# Board Game Review Prediction

We git clone https://github.com/ThaWeatherman/scrapers.git (https://github.com/ThaWeatherman/scrapers.git)

And copy the games.csv file to current directory

In [1]:
```
import sys
import pandas
import matplotlib
import seaborn
import sklearn

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

```
3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v.1912 64 bit (AMD64)]
0.23.4
2.2.3
0.9.0
0.21.1
```

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

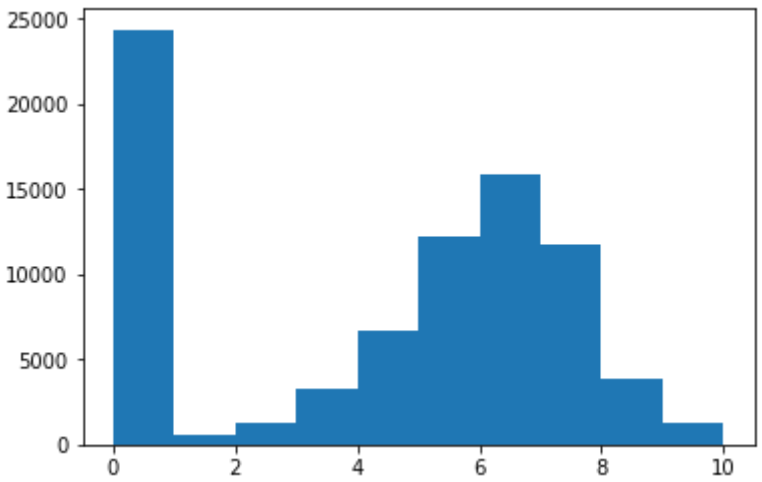
In [3]:
```
#Load our dataset

#pandas' dataframe

games = pandas.read_csv('games.csv')
```

In [4]:
```
#print the names of the columns in games
#know the shape

print(games.columns)
print(games.shape)
```

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

In [5]:
```
#make a histogram of all the ratings of in the average_rating column
#our target is the 'average_rating' column

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

Out[5]:
```
(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>)
```

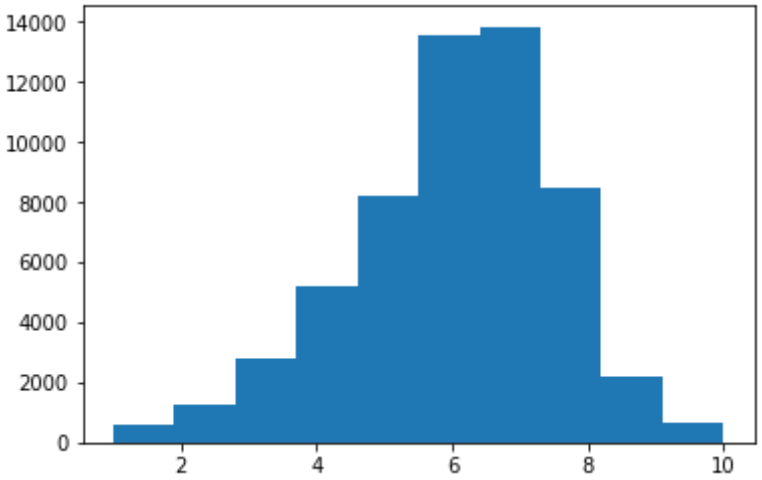


In [6]:
```
#print the first row of all the games with zero scores
print(games[games['average_rating'] == 0].iloc[0])

#print the first row of all the games with scores greater than zero
print(games[games['average_rating'] > 0].iloc[0])
```

```
id                            318
type                    boardgame
name                   Looney Leo
yearpublished                   0
minplayers                      0
maxplayers                      0
playingtime                     0
minplaytime                     0
maxplaytime                     0
minage                          0
users_rated                     0
average_rating                  0
bayes_average_rating            0
total_owners                    0
total_traders                   0
total_wanters                   0
total_wishers                   1
total_comments                  0
total_weights                   0
average_weight                  0
Name: 13048, dtype: object
id                              12333
type                        boardgame
name                Twilight Struggle
yearpublished                    2005
minplayers                          2
maxplayers                          2
playingtime                       180
minplaytime                       180
maxplaytime                       180
minage                             13
users_rated                     20113
average_rating                8.33774
bayes_average_rating          8.22186
total_owners                    26647
total_traders                     372
total_wanters                    1219
total_wishers                    5865
total_comments                   5347
total_weights                    2562
```

```
average_weight                    3.4785
Name: 0, dtype: object
```

In [7]:
```
#remove any rows with user review

games = games[games['users_rated'] > 0]

#remove any rows with missing values

games = games.dropna(axis=0)


#make a new histogram
plt.hist(games["average_rating"])
plt.show()
```

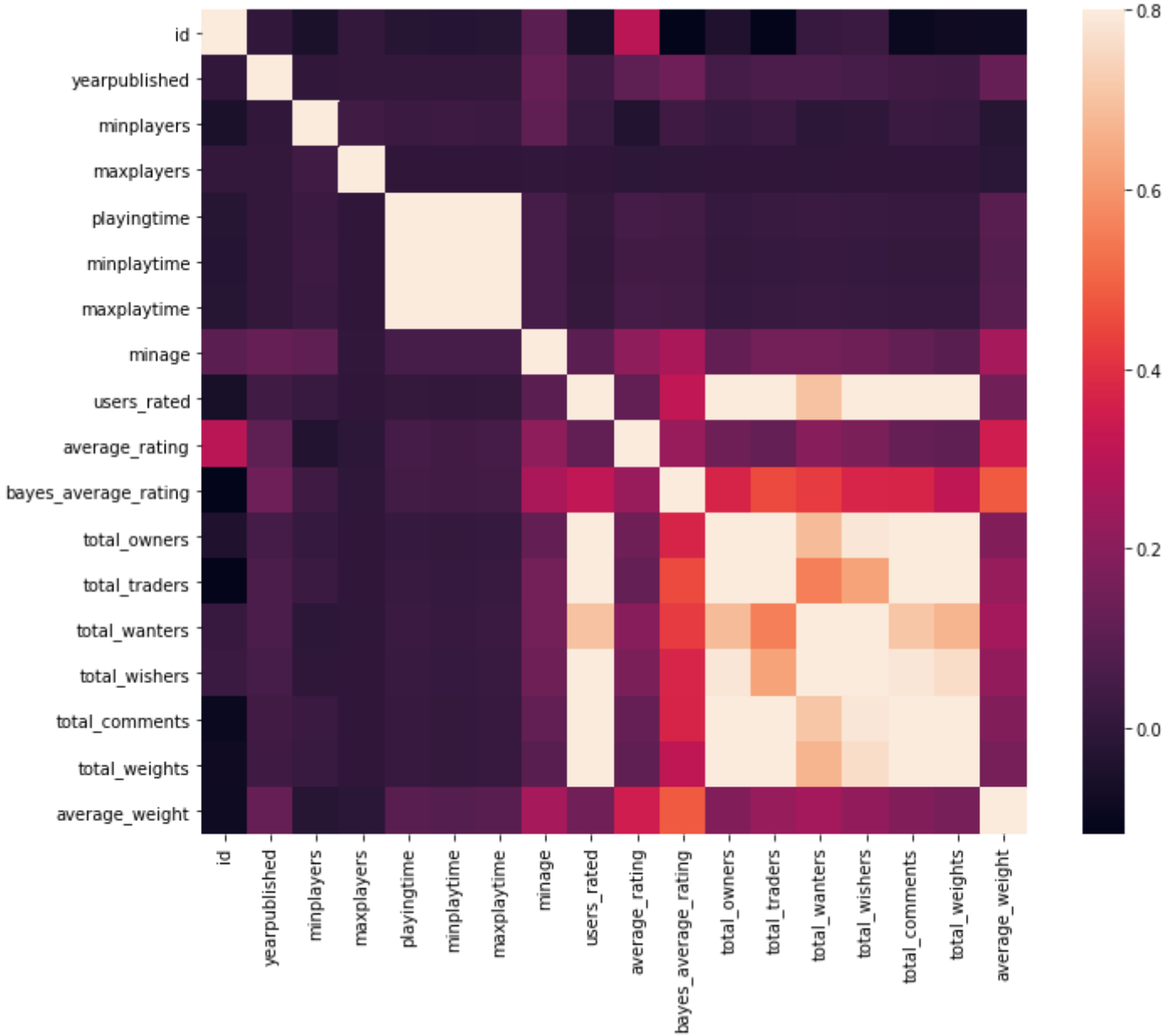



In [8]:
```
print(games.columns)
```

```
Index(['id', 'type', 'name', 'yearpublished', 'minplayers', 'maxplayers',
       'playingtime', 'minplaytime', 'maxplaytime', 'minage', 'users_rated',
       'average_rating', 'bayes_average_rating', 'total_owners',
       'total_traders', 'total_wanters', 'total_wishers', 'total_comments',
       'total_weights', 'average_weight'],
      dtype='object')
```

In [10]:

```
#correlation matrix

corrmat = games.corr()

fig = plt.figure(figsize = (12,9))

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

plt.show()
```


id
yearpublished
minplayers
maxplayers
playingtime
minplaytime
maxplaytime
minage
users_rated
average_rating
bayes_average_rating
total_owners
total_traders
total_wanters
total_wishers
total_comments
total_weights
average_weight
0.8
0.6
0.4
0.2
0.0

In [11]:
```
#part od dataset preprocessing
#get all the columns from the dataframe
columns = games.columns.tolist()

#filter the columns to remove data we do not want
columns = [c for c in columns if c not in ['bayes_average_rating','average_rating','type','name','id']]

#store the variable we'll be predicting on
target = 'average_rating'
```

In [16]:
```
#split the dataset and generate training and test datasets
from sklearn.model_selection import train_test_split

#generate traning sets
train = games.sample(frac = 0.8, random_state = 1)

#select anything not in the training set and put it in test dataset
test = games.loc[~games.index.isin(train.index)]

#print shapes
print(train.shape)
print(test.shape)
```

```
(45515, 20)
(11379, 20)
```

In [17]:
```
#import linear regression model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

#initialize the model class
LR = LinearRegression()

#fit the data in the model
LR.fit(train[columns], train[target])
```

Out[17]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [18]:
```
#generate predictions for test dataset
predictions = LR.predict(test[columns])

#compute error b/w test predictions and actual values
mean_squared_error(predictions, test[target])
```

Out[18]:
```
2.078819032629324
```

In [19]:
```
#import the random forest model
from sklearn.ensemble import RandomForestRegressor

#initialize the model
RFR = RandomForestRegressor(n_estimators = 100, min_samples_leaf = 10, random_state = 1)

#fit the model to the data
RFR.fit(train[columns], train[target])
```

Out[19]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=10, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=100,
                      n_jobs=None, oob_score=False, random_state=1, verbose=0,
                      warm_start=False)
```

In [23]:
```
#make predictions
predictions = RFR.predict(test[columns])

#compute error
mean_squared_error(predictions, test[target])
```

Out[23]:
```
1.4458560046071653
```

In [25]:
```
test[columns].iloc[0]
```

Out[25]:
```
yearpublished      2011.0000
minplayers            2.0000
maxplayers            6.0000
playingtime         200.0000
minplaytime          60.0000
maxplaytime         200.0000
minage               14.0000
users_rated       15709.0000
total_owners      17611.0000
total_traders       273.0000
total_wanters      1108.0000
total_wishers      5581.0000
total_comments     3188.0000
total_weights      1486.0000
average_weight        3.6359
Name: 9, dtype: float64
```

In [26]:
```
#predictions
rating_LR = LR.predict(test[columns].iloc[0].values.reshape(1, -1))
rating_RFR = RFR.predict(test[columns].iloc[0].values.reshape(1, -1))

#print the predictions
print(rating_LR)
print(rating_RFR)
```

```
[8.12061283]
[7.91373581]
```

In [27]:
```
test[target].iloc[0]
```

Out[27]:
```
8.07933
```