Analysis and Recommendation of Movie Types based on user Preferences

Python project

Programming with Advanced Computer Languages: Course project

Johann Mesot: 15-407-927

Alexandre Tissières: 13-607-965

Patrick Muturi: 18-604-793

Lecturer: Dr. Mario Silic

Introduction :

The rise of big data and advanced analytics has allowed companies such as Netflix, Youtube or Facebook to thrive by better matching the needs of their target groups. More generally, it has allowed scientists and researchers to use and recombine data at will to generate interesting insights and applications for a wide range of industries.

The goal of this project is to recommend a movie to users based on their preferred criteria. Our program allows users to choose a potential movie based on a wide range of factors such as average rating, number of rating, movie genre or date. To meet this purpose, we make use of two datasets provided by the movie recommendation service movie lens, which is affiliated to GroupLens Research, a research lab in the department of computer science and engineering at the University of Minnesota.

Summary statistics of the dataset:

Before modification, the dataset originally contained, 100004 ratings and 1296 tag applications across 9125 movies. The data was generated by 671 users between January 09, 1995 and October 16, 2016 (Harper & Konstan, 2015).

Our two sub-datasets, "movies" and "ratings" are made of 3 (movieid, title (containing title and date), genres) and 4 columns respectively (userid, movieid, rating, timestamp).

Brief summary of the project:

Data cleaning:

We first clean and restructure the data in order to delete outlier movies which have no title or who are lacking other key inputs such as the date. We also create a new separate column for the movie date which was grouped together with the title of the movie, making it difficult to analyze both criteria separately. We then proceed to remove the outlier rows which do not equal the "19xx" or "20xx" type. These rows contained erroneous dates or no date which prevented us from successfully creating and implementing our functions.

Visualization of descriptive statistics:

In order to have a clear view of the data at hand, we also perform an analysis of the descriptive statistics of various aspects of the data. We compute and plot the average rating of each genre, the number of movies per genre as well as the average rating per year.

Functions:

1st function: Searching for a movie

The goal of this first function is to allow the user to search for a movie by title. The challenge was to merge the two sub-datasets (movies and ratings), which both contained the movie ID variable. We made use of a wide range of functions and loops to merge the datasets and recombine them according to our needs such as "if", "is in", "merge" and "group by".

We also account for the fact that the user could have misspelled the name of the movie or that the movie is not in the dataset. In that case the function would return the message "Sorry, we could not find what you're looking for! Make sure you spelled it correctly".

2nd function: Filtering movies according to user preferences

In the second function, we further develop our code so that we can find movies that fit our particular preferences such as a range of years, a particular genre, a minimum number of times it was rated, or a minimum average rating given by the users. Please note that the movies do not go beyond the year 2016. Similarly to the first function we also make use of a wide range of functions and loops to merge the datasets and recombine them according to our needs and we also account for a potential error or false input on the part of the user by displaying an error message.

3rd function: Plotting movies according to our preferences and visualizing data

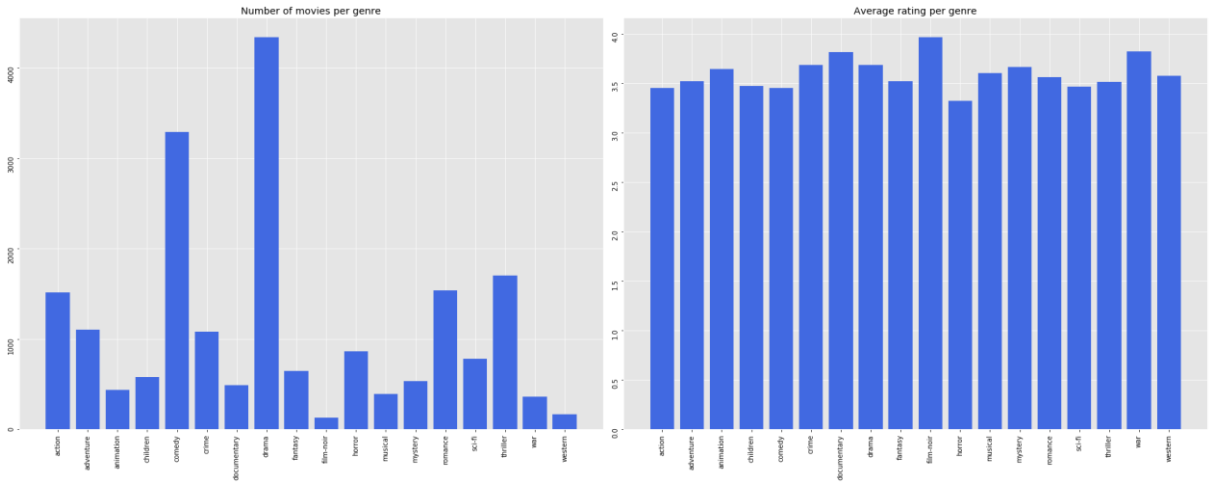
Now that we have both the ability to search for a movie by title and to find them according to certain preferred criteria (years of production, genre, minimum number of times it was rated, minimum average rating given), we create a function, which is able to plot the movies according to the aforementioned criteria. The user would ideally choose a genre and a year. The corresponding movies would then be displayed in a three-dimensional plot according to their names, average rating and number of ratings per movie.

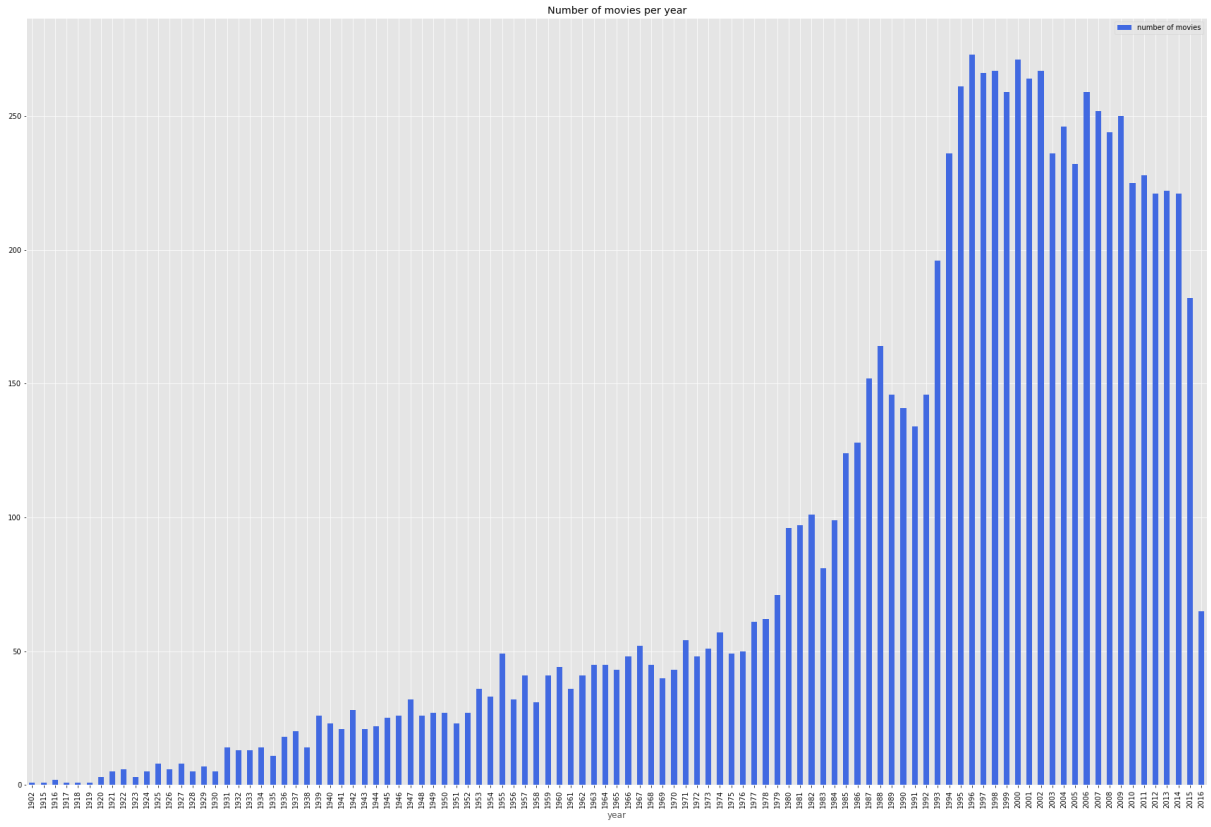
User interaction (exemplary user inputs in bold)

Now that the functions have been defined and the descriptive statistics plotted, we introduce user interactions. We do this through a while loop combined to various if loops in order to selectively take our previous functions into account according to users preferences

4 potential user inputs are available for the user to choose

[1] Get information about the dataset: *"displays and plots a wide range of summary statistics"*





[2] Search for a movie by title: *"Searches for a title and returns the movie title, production year, average rating and number of ratings"*

What movie are you looking for? **jumanji**
Results for jumanji

movieId	Movie	year	average rating	number of ratings
2	jumanji	1995	3.4	107

[3] Find movies according to certain criteria

What period of time are you interest in? Enter start and end year separated by a comm
a: **1990,1993**
What genre are you looking for?:
Genres available: action, adventure, animation, children's, comedy, crime, documentar
y, drama, fantasy, film-Noir,
horror, musical, mystery, romance, sci-fi, thriller, war, western
drama
How often should a movie at least be rated? **40**
What is the minimum average rating you want? **4.0**

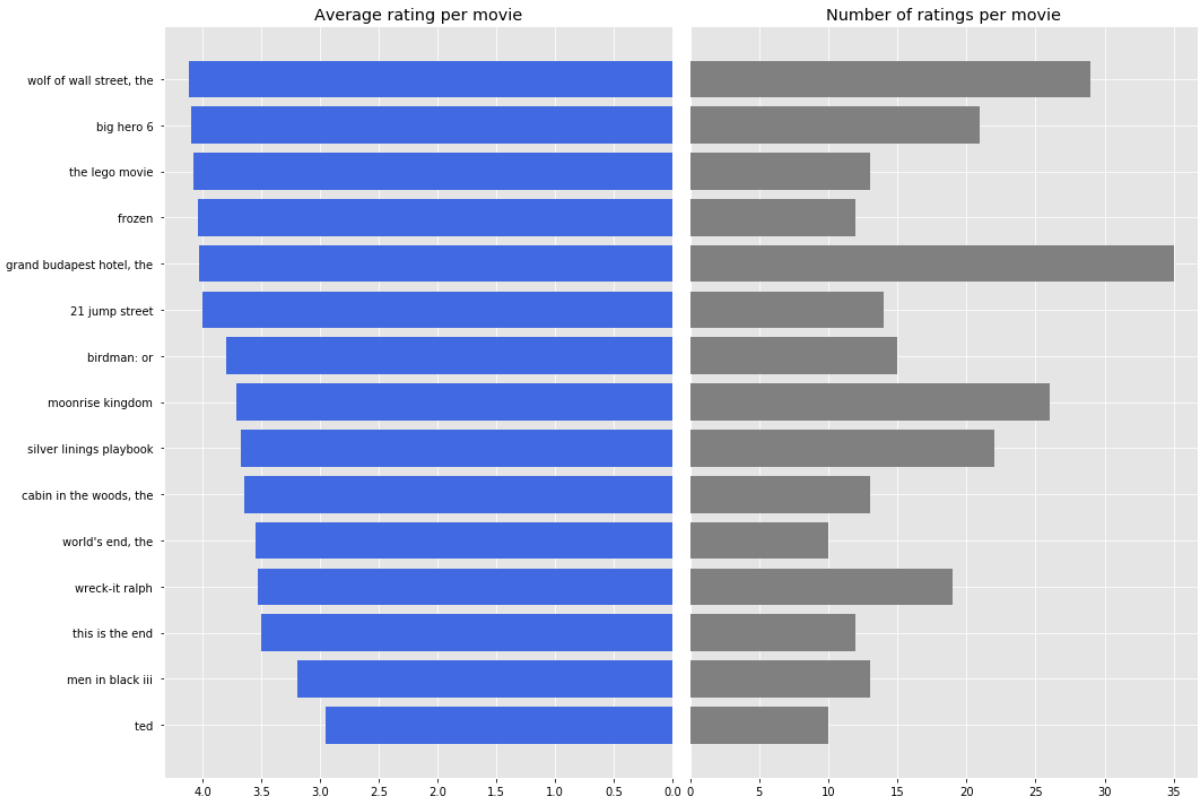
	movieId	movie name	avg rating	nbr of ratings
1	527	schindler's list	4.30	244
2	1213	goodfellas	4.20	131
3	2289	player, the	4.20	44

0 515 remains of the day, the 4.04 46

[4] Plot average ratings and number of ratings according to certain criteria *"plots the top 15 rated movies in the chosen genre and time period" (assuming at least 15 are available, please note that some movies have several ratings assigned to them)*

What period of time are you interest in? Enter start and end year separated by a comm
a: **2012,2014**
What genre are you looking for?:
Genres available: action, adventure, animation, children's, comedy, crime, documentar
y, drama, fantasy, film-Noir,
horror, musical, mystery, romance, sci-fi, thriller, war, western
comedy

Top 15 movies in the comedy genre between 2012 and 2014



[5] Exit

Do you want to try again? [y] or [n]**n**
Do you want to end the program? [y] or [n]**y**

At the end of each action, we ask the user if he/she wants to try again (yes or no) (example: look for another movie within the same function) or if he/she wants to go back to the main menu of 4 options. If he/she choses no, the user will be asked if she wants the program to end or if we she wants to perform another function (yes or no)

Code:

For reference, we attach our full code below

```
# Import packages
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import numpy as np
from scipy.sparse import csr_matrix
import warnings; warnings.simplefilter('ignore')

print('imported successfully')

# Funtion 0: Creating extra column for year of movies and clean data
def cleaning_dataset(movies):
    new = movies["title"].str.split("(", n = 1, expand = True)
    movies["movie title"]= new[0].str.lower()
    movies["year"]= new[1]
    movies["genres"] = movies["genres"].str.lower()
    movies["year"].replace(regex=True,inplace=True,to_replace=r'\D',value=r'')
    movies.drop(columns =["title"], inplace = True)
    movies = movies[['movieId', 'movie title', 'year', 'genres']]
    movies.dropna(inplace = True)

    #Create a column with the length of years in each row
    movies["year length"]= movies["year"].str.len()
    #Get indexes for which column year's length is not 4
    indexvalues = movies[movies['year length'] != 4 ].index
    #Delete these row indexes from dataFrame
    movies.drop(indexvalues , inplace=True)

    return movies


# First function, allow to get information about the dataset
def statsfunction(ratings, movies):

    #Give general info about the dataset
    print('The dataset contains: ', len(ratings), ' ratings of ', len(movies), ' movies.')

    #Data for first graph: number of movies per genre
    genres = ["action", "adventure", "animation", "children", "comedy", "crime", "documentary",
"drama", "fantasy", "film-noir", "horror", "musical", "mystery", "romance", "sci-fi", "thriller", "war",
"western"]
    number=[]

    for genre in genres:
        count = len(movies[movies['genres'].str.contains(genre)])
        number.append(count)

    d={'genre':genres,'number':number}
    df=pd.DataFrame(d)

    #Data for second graph: average rating per genre
```

```
    average_rating=[]

    for genre in genres:
        genre_movies = movies[movies['genres'].str.contains(genre)]
        avg_genre = ratings[ratings['movieId'].isin(genre_movies['movieId'])].loc[:, 'rating'].mean()
        average_rating.append(round(avg_genre,2))

    e={'genre':genres,'average rating':average_rating}
    table=pd.DataFrame(e)


    #Plot graphs 1 and 2 together:
    fig, axes = plt.subplots(ncols=2, sharey=False, figsize=(25,10))
    axes[0].bar(df['genre'], df['number'], align='center', color='royalblue', zorder=10)
    axes[0].set(title='Number of movies per genre')
    axes[1].bar(table['genre'], table['average rating'], align='center', color='royalblue', zorder=10)
    axes[1].set(title='Average rating per genre')

    axes[0].tick_params(
        axis='both',
        colors='black',
        labelrotation=90)

    axes[1].tick_params(
        axis='both',
        colors='black',
        labelrotation=90)

    fig.tight_layout()
    fig.subplots_adjust(wspace=0.035)
    plt.show()

    #Print third graph: Number of movies per year
    years=[]
    for x in movies['year']:
        if x not in years:
            years.append(x)
    years.sort()

    movie_count_year=[]
    for year in years:
        count = len(movies[movies['year'].str.contains(year)])
        movie_count_year.append(count)

    f={'year':years,'number of movies':movie_count_year}
    df2=pd.DataFrame(f)
    df2.plot(title='Number of movies per year',kind="bar", rot=90, x='year', y='number of movies',
color='royalblue', figsize=(30,20))

    plt.tick_params(
        axis='both',
        colors='black')

    plt.show()
```

```
# Second function, allow to look for a movie by title
def movie_info(ratings, movies, title):
    #Find Movie info
    search=movies[movies['movie title'].str.contains(title)]
    movie_name=movies[movies['movieId'].isin(search['movieId'])].groupby('movieId')['movie
title'].sum()
    movie_year=movies[movies['movieId'].isin(search['movieId'])].groupby('movieId')['year'].sum()

average_rating=ratings[ratings['movieId'].isin(search['movieId'])].groupby('movieId')['rating'].mean().r
ound(2)

number_of_ratings=ratings[ratings['movieId'].isin(search['movieId'])].groupby('movieId')['rating'].cou
nt()

    #Get output
    output=pd.concat([movie_name, movie_year, average_rating, number_of_ratings,], axis = 1)
    output.columns=['Movie','year', 'avg rating', 'nbr of ratings']
    if output.empty == True:
        return """Sorry we could not find what you're looking for!
Make sure you spelled it correctly!
        """
    else:
        return output

# Third function, allow to find movies that fit criteria (years, genre, avg. rating, number of ratings)

def find_movies(movies, ratings, start, end, genre, times_rated, avg_rating):
    #Filter movies data set for input
    years=movies[(movies['year']>=start) & (movies['year']<=end)]
    genre=years[years['genres'].str.contains(genre)]

    #Find average rating and number of ratings for the filtered movies

average_rating=ratings[ratings['movieId'].isin(genre['movieId'])].groupby('movieId')['rating'].mean().r
ound(2).reset_index()

number_of_ratings=ratings[ratings['movieId'].isin(genre['movieId'])].groupby('movieId')['rating'].coun
t().reset_index()
    rating_frame=average_rating.merge(number_of_ratings, on='movieId')
    rating_frame.columns=('movieId','average','count')

    #Filter for minimum number of ratings and minimum average rating
    minimum_count=rating_frame[rating_frame['count']>=times_rated]
    minimum_average=minimum_count[minimum_count['average']>=avg_rating]

movie_name=movies[movies['movieId'].isin(minimum_average['movieId'])].groupby('movieId')['mov
ie title'].sum()
    output_data=minimum_average.merge(movie_name, on='movieId')
    output_data.columns=('movieId','avg rating','nbr of ratings','movie name')
    output_data = output_data[['movieId','movie name','avg rating','nbr of ratings']]
    output_data = output_data.sort_values(by=['avg rating'], ascending=False)

    if output_data.empty == True:
        return """Sorry we could not find what you're looking for!
```

```
Make sure you spelled it correctly!
        """
    else:
        return output_data[['movie name', 'avg rating', 'nbr of ratings']]

# Fourth function, allow to plot the top 15 movies from the selection:

def plot_movies(movies, ratings, start, end, genre):
    #Filter movies data set for input
    years=movies[(movies['year']>=start) & (movies['year']<=end)]
    genre=years[years['genres'].str.contains(genre)]

    #Find average rating and number of ratings for the filtered movies

average_rating=ratings[ratings['movieId'].isin(genre['movieId'])].groupby('movieId')['rating'].mean().round(2).reset_index()

number_of_ratings=ratings[ratings['movieId'].isin(genre['movieId'])].groupby('movieId')['rating'].count().reset_index()
    rating_frame=average_rating.merge(number_of_ratings, on='movieId')
    rating_frame.columns=('movieId','average','count')

    #Filter for minimum number of ratings and minimum average rating
    minimum_count=rating_frame[rating_frame['count']>=10]

movie_name=movies[movies['movieId'].isin(minimum_count['movieId'])].groupby('movieId')['movie title'].sum()
    #Include movie name in the table
    output_data=minimum_count.merge(movie_name, on='movieId')
    output_data.columns=('movieId','average','count','movie name')
    output_data = output_data[['movieId','movie name','average','count']]
    output_data = output_data.sort_values(by=['average'], ascending=True)
    output_data = output_data.tail(15)

    #Plot the output

    #If the sub-dataframe is empty, show an error message
    if output_data.empty == True:
        return """Sorry we could not find what you're looking for!
Make sure you spelled it correctly!
        """
    #Otherwise, plot the sub-dataframe
    else:
        fig, axes = plt.subplots(ncols=2, sharey=True, figsize=(15,10))
        axes[0].barh(output_data['movie name'], output_data['average'], align='center', color='royalblue', zorder=10)
        axes[0].set(title='Average rating per movie')
        axes[1].barh(output_data['movie name'], output_data['count'], align='center', color='grey', zorder=10)
        axes[1].set(title='Number of ratings per movie')

        axes[0].invert_xaxis()
        axes[0].set(yticklabels=output_data['movie name'])
        axes[0].yaxis.tick_left()
        axes[0].tick_params(
```

```
            axis='both',
            colors='black')

        axes[1].tick_params(
            axis='y',
            left='off')

        axes[1].tick_params(
            axis='x',
            colors='black')

        fig.tight_layout()
        fig.subplots_adjust(wspace=0.035)
        plt.show()

#Sixth function, allow for user interaction - Put everything together
def interaction(movies, ratings):
    while True:
        action=input("""Hello please enter the number corresponding to your desired action.
        [1] Get information about the dataset
        [2] Search for a movie by title
        [3] Find movies according to certain criteria
        [4] Plot average ratings and number of ratings for top 15 movies of a genre and time period
        [5] Exit
        """)

        if action == '1':
            print(statsfunction(ratings, movies))


        elif action == '2':
            while True:
                title=input("What movie are you looking for? ").lower()
                print("Results for", title)
                print(movie_info(ratings, movies, title))
                restart=input('Do you want to try again? [y] or [n]').lower()
                if restart !='y':
                    break

        elif action == '3':
            while True:
                #Define the input variable for function
                try:
                    start,end=input("What period of time are you interest in? Enter start and end year
separated by a comma: ").split(',')
                    start=str(start)
                    end=str(end)
                    genre=str(input("""What genre are you looking for? :
    Genres available: action, adventure, animation, children, comedy, crime, documentary, drama,
fantasy, film-Noir,
    horror, musical, mystery, romance, sci-fi, thriller, war, western
                                """)).lower()
                    times_rated= int(input("How often should a movie at least be rated? "))
                    avg_rating=float(input("What is the minimum average rating you want? "))
                    #Print Function
```

```
                print(find_movies(movies, ratings, start, end, genre, times_rated, avg_rating))
                restart=input('Do you want to try again? [y] or [n]')
                if restart !='y':
                    break
            except ValueError:
                print("Error - Please make sure that you follow the instructions and enter the correct type
of data")

        elif action == '4':
            while True:
                #Define the input variable for function
                try:
                    start,end=input("What period of time are you interest in? Enter start and end year
separated by a comma: ").split(',')
                    start=str(start)
                    end=str(end)
                    genre=str(input("""What genre are you looking for? :
    Genres available: action, adventure, animation, children, comedy, crime, documentary, drama,
fantasy, film-Noir,
    horror, musical, mystery, romance, sci-fi, thriller, war, western
                            """)).lower()
                    #Print Function
                    print(plot_movies(movies, ratings, start, end, genre))
                    restart=input('Do you want to try again? [y] or [n]')
                    if restart !='y':
                        break
                except ValueError:
                    print("Error - Please make sure that you follow the instructions and enter the correct type
of data")

        end=input('Do you want to end the program? [y] or [n]').lower()
        if end == 'y':
            print("Thank you, bye!")
            break
```

List of references:

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=<http://dx.doi.org/10.1145/2827872>