price type. The program **'4_boardgamegeek_linearmodel.py'** performs the estimation described above. The general process is as follows:

1) Subset the data frame to a price type *i*
2) Split the resulting data frame from step 1 into the following two data frames:
    a. **Training Data:** Rows with price of type *i* not missing
    b. **Target Data:** Rows with price of type *i* missing
3) Create a linear model machine using the linear regression function in the linear model library from sklearn
4) Use Training Data to perform supervised learning on the linear model machine
5) Use the trained model to predict price *i* for each game in Target Data using the available data on average rating, geek rating, rank, and game age in that data frame
6) Go back to step 1 and perform same process for next price *i* in list of price types

The R-Squared scores for each estimation were considerably low indicating that the predictions were not very accurate.

	Price Type Estimated	R-Squared Score
0	List	0.234215
1	L_Amazon	0.124476
2	N_Amazon	0.011540
3	App	0.140731
4	Amazon	0.373978

These results seem to suggest that game prices are largely determined by variables not observed in this exercise. An additional exercise could be to loop through the game links I had scraped and scrape additional game information on each game's page. Data on the game's themes, difficulty rating, and number of reported users of the game could be important variables used to explain price.

BGG Rating Prediction:

The second exercise also used the linear model function from sklearn. This supervised learning exercise was designed to train a machine to predict the geek rating for a game. According to boardgamegeek.com, "BoardGameGeek's ranking charts are ordered using the BGG Rating, which is based on the Average Rating, but with some alterations. To prevent games with relatively few votes climbing to the top of the BGG Ranks, artificial "dummy" votes are added to the User Ratings. These votes are currently thought to be 100 votes equal to the mid range of the voting scale: 5.5, but the actual algorithm is kept secret to avoid manipulation. The effect of adding these dummy votes is to pull BGG Ratings toward the mid range."

In order to predict the BGG rating (given a set rank, average rating, game age, and number of votes), I trained the linear model with the available data using the following equation:

$$GeekRating_i = B_0 + B_1 AvgRating_i + B_2 Rank_i + B_4 GameAge_i + B_5 VoteCount + \varepsilon_i$$

The program **'4_boardgamegeek_linearmodel_price.py'** uses this linear model to predict the Geek Rating as well as uses k-fold cross validation to score the machine. The process is as follows:

1) The dataset is shuffled and split into 4 groups
2) Group *i* is selected as a test data set while the remaining 3 groups are used to train the model
3) The trained model then predicts the geek rating on group *i*, and an R-squared score is computed
4) Repeat the process three more times using the remaining 3 groups as test datasets.

The R-squared for each split is reported below:

	R-Squared Score	Split
0	0.705858	1.0
1	0.667255	2.0
2	0.729234	3.0
3	0.709359	4.0

This suggests that the model can explain about 66-70% of the variation in Geek Rating. Similar to the issues stated with the game price model above, data on the game's themes, difficulty rating, and number of reported users of the game could be important variables used to explain price.