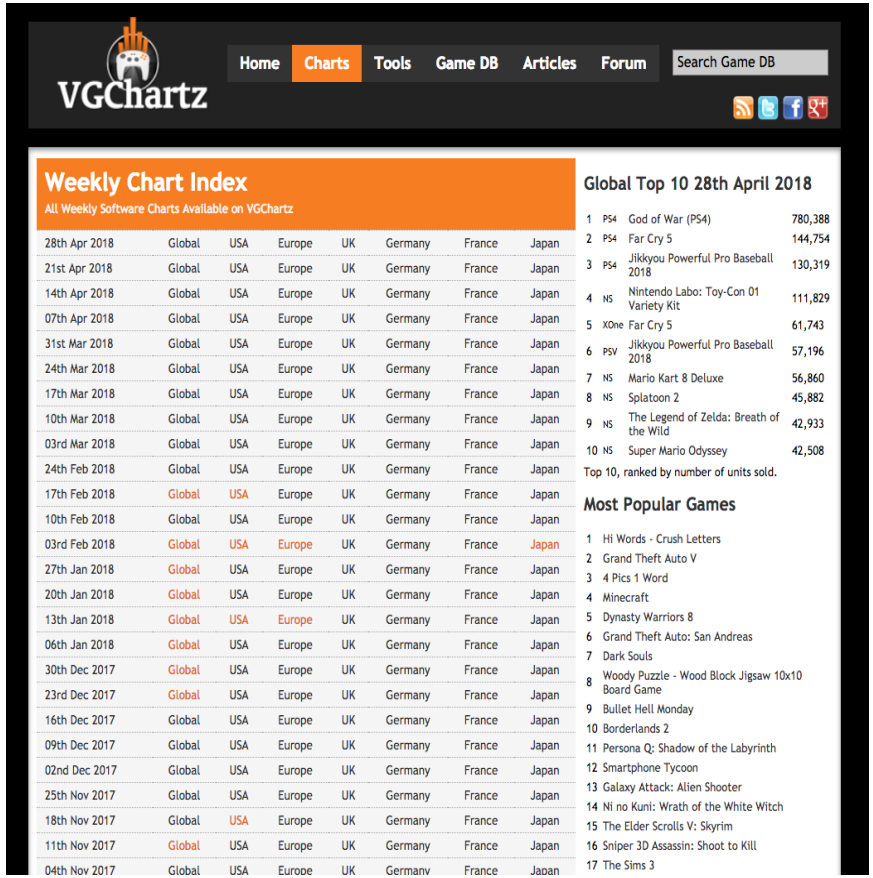
How Many Videogames Will You Sell?

Sarah Floris

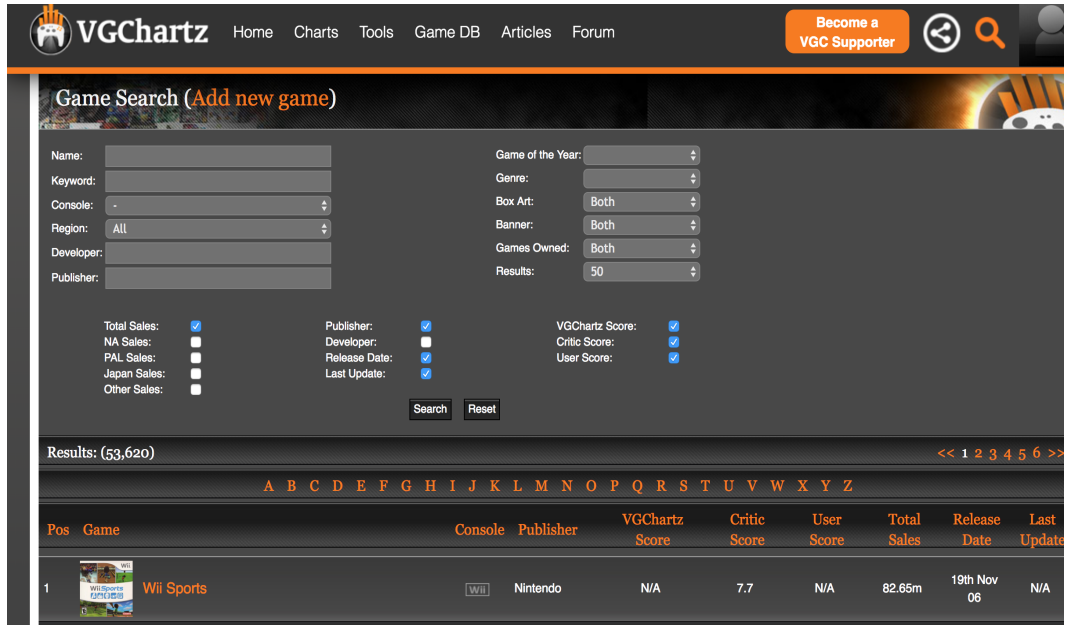
Things to Note

1. Tweets are not differentiated by consoles.
2. The maximum number of tweets is 500 per week.
3. Weekly sales only provides the top 30 videogames.
4. Total sales includes all videogames provided by the VGChartz database.
5. Europe includes United Kingdom, Germany , and France
6. Global population is the population of United States, Japan, and Europe.

Obtaining Data: VGChartz



Right: Online Database featuring, game, console, publisher, VGChartz score, critic score, user score, total sales, release date, and last update



Left : Photo of the weekly sales database, listing weekly sales for global, United States, Europe, Germany , France, and Japan population.

Obtaining Data: Twitter

Main problems with obtaining large set of tweets:

1. RESTful API only does historical tweets for the last 7 days
2. Only lets us query a limited amount (up to 1000) for free
3. Access to historical data if you want to pay for Enterprise

How do we solve these problems?

1. Set a tweet's class attributes to formulate an url (such as video game title, date ranges, etc.)
2. Use this url to web scrape the tweets that contain the videogame title both in the body and in the hashtag
3. Keep searching until max number of tweets has been reached

Obtaining Data: Twitter

Web scraped the tweets that contain the videogame hashtag from week to week after searching

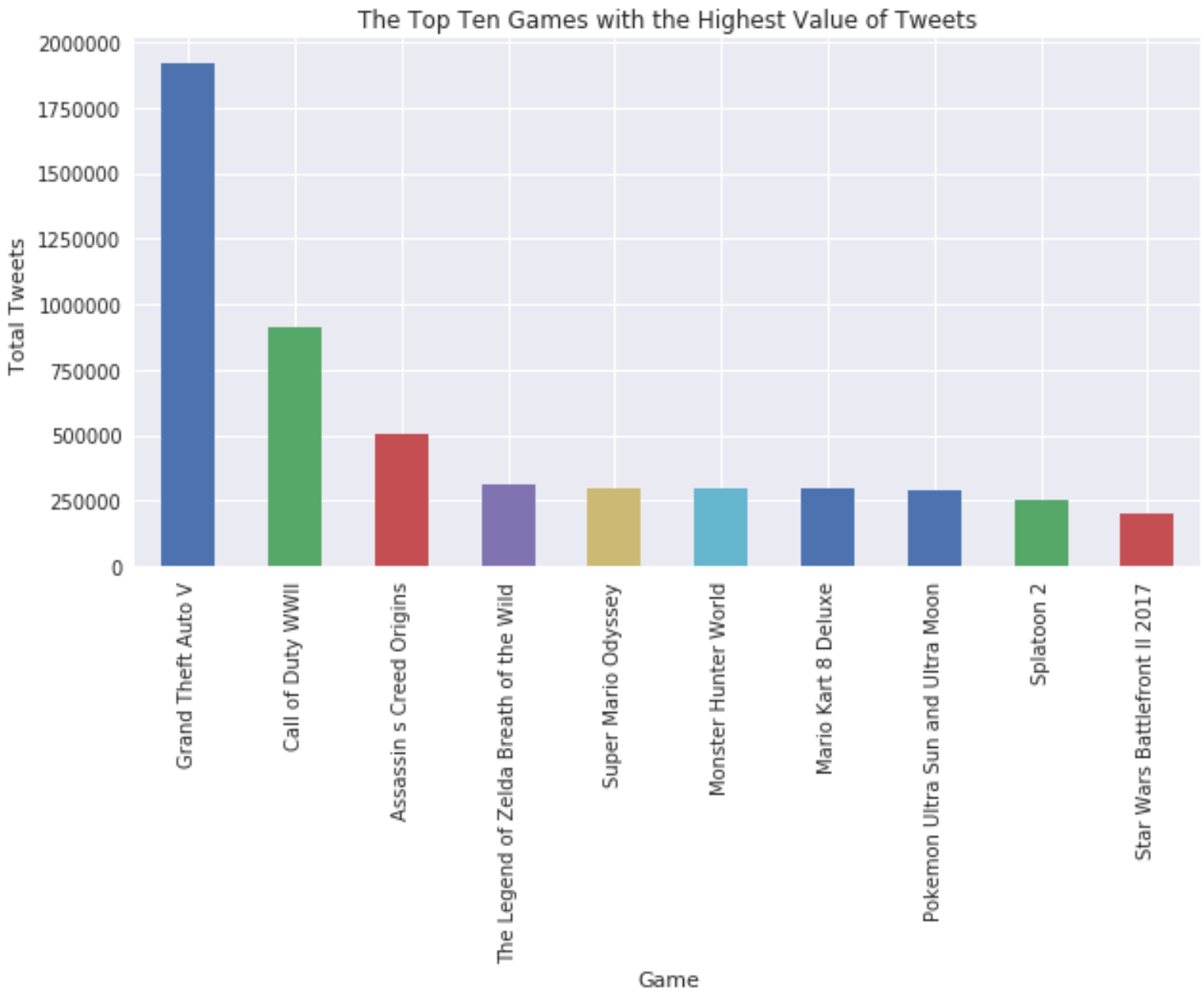
Example

```
def finding_tweets(videogame):
    """Description"""
    name = []
    start_date = []
    tweet_date = []
    tweet_text = []
    assert isinstance(videogame[0],str)
    assert isinstance(videogame[1],str)
    assert isinstance(videogame[2],str)
    try:
        tweetCriteria = TweetCriteria().setQuerySearch(videogame[0]) \
.setSince(videogame[1]).setUntil(videogame[2]).setMaxTweets(500)
        tweets = TweetManager.getTweets(tweetCriteria)
        for tweet in tweets:
            tweet_date.append(tweet.date)
            tweet_text.append(tweet.text)
    except:
        print(videogame[0],videogame[1],videogame[2], 'empty')
    df = pd.DataFrame(np.column_stack((tweet_date,tweet_text)))
    df['Name']= videogame[0]
    df['start_date'] = videogame[1]
    df['end_date'] = videogame[2]
    return df
```

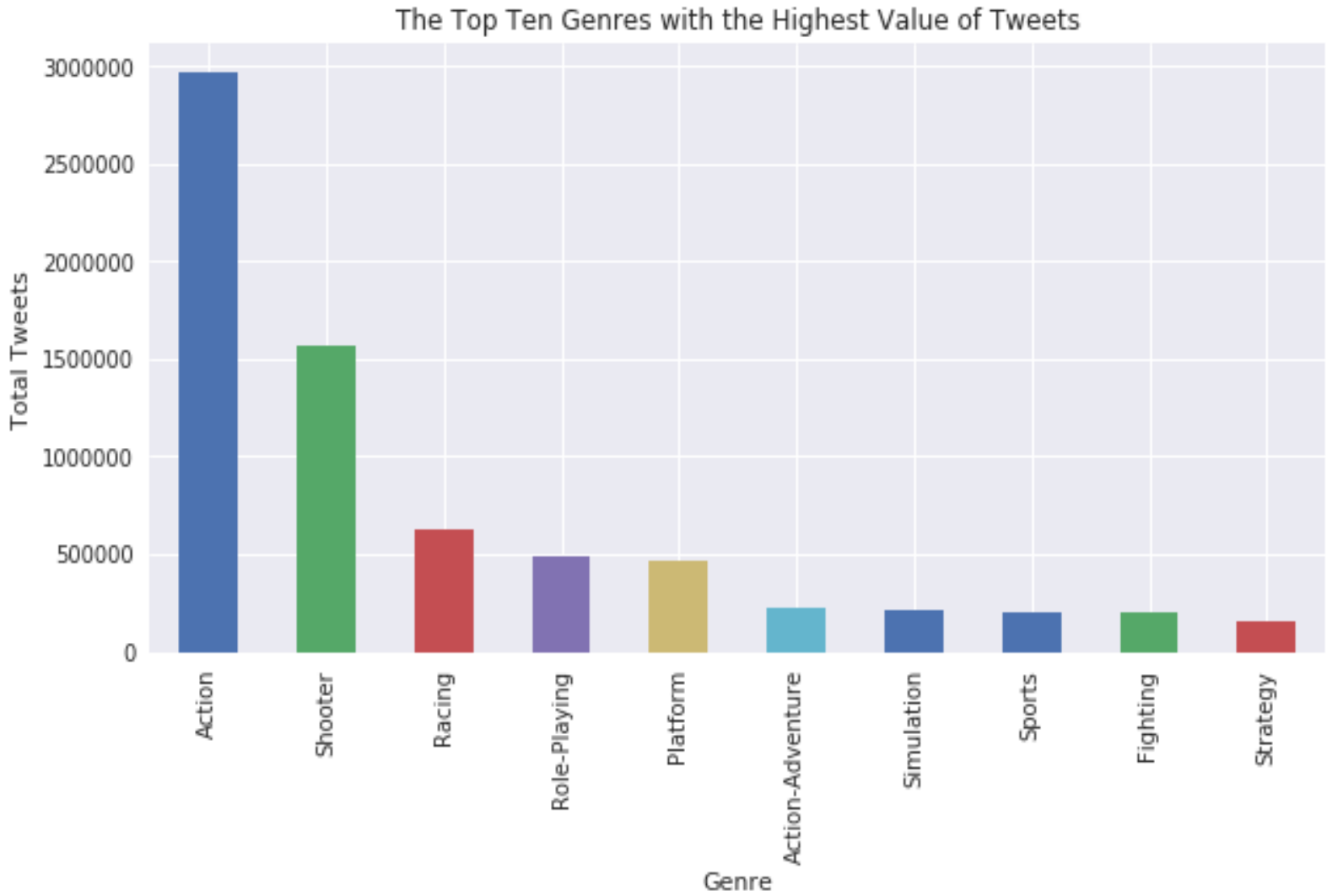
Interested in seeing all of the code? Look at my github!

Let's take a look at the data

Top Ten Games

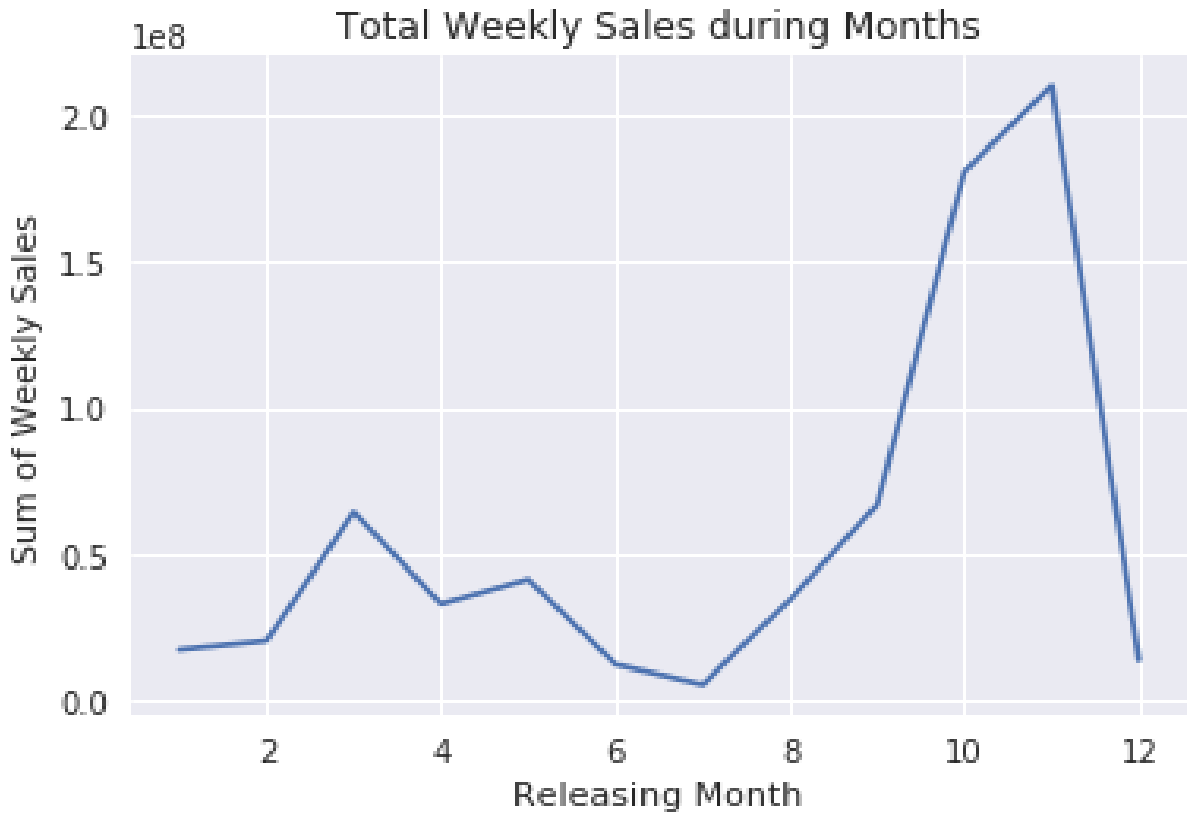


Top Ten Genres

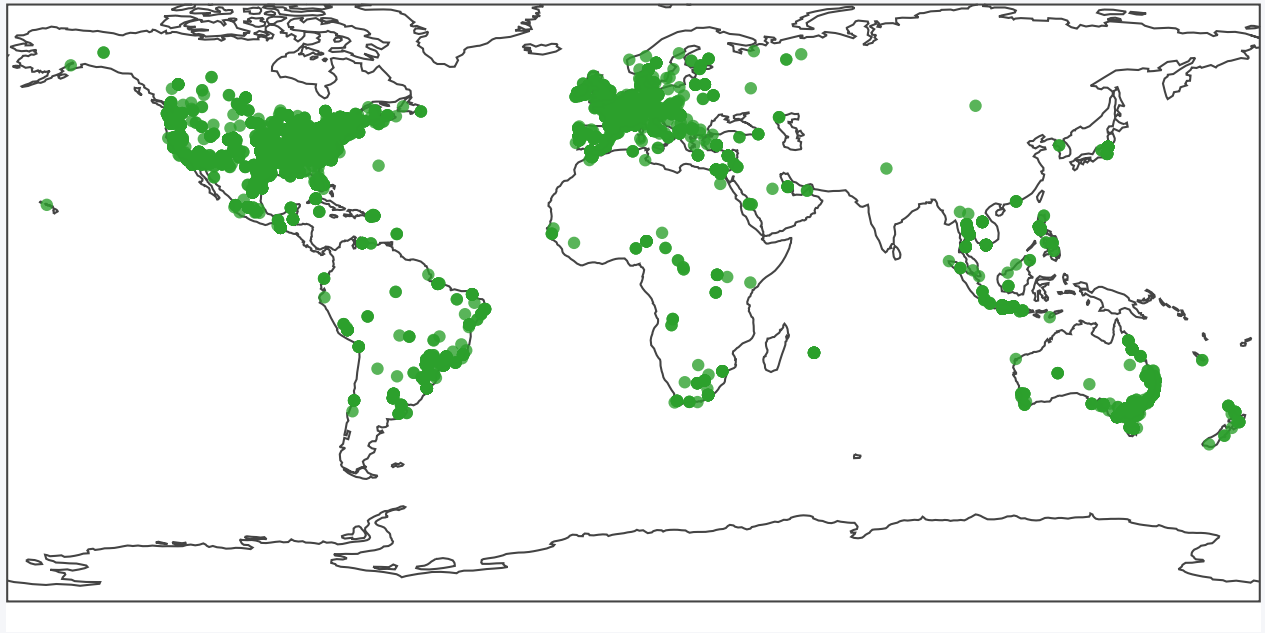


Monthly Game Unit Sales and Total Sales

Month	Number of Games
1	519
2	1138
3	2525
4	1491
5	1554
6	730
7	343
8	1134
9	1596
10	4748
11	5223
12	892



Videogame Publisher's Location



Predicting the next weeks sales

Machine Learning Methods

- Linear Regression - 0.198
- Elastic Net - 0.160
- Random Forest Regression - 0.035
- GradientBoostingregressor - 0.187
- AdaBoostRegressor with RandomForest Regression as initial estimator - 0.384

The best machine learning method was AdaBoostRegressor with a RandomForestRegressor as a base estimator.
Test set accuracy score for best params: 0.375

```
AdaBoostRegressor(base_estimator=RandomForestRegressor(bootstrap=True, criterion='mse',
max_depth=None, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0,
min_impurity_split=None, min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=9, n_jobs=-1, oob_score=False,
random_state=None, verbose=0, warm_start=False), learning_rate=1.0,
loss='linear', n_estimators=50, random_state=None)
```

begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction.

Predictions for the Top 30 Games Sold