In [56]:

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

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

```
3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]
1.0.1
3.1.3
0.22.1
0.10.0
```

In [57]:

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

In [58]:

```
# Read in the data.
games = pandas.read_csv("games.csv")
```

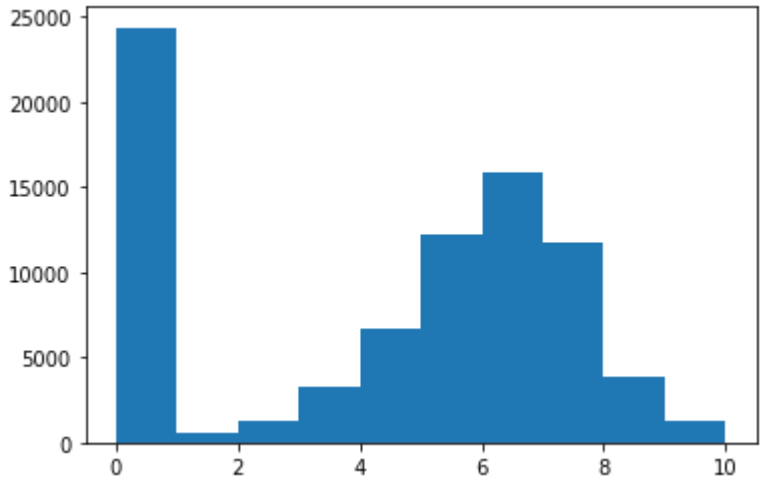
In [59]:

```
# Print the names of the columns in games.
print(games.columns)
print(games.shape)
```

```
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')
(81312, 20)
```

In [60]:

```
# Make a histogram of all the ratings in the average_rating column.
plt.hist(games["average_rating"])
# Show the plot.
plt.show()
```



In [61]:

```
# 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 0.
print(games[games["average_rating"] > 0].iloc[0])
```

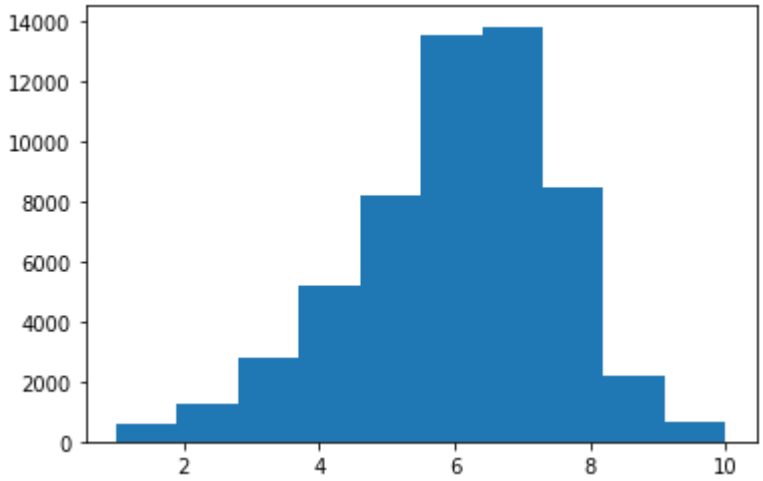
```
id                              318
type                      boardgame
name                     Looney Leo
yearpublished                     0
minplayers                        0
maxplayers                        0
playingtime                       0
minplaytime                       0
maxplaytime                       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 [62]:

```
# Remove any rows without user reviews.
games = games[games["users_rated"] > 0]
# Remove any rows with missing values.
games = games.dropna(axis=0)
```

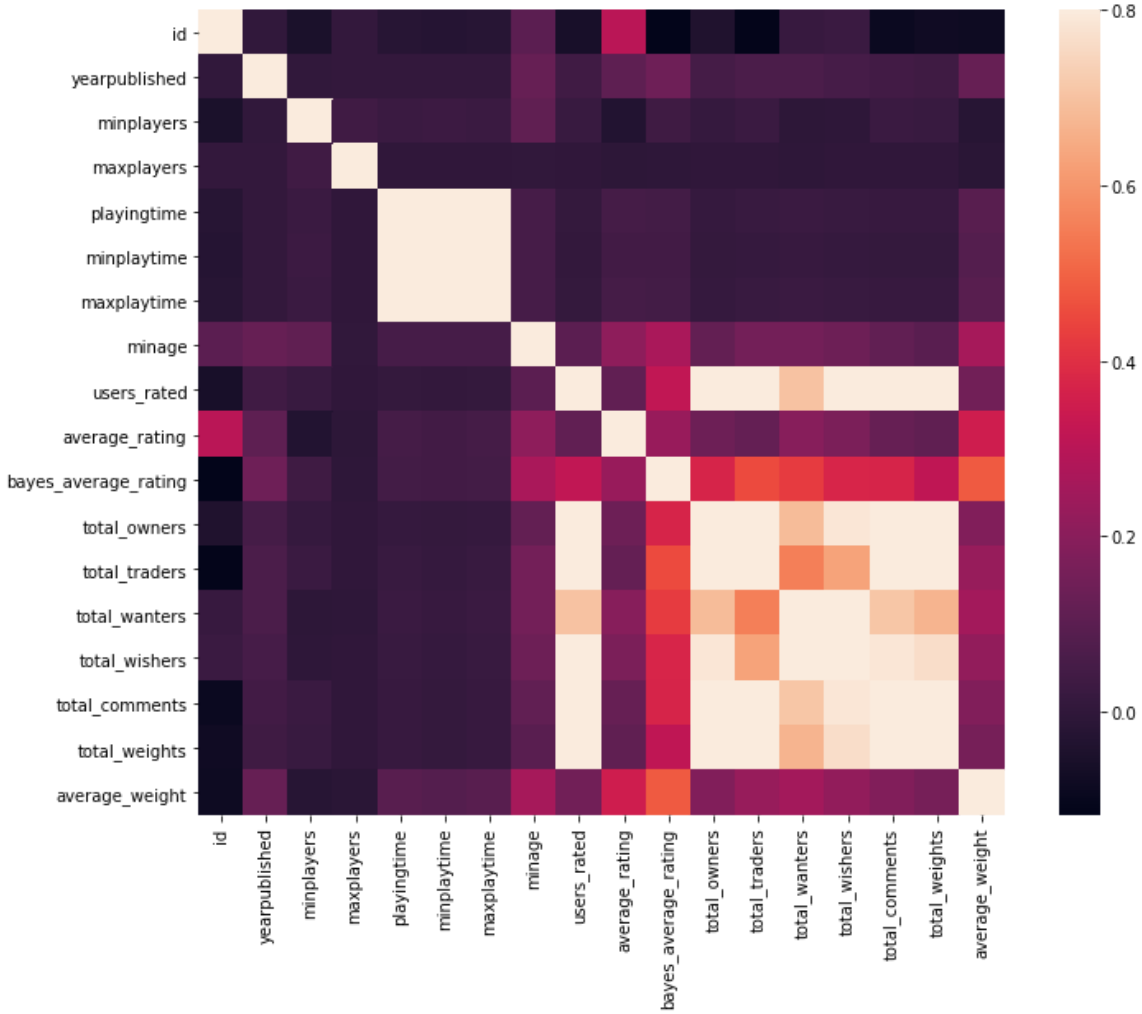
In [63]:

```
plt.hist(games["average_rating"])
plt.show()
```



In [64]:

```
#correlation matrix
corrmat = games.corr()
fig = plt.figure(figsize = (12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
plt.show()
```



In [80]:

```
columns = games.columns.tolist()
columns = [c for c in columns if c not in ["id", "name", "type", "average_rating", "bay
es_average_rating"]]
print(columns)
target="average_rating"
```

```
['yearpublished', 'minplayers', 'maxplayers', 'playingtime', 'minplaytim
e', 'maxplaytime', 'minage', 'users_rated', 'total_owners', 'total_trader
s', 'total_wanters', 'total_wishers', 'total_comments', 'total_weights',
'average_weight']
```

In [81]:

```
from sklearn.model_selection import train_test_split
train = games.sample(frac=0.8,random_state=1)
test = games.loc[(~games.index.isin(train.index))]
print(train.shape)
print(test.shape)
```

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

In [82]:

```
# Import the linear regression model.
from sklearn.linear_model import LinearRegression

# Initialize the model class.
LR=LinearRegression()

# Fit the model to the training data.
LR.fit(train[columns],train[target])

# Import the scikit-learn function to compute error.
from sklearn.metrics import mean_squared_error

# Generate our predictions for the test set.
predictions=LR.predict(test[columns])

# Compute error between our test predictions and the actual values.
mean_squared_error(predictions, test[target])
```

Out[82]:

```
2.0788190326293243
```

In [83]:

```
# Import the random forest model.
from sklearn.ensemble import RandomForestRegressor

# Initialize the model with some parameters.
RF = RandomForestRegressor(n_estimators=100, min_samples_leaf=10, random_state=1)
# Fit the model to the data.
RF.fit(train[columns], train[target])
# Make predictions.
predictions = RF.predict(test[columns])
# Compute the error.
mean_squared_error(predictions, test[target])
```

Out[83]:

```
1.4458560046071653
```

In [84]:

```
test[columns].iloc[0]
test[target].iloc[0]
```

Out[84]:

```
8.07933
```

In [85]:

```
rating_LR=LR.predict(test[columns].iloc[0].values.reshape(1,-1))
rating_RF=RF.predict(test[columns].iloc[0].values.reshape(1,-1))
print(rating_LR)
print(rating_RF)
```

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