1. Introduction

In this project, a dataset of about 82000 rows and 20 columns was given regarding the reviews of various board games available. A supervised machine learning model as then been trained and evaluated to predict the ratings given to a particular board game based on several features of the datasets.

In order to choose the machine learning algorithms best suited to make predictions of the average rating of the board games, number of regression or similar models were considered and the data has been investigated to choose the best possible models:

Considering that the problem is to predict a numerical value several options such as follows could be chosen:

i. Linear Regression:

Can be chosen if the variables show a linear correlation with the label. If the correlation is not linear, the linear regression model would not be accurate.

ii. Decision Trees and Random forest regression:

A Decision Tree is an intuitive model where by one traverses down the branches of the tree and selects the next branch to go down based on a decision at a node.While building the tree, the goal is to split on the attributes which create the purest child nodes possible, which would keep to a minimum the number of splits that would need to be made in order to classify all instances in our dataset. Purity is measured by the concept of information gain, which relates to how much would need to be known about a previously-unseen instance in order for it to be properly classified. Random Forests are simply an ensemble of decision trees. The input vector is run through multiple decision trees. For regression, the output value of all the trees is averaged; for classification a voting scheme is used to determine the final class. Great at learning complex, highly non-linear relationships.Very easy to interpret and understand. can be prone to major overfitting.Using larger random forest ensembles to achieve higher performance comes with the drawbacks of being slower and requiring more memory.

iii. Neural Network regression:

A Neural Network consists of an interconnected group of nodes called neurons. The input feature variables from the data are passed to these neurons as a multi-variable linear combination, where the values multiplied by each feature variable are known as weights. A non-linearity is then applied to this linear combination which gives the neural network the ability to model complex non-linear relationships. A neural network can have multiple layers where the output of one layer is passed to the next one in the same way. At the output, there is generally no non-linearity applied. Neural Networks are trained using Stochastic Gradient Descent (SGD) and the backpropagation algorithm. They are very effective for data with complex non-linear relationships with negligible consideration to the structure of the data. However, these models could be difficult to interpret and computationally challenging.

In [1]:

```
import sys
import pandas
import sklearn
import matplotlib
import seaborn

print(sys.version)
print(pandas.__version__)
print(matplotlib.__version__)
print(seaborn.__version__)
print(sklearn.__version__)
```

```
3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bit
(AMD64)]
0.23.0
2.2.2
0.8.1
0.19.1
```

In [2]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

In [3]:

```
#Loading the data
game = pandas.read_csv("games.csv")
# Observing the shape, columns and some rows of the dataframe
print(game.shape)
print(game.columns)
game.head
```

```
(81312, 20)
Index(['id', 'type', 'name', 'yearpublished', 'minplayers', 'maxplayers',
       'playingtime', 'minplaytime', 'maxplaytime', 'minage', 'users_rate
d',
       'average_rating', 'bayes_average_rating', 'total_owners',
       'total_traders', 'total_wanters', 'total_wishers', 'total_comment
s',
       'total_weights', 'average_weight'],
      dtype='object')
```

Out[3]:

```
<bound method NDFrame.head of           id                type  \
0       12333           boardgame
1      120677           boardgame
2      102794           boardgame
3       25613           boardgame
4        3076           boardgame
5       31260           boardgame
6      124742           boardgame
7       96848           boardgame
8       84876           boardgame
9       72125           boardgame
10       2651           boardgame
11     164153           boardgame
12     115746           boardgame
13     121921           boardgame
14      35677           boardgame
15      28720           boardgame
16     126163           boardgame
17     150376           boardgame
18      68448           boardgame
19      18602           boardgame
20     122515           boardgame
21      40834           boardgame
22      62219           boardgame
23      28143           boardgame
24     103885           boardgame
25         93           boardgame
26     146021           boardgame
27     110327           boardgame
28      37111           boardgame
29      12493           boardgame
...       ...                 ...
81282  184275  boardgameexpansion
81283  184281  boardgameexpansion
81284  184287           boardgame
81285  184292           boardgame
81286  184293           boardgame
81287  184298           boardgame
81288  184301           boardgame
81289  184306           boardgame
81290  184308  boardgameexpansion
81291  184311           boardgame
81292  184327           boardgame
81293  184337           boardgame
81294  184349           boardgame
81295  184351           boardgame
81296  184355  boardgameexpansion
81297  184357           boardgame
81298  184364           boardgame
81299  184368           boardgame
81300  184383           boardgame
81301  184390           boardgame
81302  184399           boardgame
81303  184403           boardgame
81304  184424           boardgame
81305  184439           boardgame
81306  184440           boardgame
81307  184441  boardgameexpansion
81308  184442           boardgame
```

```
81309  184443           boardgame
81310  184449           boardgame
81311  184451           boardgame

                                                  name  yearpublished  \
0                                     Twilight Struggle         2005.0
1                                         Terra Mystica         2012.0
2                               Caverna: The Cave Farmers         2013.0
3                  Through the Ages: A Story of Civilization         2006.0
4                                           Puerto Rico         2002.0
5                                             Agricola         2007.0
6                                    Android: Netrunner         2012.0
7                                 Mage Knight Board Game         2011.0
8                                The Castles of Burgundy         2011.0
9                                               Eclipse         2011.0
10                                          Power Grid         2004.0
11                               Star Wars: Imperial Assault         2014.0
12                         War of the Ring (second edition)         2012.0
13          Robinson Crusoe: Adventures on the Cursed Island         2012.0
14                                              Le Havre         2008.0
15                                                 Brass         2007.0
16                             Tzolk'in: The Mayan Calendar         2012.0
17                       Dead of Winter: A Crossroads Game         2014.0
18                                          7 Wonders         2010.0
19                                             Caylus         2005.0
20                                          Keyflower         2012.0
21                                Dominion: Intrigue         2009.0
22                                  Dominant Species         2010.0
23                               Race for the Galaxy         2007.0
24                       Star Wars: X-Wing Miniatures Game         2012.0
25                                          El Grande         1995.0
26                                     Eldritch Horror         2013.0
27                                   Lords of Waterdeep         2012.0
28                                Battlestar Galactica         2008.0
29                       Twilight Imperium (Third Edition)         2005.0
...                                                ...            ...
81282       Secrets of the Lost Tomb: The Elite Missions         2015.0
81283                             Coloretto: +3-Karte         2015.0
81284                                         Frantic         2015.0
81285                                          Patron         2015.0
81286            Jenga: Transformers  Rise of the Fallen         2009.0
81287                          WarChess-X: The Armageddon         2014.0
81288                              King Cat: Feline Feuds         2015.0
81289                                   Re-Extinction            0.0
81290                      Hellweg westfalicus: Dortmund         2015.0
81291                                   Fliegenschmaus         2015.0
81292                                     Rogue Squad         2016.0
81293                    Risk: Marvel Cinematic Universe         2015.0
81294                           The Luzon Campaign, 1945         2018.0
81295                                        Honey Wars         2015.0
81296                    Eaten by Zombies!: Burn it down!         2016.0
81297                  Wilson Gridiron Strategy Football         1970.0
81298                                Terra Incognita         2015.0
81299                                        Bone Head         2015.0
81300                             Disney Eye Found It!         2013.0
81301                                           Idfutr         2015.0
81302                                        Micropolis         2014.0
81303                                    Big Dope Deal         2008.0
81304                                  Mega Civilization         2015.0
81305                                       Succession!         2016.0
81306                                    Stick and Stones         2015.0
```

```
81307                                Micro Rome: Aegyptus         2015.0
81308  Trivial Pursuit: Marvel Cinematic Universe  Da...         2013.0
81309                                          BEARanoia         2015.0
81310                                            Freight         2015.0
81311                                Bingo Animal Kids         2010.0

       minplayers  maxplayers  playingtime  minplaytime  maxplaytime  mina
ge  \
0             2.0         2.0        180.0        180.0        180.0    1
3.0
1             2.0         5.0        150.0         60.0        150.0    1
2.0
2             1.0         7.0        210.0         30.0        210.0    1
2.0
3             2.0         4.0        240.0        240.0        240.0    1
2.0
4             2.0         5.0        150.0         90.0        150.0    1
2.0
5             1.0         5.0        150.0         30.0        150.0    1
2.0
6             2.0         2.0         45.0         45.0         45.0    1
4.0
7             1.0         4.0        150.0        150.0        150.0    1
4.0
8             2.0         4.0         90.0         30.0         90.0    1
2.0
9             2.0         6.0        200.0         60.0        200.0    1
4.0
10            2.0         6.0        120.0        120.0        120.0    1
2.0
11            2.0         5.0         90.0         90.0         90.0
0.0
12            2.0         4.0        150.0        150.0        150.0    1
3.0
13            1.0         4.0        180.0         90.0        180.0    1
4.0
14            1.0         5.0        200.0        100.0        200.0    1
2.0
15            3.0         4.0        180.0        120.0        180.0    1
3.0
16            2.0         4.0         90.0         90.0         90.0    1
3.0
17            2.0         5.0        210.0         45.0        210.0    1
4.0
18            2.0         7.0         30.0         30.0         30.0    1
0.0
19            2.0         5.0        150.0         60.0        150.0    1
2.0
20            2.0         6.0        120.0         90.0        120.0    1
2.0
21            2.0         4.0         30.0         30.0         30.0    1
3.0
22            2.0         6.0        240.0        120.0        240.0    1
4.0
23            2.0         4.0         60.0         30.0         60.0    1
2.0
24            2.0         4.0         60.0         60.0         60.0    1
4.0
25            2.0         5.0        120.0         60.0        120.0    1
2.0
26            1.0         8.0        240.0        120.0        240.0    1
```

```
4.0
27          2.0         5.0         60.0         60.0         60.0    1
2.0
28          3.0         6.0        240.0        120.0        240.0    1
0.0
29          3.0         6.0        240.0        180.0        240.0    1
2.0
...         ...         ...          ...          ...          ...
...
81282       1.0         6.0        180.0         60.0        180.0    1
3.0
81283       2.0         5.0         30.0         30.0         30.0
8.0
81284       2.0         8.0         45.0          5.0         45.0    1
2.0
81285       2.0         5.0         40.0         10.0         40.0
8.0
81286       2.0        99.0          0.0          0.0          0.0
6.0
81287       2.0        15.0          0.0         60.0          0.0    1
4.0
81288       2.0         4.0         60.0         30.0         60.0    1
2.0
81289       2.0         4.0         20.0          5.0         20.0
0.0
81290       2.0         4.0         45.0          0.0         45.0    1
0.0
81291       2.0         4.0         20.0         15.0         20.0
6.0
81292       1.0         4.0          0.0          0.0          0.0    1
3.0
81293       2.0         5.0         60.0         60.0         60.0    1
0.0
81294       1.0         1.0          0.0          0.0          0.0
0.0
81295       2.0         4.0         90.0         45.0         90.0    1
2.0
81296       2.0         6.0         90.0         30.0         90.0    1
2.0
81297       2.0         2.0        120.0         60.0        120.0    1
2.0
81298       1.0         1.0        120.0         45.0        120.0    1
2.0
81299       2.0         9.0          0.0          0.0          0.0
0.0
81300       1.0         6.0         30.0         30.0         30.0
4.0
81301       2.0         6.0         90.0         60.0         90.0    1
0.0
81302       2.0         4.0         60.0         30.0         60.0    1
2.0
81303       2.0         6.0          0.0         60.0          0.0    1
8.0
81304       5.0        18.0        720.0        360.0        720.0    1
2.0
81305       1.0         6.0         10.0          5.0         10.0
8.0
81306       2.0         2.0        120.0         45.0        120.0
0.0
81307       1.0         1.0          0.0         30.0          0.0    1
0.0
```

```
81308         2.0         0.0          0.0          0.0          0.0    1
2.0
81309         2.0        15.0          1.0          1.0          1.0
0.0
81310         2.0         4.0         60.0         30.0         60.0
8.0
81311         1.0         6.0         10.0         10.0         10.0
2.0

       users_rated  average_rating  bayes_average_rating  total_owners  \
0            20113         8.33774               8.22186         26647
1            14383         8.28798               8.14232         16519
2             9262         8.28994               8.06886         12230
3            13294         8.20407               8.05804         14343
4            39883         8.14261               8.04524         44362
5            39714         8.11957               8.03847         47522
6            15281         8.16760               7.97822         24381
7            12697         8.15901               7.96929         18769
8            15461         8.07879               7.95011         20558
9            15709         8.07933               7.93244         17611
10           34422         7.98880               7.91794         38633
11            3980         8.43944               7.91643          8477
12            3870         8.35044               7.88643          6257
13           10539         8.09283               7.88503         15896
14           15774         7.99115               7.88172         16429
15            8785         8.03071               7.85824          9171
16           12143         7.98673               7.83148         13958
17            9188         8.05776               7.82389         13692
18           36732         7.87047               7.79413         44982
19           19160         7.89829               7.78071         18885
20            6753         7.98786               7.74780          8599
21           19261         7.85479               7.73936         26403
22           10187         7.89276               7.72445         11003
23           28655         7.80281               7.72433         33736
24           11194         7.93301               7.72151         19899
25           15853         7.83279               7.71949         15556
26            8388         7.97188               7.71614         13668
27           19864         7.82181               7.70704         24419
28           20833         7.82752               7.70269         22735
29           12064         7.88473               7.69170         13771
...            ...             ...                   ...           ...
81282            0         0.00000               0.00000             2
81283            0         0.00000               0.00000             1
81284            0         0.00000               0.00000             0
81285            0         0.00000               0.00000             2
81286            0         0.00000               0.00000             0
81287            0         0.00000               0.00000             0
81288            0         0.00000               0.00000             0
81289            0         0.00000               0.00000             0
81290            0         0.00000               0.00000             0
81291            0         0.00000               0.00000             0
81292            0         0.00000               0.00000             0
81293            0         0.00000               0.00000             0
81294            0         0.00000               0.00000             0
81295            0         0.00000               0.00000             0
81296            0         0.00000               0.00000             0
81297            0         0.00000               0.00000             0
81298            0         0.00000               0.00000             1
81299            0         0.00000               0.00000             0
81300            0         0.00000               0.00000             1
81301            0         0.00000               0.00000             0
```

```
81302            0         0.00000              0.00000            0
81303            0         0.00000              0.00000            0
81304            0         0.00000              0.00000            0
81305            0         0.00000              0.00000            0
81306            0         0.00000              0.00000            0
81307            0         0.00000              0.00000            0
81308            0         0.00000              0.00000            0
81309            0         0.00000              0.00000            0
81310            0         0.00000              0.00000            0
81311            0         0.00000              0.00000            0

       total_traders  total_wanters  total_wishers  total_comments  \
0                372           1219           5865            5347
1                132           1586           6277            2526
2                 99           1476           5600            1700
3                362           1084           5075            3378
4                795            861           5414            9173
5                837            958           6402            9310
6                680            627           3244            3202
7                367           1116           5427            2861
8                215            929           3681            3244
9                273           1108           5581            3188
10               550           1171           6157            7531
11                57            701           2970             736
12                71            677           2431             771
13               217           1379           5821            2109
14               205           1343           5149            3458
15               149            798           2858            2259
16               120           1056           3945            2144
17               144           1086           4956            1602
18               464           1046           5806            7126
19               353            878           4011            4984
20                78           1017           3197            1442
21               374            461           2281            3005
22               243           1017           4285            2613
23               708            761           4597            6807
24               334            421           1826            2183
25               261           1130           3787            4250
26               147            643           3447            1585
27               257            995           4706            3898
28               514            643           4005            4817
29               287            927           4650            3369
...              ...            ...            ...             ...
81282              0              0              0               0
81283              0              1              1               0
81284              0              0              0               0
81285              0              0              1               1
81286              0              0              0               0
81287              0              0              0               0
81288              0              0              0               0
81289              0              0              0               0
81290              0              1              1               0
81291              0              0              0               0
81292              0              0              1               0
81293              0              0              0               0
81294              0              0              0               0
81295              0              0              0               0
81296              0              1              2               0
81297              0              0              0               0
81298              0              0              1               0
81299              0              0              0               0
```

81300	0	0	0	0
81301	0	0	0	0
81302	0	0	1	0
81303	0	0	0	0
81304	0	0	3	0
81305	0	0	0	0
81306	0	0	0	0
81307	0	0	0	0
81308	0	0	0	0
81309	0	0	0	0
81310	0	0	0	0
81311	0	0	0	0

	total_weights	average_weight
0	2562	3.4785
1	1423	3.8939
2	777	3.7761
3	1642	4.1590
4	5213	3.2943
5	5065	3.6160
6	1260	3.3103
7	1409	4.1292
8	1176	3.0442
9	1486	3.6359
10	3998	3.2911
11	360	3.2250
12	288	3.9375
13	896	3.6328
14	1450	3.7531
15	1012	3.8646
16	933	3.5595
17	608	2.9408
18	2917	2.3384
19	2894	3.8252
20	517	3.3056
21	1121	2.4469
22	1070	4.0103
23	2922	2.9617
24	911	2.4577
25	1861	3.0919
26	674	3.2834
27	1493	2.5177
28	1783	3.2013
29	1761	4.1942
...	...	...
81282	0	0.0000
81283	0	0.0000
81284	0	0.0000
81285	1	3.0000
81286	0	0.0000
81287	0	0.0000
81288	0	0.0000
81289	0	0.0000
81290	0	0.0000
81291	0	0.0000
81292	0	0.0000
81293	0	0.0000
81294	0	0.0000
81295	0	0.0000
81296	0	0.0000
81297	0	0.0000

```
81298              0          0.0000
81299              0          0.0000
81300              0          0.0000
81301              0          0.0000
81302              0          0.0000
81303              0          0.0000
81304              0          0.0000
81305              0          0.0000
81306              0          0.0000
81307              0          0.0000
81308              0          0.0000
81309              0          0.0000
81310              0          0.0000
81311              0          0.0000

[81312 rows x 20 columns]>
```

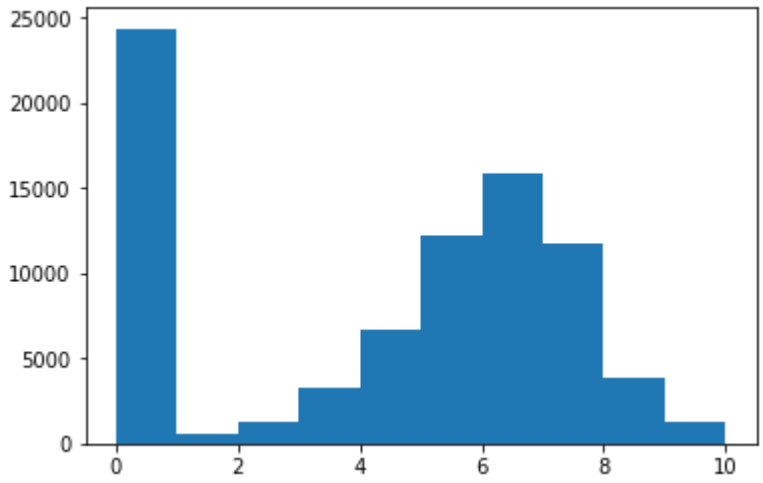
2. Visualizing and Exploring the data

In [4]:

```
plt.hist(game['average_rating'])
```

Out[4]:

```
(array([24380.,   606.,  1325.,  3303.,  6687., 12277., 15849., 11737.,
         3860.,  1288.]),
 array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.]),
 <a list of 10 Patch objects>)
```



As can be observed from the above histogram, a majority of the games in the dataframe are given an average rating of zero. These rows need to be closely observe to determine the reason for the zero rating.

In [5]:

```
# printing the 10 of the rows which has 0 average rating and also printing those which
 have average rating greater than 0

print(game[game["average_rating"] == 0].iloc[0:10])
print(game[game["average_rating"] > 0].iloc[0:10])
```

	id	type	name	yearpublished	minplayers	maxplayers	playingtime	minplaytime	maxplaytime	minage
13048	318	boardgame	Looney Leo	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13068	579	boardgame	Field of Fire	2002.0	2.0	0.0	0.0	0.0	0.0	12.0
13114	894	boardgame	LifeLinks	2000.0	1.0	8.0	10.0	10.0	10.0	8.0
13116	897	boardgame	Dear Abby	1992.0	3.0	4.0	60.0	60.0	60.0	13.0
13124	946	boardgame	Rolazone	1999.0	2.0	2.0	30.0	30.0	30.0	0.0
13127	962	boardgame	Contango	2000.0	2.0	6.0	90.0	90.0	90.0	10.0
13169	1097	boardgame	Don't Give Up Your Day Job!	2000.0	6.0	41.0	120.0	120.0	120.0	8.0
13180	1151	boardgame	Money, Power, Respect	2000.0	2.0	6.0	90.0	90.0	90.0	8.0
13181	1154	boardgame	Nuts! To You	1987.0	2.0	2.0	20.0	20.0	20.0	6.0
13188	1180	boardgame	Tower of Babylon	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	users_rated	average_rating	bayes_average_rating	total_owners	total_traders	total_wanters	total_wishers	total_comments	total_weights	average_weight
13048	0	0.0	0.0	0	0	0	1	0	0	0.0
13068	0	0.0	0.0	2	0	0	1	0		
13114	0	0.0	0.0	2	0	1	1	0		
13116	0	0.0	0.0	7	1	0	0	0		
13124	0	0.0	0.0	0	0	0	3	0		
13127	0	0.0	0.0	0	0	0	2	1		
13169	0	0.0	0.0	0	0	0	2	0		
13180	0	0.0	0.0	1	0	0	2	0		
13181	0	0.0	0.0	2	0	0	1	0		
13188	0	0.0	0.0	5	0	2	2	0		

```
13068              0             0.0
13114              0             0.0
13116              0             0.0
13124              0             0.0
13127              0             0.0
13169              0             0.0
13180              0             0.0
13181              0             0.0
13188              0             0.0
       id       type                                    name  \
0   12333  boardgame                       Twilight Struggle
1  120677  boardgame                           Terra Mystica
2  102794  boardgame                  Caverna: The Cave Farmers
3   25613  boardgame  Through the Ages: A Story of Civilization
4    3076  boardgame                             Puerto Rico
5   31260  boardgame                                Agricola
6  124742  boardgame                      Android: Netrunner
7   96848  boardgame                   Mage Knight Board Game
8   84876  boardgame                 The Castles of Burgundy
9   72125  boardgame                                  Eclipse

   yearpublished  minplayers  maxplayers  playingtime  minplaytime  \
0         2005.0         2.0         2.0        180.0        180.0
1         2012.0         2.0         5.0        150.0         60.0
2         2013.0         1.0         7.0        210.0         30.0
3         2006.0         2.0         4.0        240.0        240.0
4         2002.0         2.0         5.0        150.0         90.0
5         2007.0         1.0         5.0        150.0         30.0
6         2012.0         2.0         2.0         45.0         45.0
7         2011.0         1.0         4.0        150.0        150.0
8         2011.0         2.0         4.0         90.0         30.0
9         2011.0         2.0         6.0        200.0         60.0

   maxplaytime  minage  users_rated  average_rating  bayes_average_rating
\
0        180.0    13.0        20113         8.33774               8.22186
1        150.0    12.0        14383         8.28798               8.14232
2        210.0    12.0         9262         8.28994               8.06886
3        240.0    12.0        13294         8.20407               8.05804
4        150.0    12.0        39883         8.14261               8.04524
5        150.0    12.0        39714         8.11957               8.03847
6         45.0    14.0        15281         8.16760               7.97822
7        150.0    14.0        12697         8.15901               7.96929
8         90.0    12.0        15461         8.07879               7.95011
9        200.0    14.0        15709         8.07933               7.93244

   total_owners  total_traders  total_wanters  total_wishers  total_commen
ts  \
0         26647            372           1219           5865            53
47
1         16519            132           1586           6277            25
26
2         12230             99           1476           5600            17
00
3         14343            362           1084           5075            33
78
4         44362            795            861           5414            91
73
5         47522            837            958           6402            93
10
6         24381            680            627           3244            32
```

```
02
7          18769             367            1116             5427             28
61
8          20558             215             929             3681             32
44
9          17611             273            1108             5581             31
88
```

	total_weights	average_weight
0	2562	3.4785
1	1423	3.8939
2	777	3.7761
3	1642	4.1590
4	5213	3.2943
5	5065	3.6160
6	1260	3.3103
7	1409	4.1292
8	1176	3.0442
9	1486	3.6359

Thus, from observing the rows with 0 average rating it can be reasonably concluded that all the games which have 0 rating have 0 users rated. Thus, for all the games which have not been played or published or not rated the average rating showed up to be 0. Thus, these rows could be removed from the data frame. Further, if any missing values are prevalent in the dataframe, those rows must also be removed.

In [6]:

```
# Finding the number of missing values in each column
game.isnull().sum()
```

Out[6]:

```
id                      0
type                    0
name                   41
yearpublished           3
minplayers              3
maxplayers              3
playingtime             3
minplaytime             3
maxplaytime             3
minage                  3
users_rated             0
average_rating          0
bayes_average_rating    0
total_owners            0
total_traders           0
total_wanters           0
total_wishers           0
total_comments          0
total_weights           0
average_weight          0
dtype: int64
```

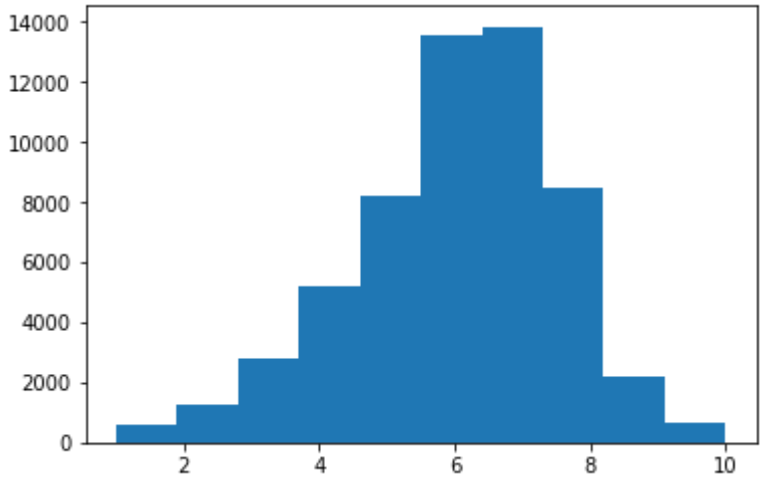
Thus, it can be observed that a total number of missing values is 41 which could be removed from the dataframe.

In [7]:

```
# Removing the rows with missing values
game = game.dropna(axis = 0)

# Removing rows with 0 user reviews
game = game[game["users_rated"] > 0]

#Plotting the histogram again
plt.hist(game["average_rating"])
```

Out[7]:

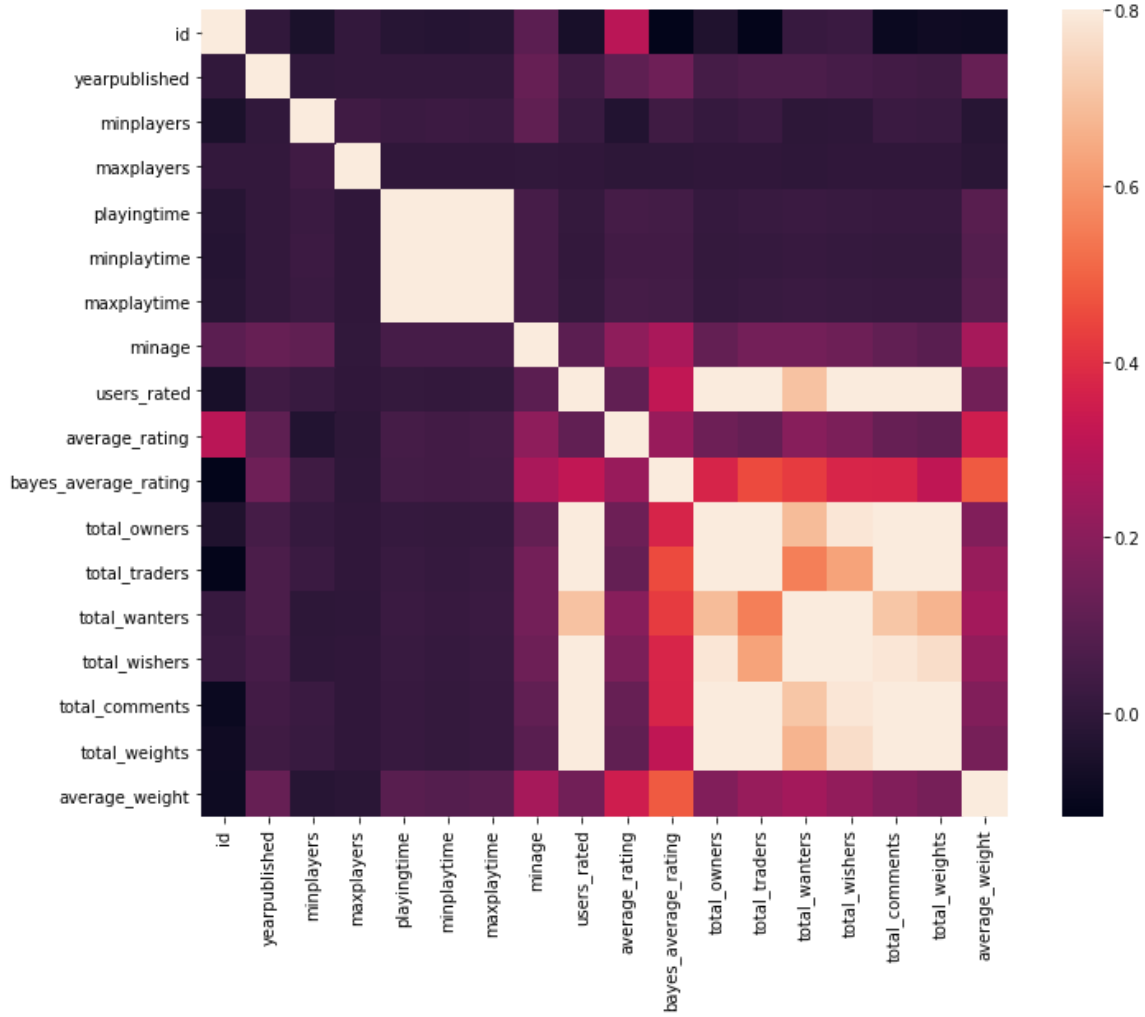
```
(array([  602.,  1231.,  2824.,  5206.,  8223., 13593., 13849.,  8470.,
         2224.,   672.]),
 array([ 1. ,  1.9,  2.8,  3.7,  4.6,  5.5,  6.4,  7.3,  8.2,  9.1, 10.
]),
 <a list of 10 Patch objects>)
```



In order to know if there are any strong correlations prevalent in the dataset, correlation matrix has been plotted as follows

In [8]:

```
#  Correlation Matrix
corrmat = game.corr()
fig = plt.figure(figsize =(12,9))

sns.heatmap(corrmat, vmax =0.8, square = True)
plt.show()
```



From the correlation matrix the correlation between values of different columns could be established. It can be seen that if the square colour is lighter (towards white), the columns were highly correlated and darker implies no correlation. Moreover, columns which are collinear showed maximum correlation value like the 'playingtime', 'minplaytime' and 'maxplaytime' and 'average_rating' and 'bayes_average_rating' Further, some of the columns like 'type', 'name' and 'yearpublished' could be removed rightaway as they provide negligible information pertaining to the predictor 'average_rating'. Lastly, columns such as 'id' and 'bayes_average_rating' must be removed as the high correlation with the average_rating and collinearity respectively would adversely affect the machine learning model.

In order to ascertain the type of relationship the label 'average_rating' has with the determined variables, a scatter plot matrix between each variable column and the label column could be plotted

In [9]:

```
game
```

Out[9]:

	id	type	name	yearpublished	minplayers	maxplay
0	12333	boardgame	Twilight Struggle	2005.0	2.0	2.0
1	120677	boardgame	Terra Mystica	2012.0	2.0	5.0
2	102794	boardgame	Caverna: The Cave Farmers	2013.0	1.0	7.0
3	25613	boardgame	Through the Ages: A Story of Civilization	2006.0	2.0	4.0
4	3076	boardgame	Puerto Rico	2002.0	2.0	5.0
5	31260	boardgame	Agricola	2007.0	1.0	5.0
6	124742	boardgame	Android: Netrunner	2012.0	2.0	2.0
7	96848	boardgame	Mage Knight Board Game	2011.0	1.0	4.0
8	84876	boardgame	The Castles of Burgundy	2011.0	2.0	4.0
9	72125	boardgame	Eclipse	2011.0	2.0	6.0
10	2651	boardgame	Power Grid	2004.0	2.0	6.0
11	164153	boardgame	Star Wars: Imperial Assault	2014.0	2.0	5.0
12	115746	boardgame	War of the Ring (second edition)	2012.0	2.0	4.0
13	121921	boardgame	Robinson Crusoe: Adventures on the Cursed Island	2012.0	1.0	4.0
14	35677	boardgame	Le Havre	2008.0	1.0	5.0
15	28720	boardgame	Brass	2007.0	3.0	4.0

	id	type	name	yearpublished	minplayers	maxplay
16	126163	boardgame	Tzolk'in: The Mayan Calendar	2012.0	2.0	4.0
17	150376	boardgame	Dead of Winter: A Crossroads Game	2014.0	2.0	5.0
18	68448	boardgame	7 Wonders	2010.0	2.0	7.0
19	18602	boardgame	Caylus	2005.0	2.0	5.0
20	122515	boardgame	Keyflower	2012.0	2.0	6.0
21	40834	boardgame	Dominion: Intrigue	2009.0	2.0	4.0
22	62219	boardgame	Dominant Species	2010.0	2.0	6.0
23	28143	boardgame	Race for the Galaxy	2007.0	2.0	4.0
24	103885	boardgame	Star Wars: X-Wing Miniatures Game	2012.0	2.0	4.0
25	93	boardgame	El Grande	1995.0	2.0	5.0
26	146021	boardgame	Eldritch Horror	2013.0	1.0	8.0
27	110327	boardgame	Lords of Waterdeep	2012.0	2.0	5.0
28	37111	boardgame	Battlestar Galactica	2008.0	3.0	6.0
29	12493	boardgame	Twilight Imperium (Third Edition)	2005.0	3.0	6.0
...	...	...	...	...	...	...
81134	183804	boardgame	Super Dude Bros	2015.0	2.0	6.0
81160	183860	boardgameexpansion	XenoShyft: Onslaught Hive Expansion	2015.0	1.0	4.0
81162	183868	boardgame	Awkward Turtle	0.0	4.0	0.0

	id	type	name	yearpublished	minplayers	maxplay
81176	183915	boardgameexpansion	Rum & Bones: Mercenaries Promo Set #1	2015.0	2.0	6.0
81177	183917	boardgameexpansion	Rum & Bones: Mercenaries Promos Set #2	2015.0	2.0	6.0
81178	183918	boardgameexpansion	Rum & Bones: Mercenaries Heroes Set #2	2015.0	2.0	6.0
81179	183919	boardgameexpansion	Rum & Bones: Bone Devils Mix	2015.0	2.0	6.0
81180	183920	boardgameexpansion	Rum & Bones: Wellsport Brotherhood Mix	2015.0	2.0	6.0
81181	183921	boardgameexpansion	Rum & Bones: Helrokkers	2015.0	2.0	6.0
81184	183942	boardgame	We Happy Few: The Battle of Agincourt	2015.0	1.0	2.0
81188	183959	boardgame	Latice	2015.0	2.0	4.0
81190	183963	boardgame	Invaders from Dimension X!	2015.0	1.0	2.0
81192	183966	boardgameexpansion	Star Wars: X-Wing Miniatures Game T-70 X-Wing...	2015.0	2.0	2.0

	id	type	name	yearpublished	minplayers	maxplay
81193	183967	boardgameexpansion	Star Wars: X-Wing Miniatures Game TIE/fo Figh...	2015.0	2.0	2.0
81195	183969	boardgameexpansion	Spyfall: The Box Is Not Enough	2015.0	3.0	8.0
81198	183975	boardgame	Smile & Money	2015.0	2.0	4.0
81199	183976	boardgame	Dice Bazaar	2016.0	2.0	4.0
81204	184002	boardgame	Angels and Demons: Battle for Humanity	2015.0	6.0	25.0
81234	184079	boardgame	Go Monster!	2015.0	2.0	4.0
81249	184159	boardgame	Elefun & Friends Mouse Trap	2013.0	2.0	3.0
81254	184174	boardgameexpansion	Rum & Bones: Bone Devils Heroes Set #1	2015.0	2.0	6.0
81255	184180	boardgameexpansion	Rum & Bones: Wellsport Brotherhood Heroes Set #1	2015.0	2.0	6.0
81256	184182	boardgameexpansion	Rum & Bones: Mazu's Dreadful Curse Heroes Set #1	2015.0	2.0	6.0
81257	184183	boardgameexpansion	Rum & Bones: La Brise Sanguine Heroes Set #1	2015.0	2.0	6.0

	id	type	name	yearpublished	minplayers	maxplay
81258	184184	boardgameexpansion	Rum & Bones: Mercenaries Heroes Set #1	2015.0	2.0	6.0
81260	184187	boardgameexpansion	Rum & Bones: Skullkicker heroes	2015.0	2.0	6.0
81261	184189	boardgameexpansion	Rum & Bones: Luck Goddesses	2015.0	2.0	6.0
81263	184195	boardgameexpansion	Rum & Bones: Mercenary Tide Deck	2015.0	2.0	6.0
81278	184258	boardgame	Rocket Shogi	2012.0	2.0	2.0
81279	184260	boardgame	Tricky Pirates	2015.0	2.0	4.0

56894 rows × 20 columns

First, the columns to be chosen as variables and the target are converted into a list. This is critical because, sklearn package is not able to work with dataframes.

Second, the columns mentioned above are removed from the lists of columns created from the dataframe.

Thirdly, the dataframe is split randomly into test and training dataframes.

Finally, the values of the column to be predicted (in this case 'average_rating') is saved as a seperate list and the columns chosen as the variables were saved seperately.

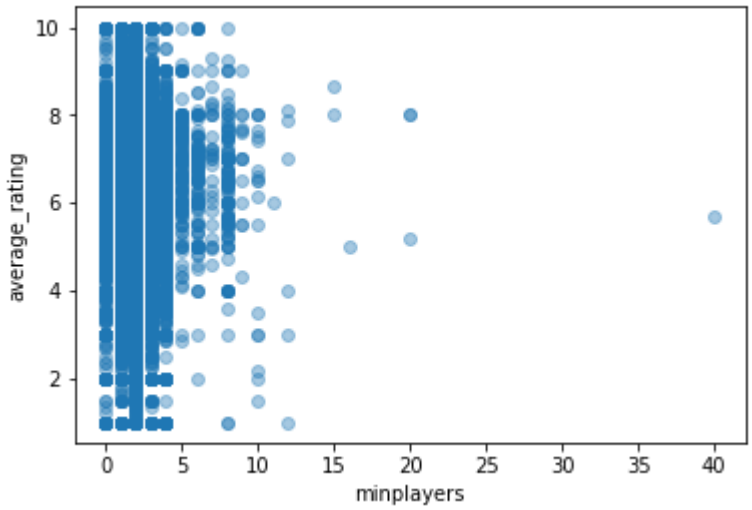
In [10]:

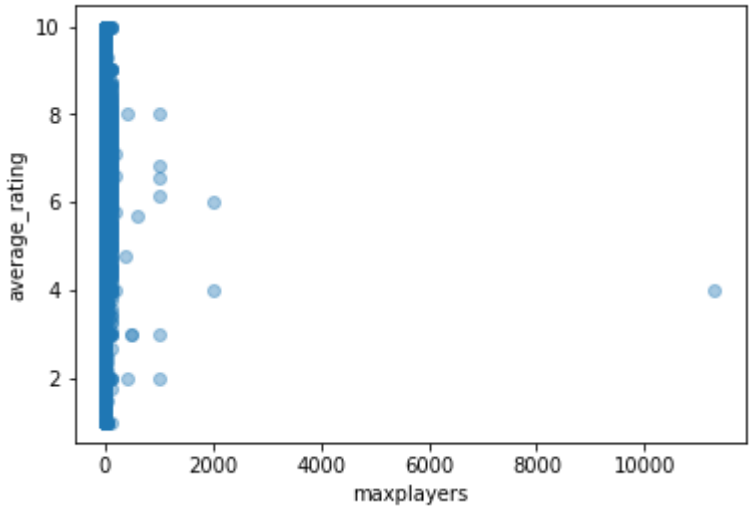
```
columns = game.columns.tolist()
# Filtering the columns to be removed
columns =  [c for c in columns if c not in ['id', 'name', 'type', 'average_rating', 'ba
yes_average_rating', 'yearpublished']]
target = 'average_rating'

# Separating the variables and target and storing as arrays
X_Var = game[columns].values
Y_tar = game[target].values

print(X_Var)
print(Y_tar)
```

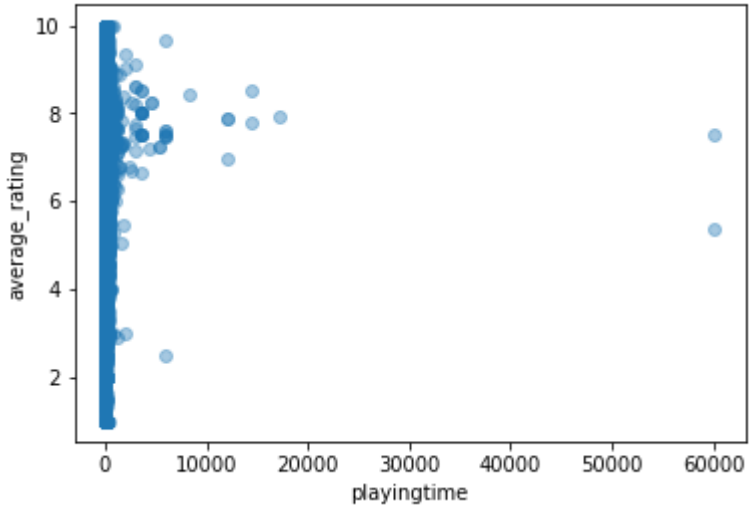
```
[[2.0000e+00 2.0000e+00 1.8000e+02 ... 5.3470e+03 2.5620e+03 3.4785e+00]
 [2.0000e+00 5.0000e+00 1.5000e+02 ... 2.5260e+03 1.4230e+03 3.8939e+00]
 [1.0000e+00 7.0000e+00 2.1000e+02 ... 1.7000e+03 7.7700e+02 3.7761e+00]
 ...
 [2.0000e+00 6.0000e+00 0.0000e+00 ... 2.0000e+00 1.0000e+00 1.0000e+00]
 [2.0000e+00 2.0000e+00 0.0000e+00 ... 1.0000e+00 0.0000e+00 0.0000e+00]
 [2.0000e+00 4.0000e+00 6.0000e+01 ... 0.0000e+00 2.0000e+00 1.5000e+00]]
[8.33774 8.28798 8.28994 ... 8.      7.      7.     ]
```

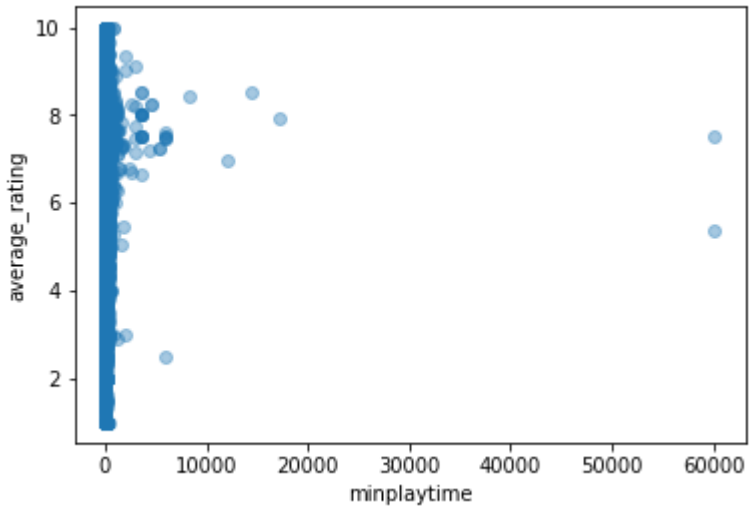
In [11]:

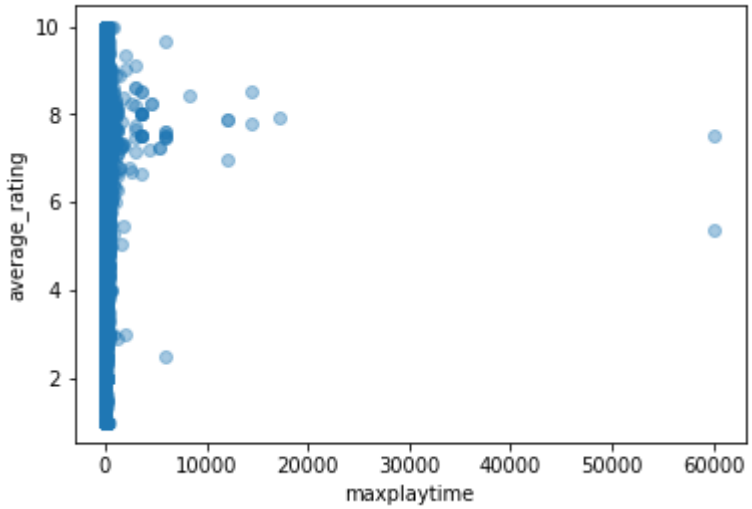
```
# Plotting scatter plots between
for c in columns:
    plt.scatter(game[c],game[target], alpha = 0.4)
    plt.xlabel(c)
    plt.ylabel(target)
    plt.show()
```

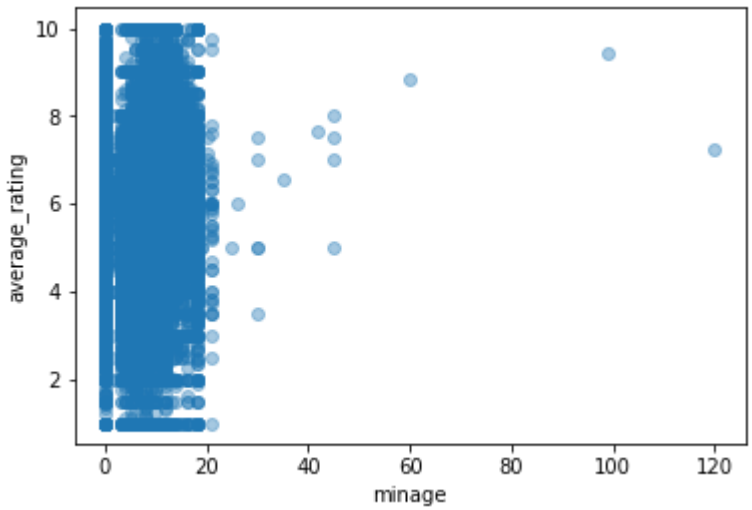

average_rating
minplayers
0
5
10
15
20
25
30
35
40
2
4
6
8
10

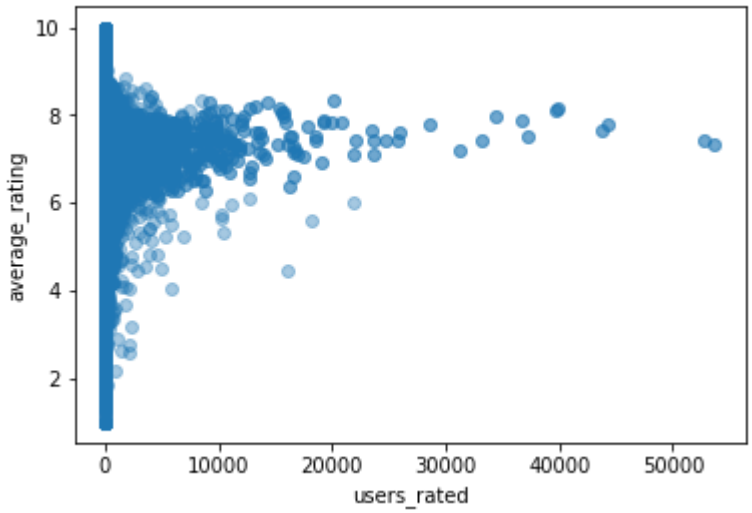

average_rating
maxplayers
0
2000
4000
6000
8000
10000
2
4
6
8
10

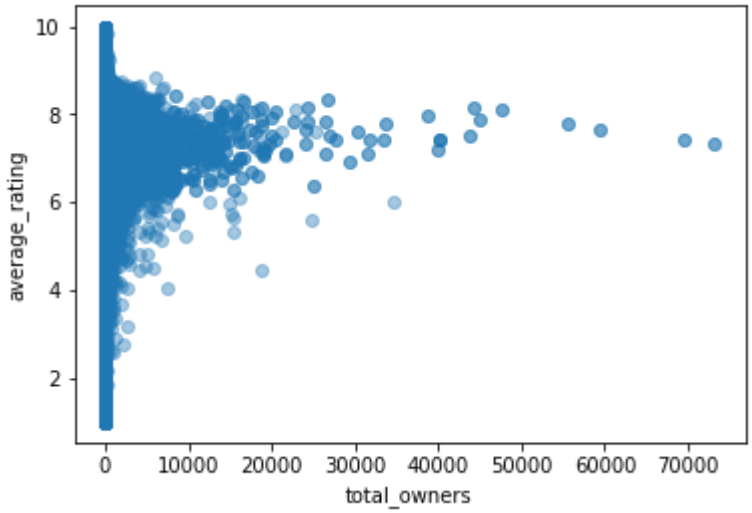

average_rating
playingtime
0
10000
20000
30000
40000
50000
60000
2
4
6
8
10

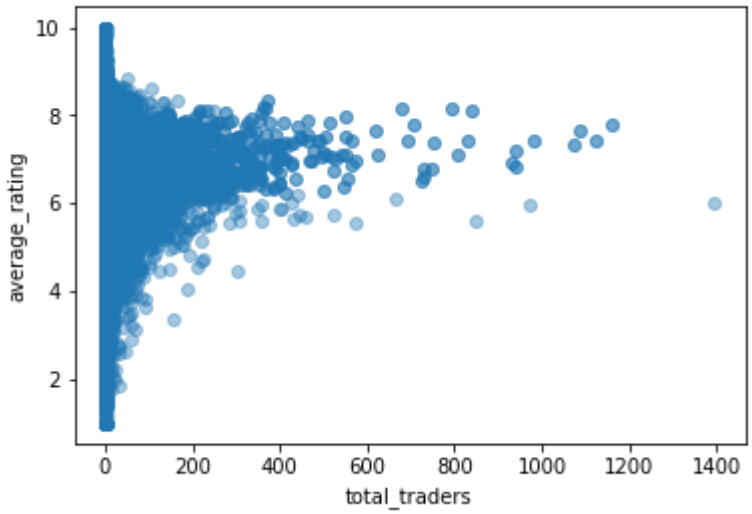

average_rating
minplaytime
0
10000
20000
30000
40000
50000
60000
2
4
6
8
10

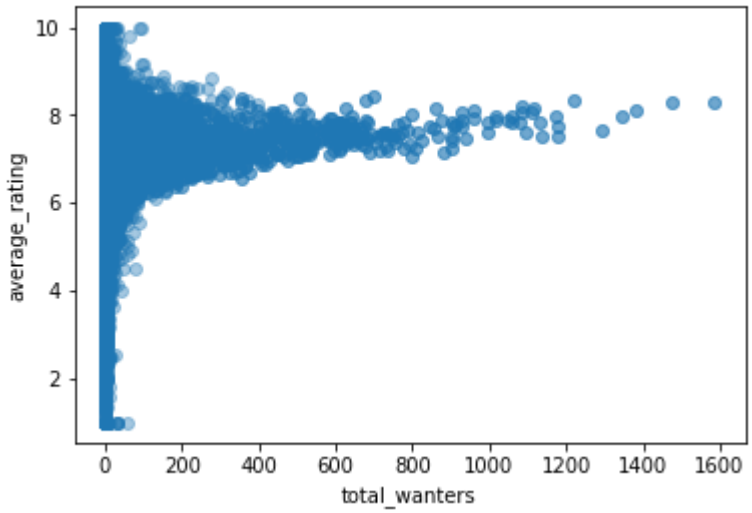

average_rating
maxplaytime
0
10000
20000
30000
40000
50000
60000
2
4
6
8
10

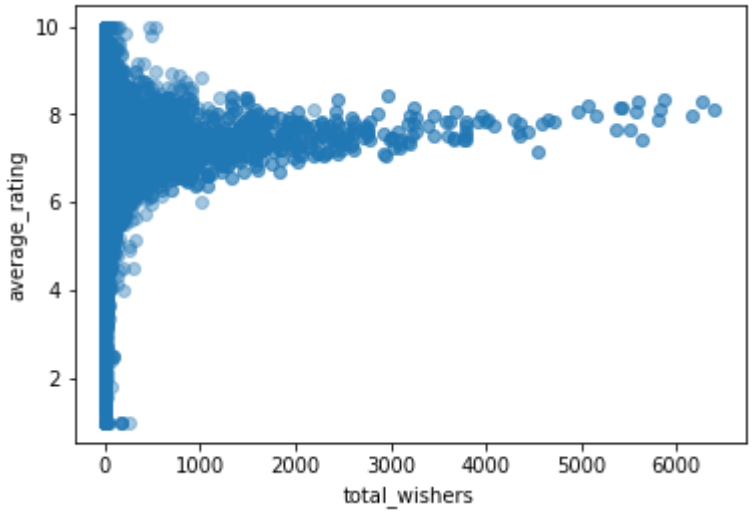

average_rating
minage
0
20
40
60
80
100
120
2
4
6
8
10

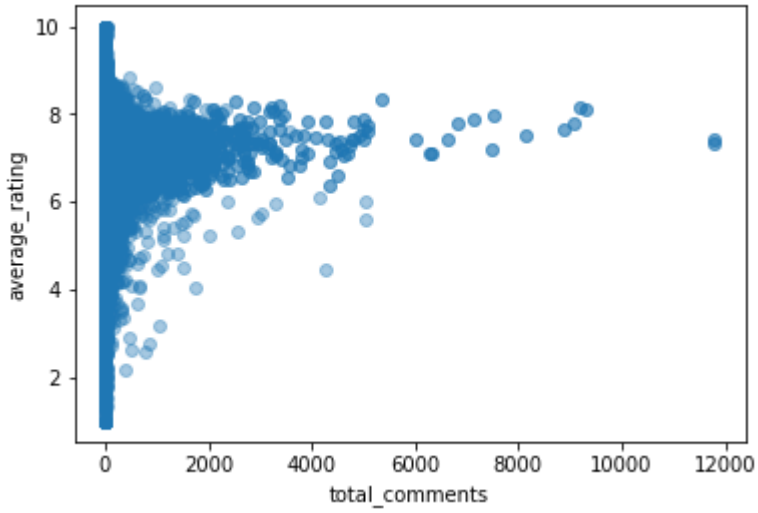

average_rating
users_rated
0
10000
20000
30000
40000
50000
2
4
6
8
10

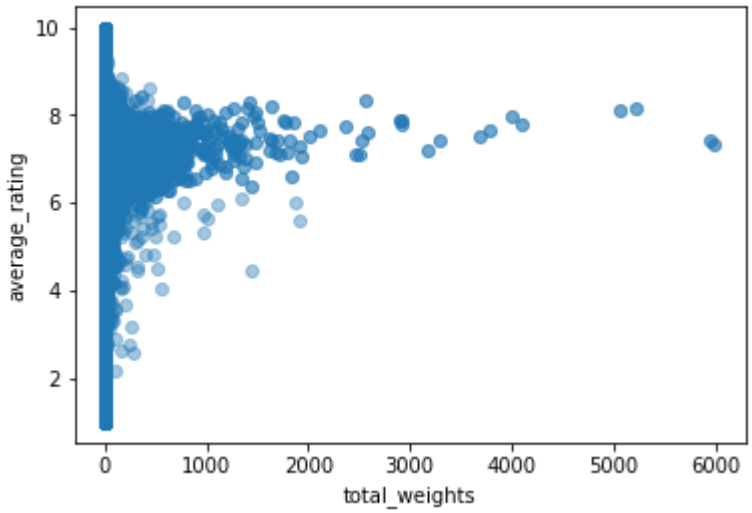

average_rating
total_owners
0
10000
20000
30000
40000
50000
60000
70000
2
4
6
8
10

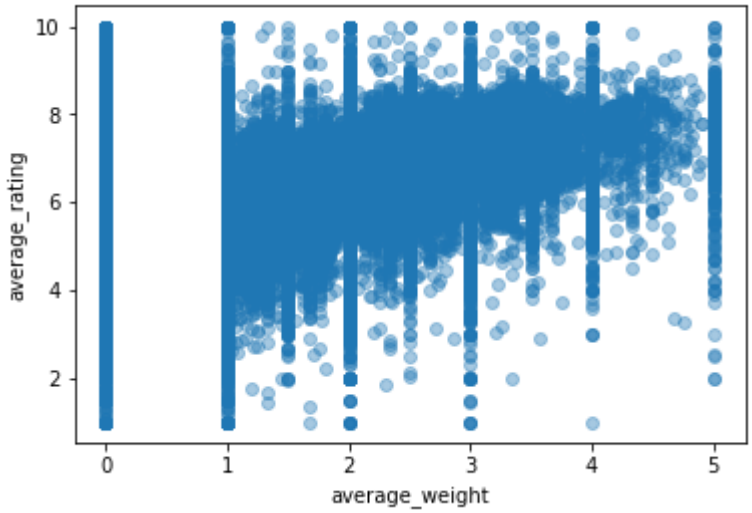

average_rating
total_traders
0
200
400
600
800
1000
1200
1400
2
4
6
8
10


average_rating
total_wanters
0
200
400
600
800
1000
1200
1400
1600
2
4
6
8
10


average_rating
total_wishers
0
1000
2000
3000
4000
5000
6000
2
4
6
8
10


average_rating
total_comments
0
2000
4000
6000
8000
10000
12000
2
4
6
8
10





Judging from the previous plots, following observations and changes could be made in the future if there is any improvement to be made to the model:

1. most do not correlate linearly
2. the values could be scaled such that each feature is in proper scale (Consider this as a potential improvement for the future)

3. Data Preparation and Model Training

In [12]:

```
# Splitting the above obtained arrays into testing and training arrays

X_train, X_test, Y_train, Y_test = train_test_split(X_Var, Y_tar, test_size = 0.2, rand
om_state = 53)

print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(45515, 14)
(11379, 14)
(45515,)
(11379,)
```

Thus, the test and training datasets were chosen using the train_test_split method and the models would be trained and evaluated using these data sets.

Chosing to compare between multivariate linear regression, decision forest/random forest regression (and neural network regression models), each model was trained using the training set and cross validated on the training set before evaluating using the testing set

In [23]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor

LR = LinearRegression()
RFR = RandomForestRegressor(n_estimators = 100, min_samples_leaf = 10, random_state = 1
)

LR.fit(X_train,Y_train)
LR_prediction = LR.predict(X_test)
mse_LR = mean_squared_error(LR_prediction, Y_test)

RFR.fit(X_train,Y_train)
RFR_prediction = RFR.predict(X_test)
mse_RFR = mean_squared_error(RFR_prediction, Y_test)

print('Mean Square Error for Linear Regression Model is {}', mse_LR)
print('Mean Square Error for Linear Regression Model is {}', mse_RFR)
```

```
Mean Square Error for Linear Regression Model is {} 2.1088236681874415
Mean Square Error for Linear Regression Model is {} 1.5791483794644599
```