Title: IMDb Web Scraping and Analysis with Selenium Python

Authors:
Sophia Tsilerides, smt570
Ilana Weinstein, igw212
Amanda Kuznecov, anr431

Abstract: This project focuses on the elements that comprise a critically acclaimed film as defined by IMDb's Top 250 Movie list. We approach this problem using web scraping and web-based automation tools in Python. Our results show that this is an effective and efficient way to collect data which visualizations can be made from to draw conclusions.

1 Introduction and Motivation

In order to put tools from class into practice, we have gathered data from the IMDb website to analyze and observe trends in the movie industry, with the goal of creating meaningful visualizations. We chose to use data from IMDb, an online database containing information about movies, because we share a common interest in movies and were excited to work with the information in this dataset. A great deal of information about movies is presented on IMDb's website, however, it is not presented in a way where meaningful insights can be visualized.

The Top 250 Movie list on IMDb's website was specifically chosen for further analysis as it was very well structured. Website scraping was automated for efficiency and reusability using Selenium. To maximize value, we merged the IMDb dataset with another dataset containing biographical information of actors. Upon scraping IMDb and merging with the other dataset, data cleaning and inspection were required before continuing any work. By creating an aggregated dataset, we were able to analyze correlations between movie genres, ratings, revenue, and actors' age and gender.

2 Methodology

For the duration of the project, we used Jupyter Notebook to write code in Python, and collaborated on our notebooks through git. Our repo can be found here: https://github.com/akuz91/DS-GA-1007-Project.

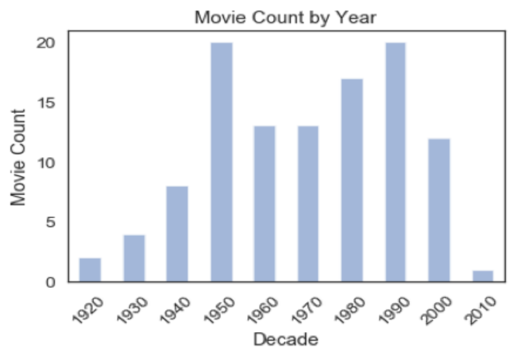
Web scraping was utilized to extract the top 250 movie information from the IMDb website as there was no direct way to download it [1]. We used Python's BeautifulSoup module, which is a package for parsing HTML documents [2]. We began by importing necessary packages, including the *urlib.request* module to open the URLs. Since the web pages were in HTML, we learned the structure by right-clicking on the page and selecting "Inspect". By hovering over the different HTML tags, we were able to determine which features the HTML tags were referencing and subsequently identified these while writing the code to web scrape [3]. Using the *soup.find_all* function from BeautifulSoup, features extracted included movie title, year, rating, duration, genres, stars, Metascore, votes, revenue, director, and lead actor. Each feature was extracted in separate lists and then converted to a single data frame for manipulation. HTML tags represented single lines on the web page which were repeated in a similar fashion for each movie. For example, by referencing the HTML tag *'h3',{'class': 'lister-item-header'}*, the movie title and year for each movie on the page were extracted by using a for-loop to find all applicable items depending on their location within the line of text.

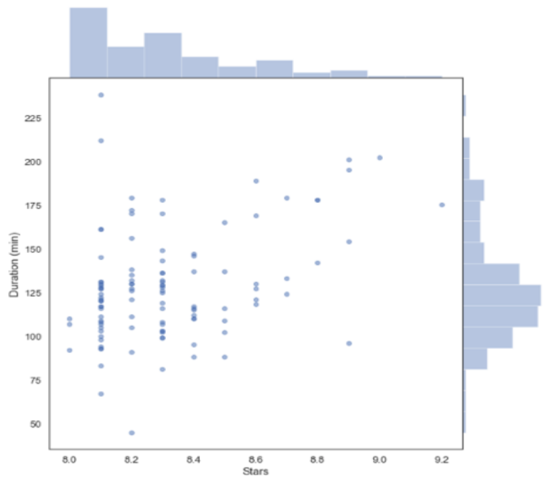
As only 50 movies appeared on each webpage, the best way to extract all 250 movies was to click through each page and scrape the current page. To do this seamlessly we used Selenium, an automated testing suite for web applications. Our code opens the browser to the first page of the IMDB list, scrapes the data of the current page, finds and clicks the "next" page and continues this cycle until the last page. Within the code is a function that scrapes the current driver page source to a data frame, resulting in a master data frame with information from each page. An important design decision in the scraping function was to incorporate code that handles movies that do not contain a rating, offsetting data for the remaining features of the movie instance. This design decision improved continuity and decreased the need for data cleaning.

To complete the dataset, the web scraped data from IMDb was merged with an actor information dataset [4]. This dataset included biographical information such as actor name, gender, date of birth, and role type. Some instances within the dataset were missing birth years, so we manually checked Wikipedia for these missing values. The birth year column was also used to create a new column representing the actor's age at the time the movie was released.

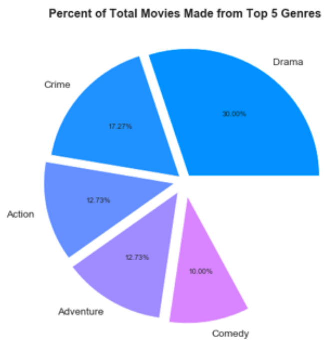
3 Results

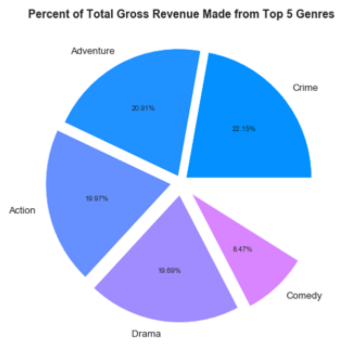
In order to understand our results, we developed a number of visualizations. Below are some of the most interesting discoveries made from this dataset.

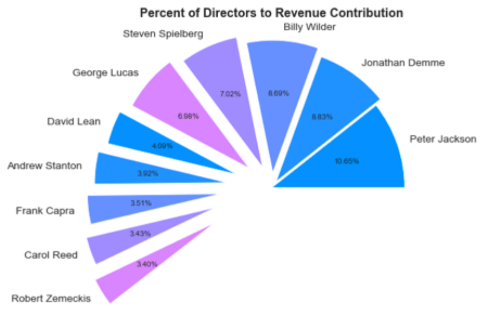




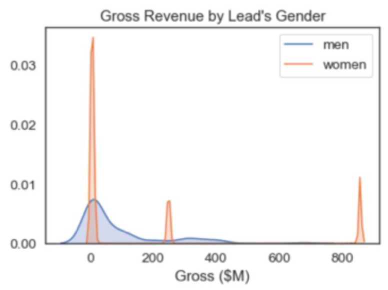
Binning by decade, we can see the best movies were made in the 1950s and 1990s as shown in the bar chart above. Between the two decades, we saw 40 of the best movies as provided by this dataset that make up 16% of our data. To the right, the scatter plot with appended histograms show the correlation between the number of stars a movie has and how long the movie is. There is a positive correlation between the two features, although one must keep in mind our dataset is biased to include only the best movies. In reality, we hypothesize the opposite correlation is true and these movies would be considered outliers within the total population of movies. The histograms also show us that the distribution of movie duration is almost normal with a left skewed tail, and the majority of the movies have 8 stars.

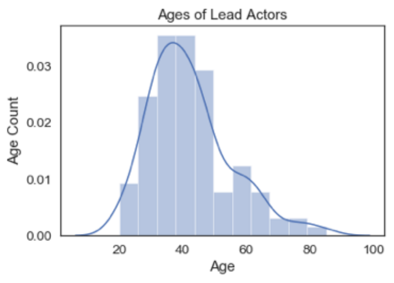


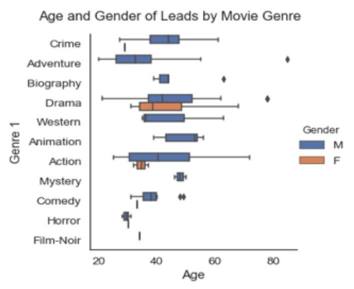




Next, we wanted to better understand the factions of our data. This is best done by pie charts as shown above. The leftmost pie chart shows drama, crime, and action movies are the most popular genres in our dataset, however it is adventure movies that competes with crime movies to produce the highest grossing genres as shown in the middle chart. The chart on the right presents the 10 directors that contributed the most to total gross revenue. On the list is Peter Jackson who directs the Lord of the Rings movies, Jonathan Demme who directed The Silence of the Lambs and Billy Wilder, the director of Witness for the Prosecution and Sunset Blvd. Jackson's gross for three movies compares to one of Demme's and two of Wilder's.







Lastly, we were interested in how gender and age played a role in our dataset. While female leads bring in more money, there are less female leads than male leads as shown by the width of distributions in the leftmost density plot. In fact, our whole dataset only saw nine female leads. The middle histogram shows the distribution of the ages of leads in our dataset which is roughly normal with a right tail. The oldest lead actor was William Holden in The Bridge on the River Kwai at 85 years old, and the youngest actor in our dataset was Elijah Wood in Lord of the Rings at 20 years old. Finally, interesting results were produced when we compared age and gender with genre in our dataset. Females were casted in lead roles for four dramas, two action movies, and one comedy, crime and horror movie. Additionally, from the age distribution, it was concluded that females were much younger than the men casted for the same genre, as shown by the empirical median of our box plots. The mean age of women in our dataset is 37 while the mean age of men is 42.

4 Discussion

While web scraping is a useful tool and is used by many companies to drive business decisions, it is not reproducible. The code we wrote to scrape data is only applicable to the specific IMDb list and webpages for that list. Web scraping produced some errors such as random characters in the year column, which was addressed with manual treatment and automation with Selenium.

Several other issues came up during the extraction process, where some features required conversion to a format that could be utilized in our analysis. For example, the duration feature was extracted as a string containing the length of the movie and the word *"min"*. We converted the string to an integer and stripped *"min"* from the variable. Additionally, when the movie years were extracted, the output was a numbered list with each year enclosed by parentheses. To produce a clean list, Pandas replace functions were utilized to grab important year information. Some movies listed multiple genres (up to three), which we split on and created new columns to represent individual genres. Similarly, some instances had part of their title included in the Year column and were removed with Pandas replace method. Metascore was also difficult to extract as the output list contained multiple parentheses and line breaks between each

Although this is not favorable in best practice, for our purposes this data cleaning helped us detect and mitigate issues. It is not recommended that the web scraping tools used be applied to larger datasets where it is likely that more errors will occur with the mapping process. It is recommended to use portions of the project that can apply to other websites such as the method used to loop through the web pages.

5 Conclusion

Using Python's BeautifulSoup module and the automation of Selenium Python, we were able to analyze the Top 250 Movie list as presented by IMDb.com in conjunction with a database of actors' data. Conclusions drawn from this process are that movie genres that perform well in the box office include drama, crime and action movies, however adventure movies contribute largely to total gross revenue percent. Additionally, women are historically not cast in lead roles, and when they are, their ages are, on average, less than that of men in the same genre. Our analysis can be used by movie goers and aspiring directors alike to see what elements make up the most highly rated movies of the century.

References:

[1] "IMDb 'Top 250.'" IMDb, IMDb.com, www.imdb.com/search/title/?groups=top_250&amp;sort=user_rating.
[2] "Beautiful Soup Documentation." Beautiful Soup Documentation - Beautiful Soup 4.4.0 Documentation, www.crummy.com/software/BeautifulSoup/bs4/doc/.
[3] Intro to Web Scraping with Python and Beautiful Soup, Data Science Dojo, 6 Jan. 2017, www.youtube.com/watch?v=XQgXKtPSzUI&amp;list=PL8eNk_zTBST-SaABhXwBFbKvvA0tlRSRV.
[4] DeathReaper0965. "DeathReaper0965/Movie_ratings_prediction." GitHub, github.com/DeathReaper0965/Movie_ratings_prediction.