Box Office Data Scraping and Analysis Program

Student Name: Zhongya Ge (Zack)

Student ID: 28479745

Table of Contents:

Introduction:

At the beginning of the project, I would like to do research and analyse if there is any relationship between the length of taglines and box office performance. After I started the project, I found something more interesting that is Box Office data and I would like to scrape data by myself instead of downloading existing data from online. The Numbers is the website where I scraped the data from. From this website, I can get movies names, domestic box office and international box office, movies budgets and genres. Here I am going to explain how I designed the code and how to use them.

Code Designation:

I will introduce the code designation and how to run them by separating them into 3 parts, including data scraping, data transformation and data analysis.

Data scraping:

Function get_page_urls_array(link, maxpage):

I visited The Number website and see how the URLs and its path link to each other. What I found is the path is its pagination. And then based on the base URL and pagination, I build a function that generates a list od all URLS.

Function get_html(page_url):

In order to extract the information from URL, I need to get html of the webpage. So this function is to get the html format from the input URL.

Function extract_ranking_from_html(raw_html):

Once I get the html format for each page, I create a function to get texts and information that I need, such as ranking, movies name, releasing year, href, Box office amount, etc.

Function get_Box Office_ranking_parallel(urls, column_num) :

There are 16901 movies in the website list. Each page contain 100 movies, which means it needs to my program will visited around 1,700 URLS. I googled if there is any way to speed the process. I found multithreading and build a parallel function based on that. This function will call previous 2 functions. 500 executors will be working together to get html, and then pass the result to extracting function.

Function get_csv_header(dict_list) & create_csv(filename, dict_list, header):
The result we get from the above is dict. And I build these functions to create csv header and create a csv file.

Function def get_movie_after2000(boxOffice_result):
Movies released before Year 2000 are highly likely miss some information. So I have filter out by using key “Year”.

By calling these functions above, we can get "Boxoffice_Metadata.csv", "Budget_Metadata.csv", "ranking_after2000.csv". We also need genres information. So here will be the function to get genres.

Function get_genres_box_links(boxOffice_result):
In "ranking_after2000.csv" file, we have movies href, which will be use to glue with "www.the-numbers.com". This URL will be used to extract genres information. So this function gives us a list of genres URL.

Function get_summary(parsed_html, results, url):
By putting the parsed html, we can extract needed information. From here, we can get genres, movies href, movies running time, language, box office amounts by country, etc.

Function get_Genres_time_language(genres_links):
This is another parallel function to execute 500 URLS, instead of doing this one by one, which saves a lot of time.

And then we save this file as well. 4 files are saved by now.

Data transformation:
In order to work on these data, we need to import these csv file into Python. The format will be data frame.

Function import_file(filename):

This function is designed to import csv file by inputting filename. The result will be data frame. In this function I also call print(file.columns.values) to see what the column names are.

After importing these files, we will start tidying up the data.
Function deleth_and_convert_to_int(df_name, column_names_list, sign)::
This function is created to delete dollar sign or any other signs that are not needed, and convert the column to integer type.

Funciton get_joined_file_on_href(left_file, right_file):
In order to consolidate files into one dataframe, I use merge function and the key will be movies_href. I do not use movies name as key, because they may have duplicate, but movie href will be 100 percent unique.

Function select_columns(columns_name_list):
We have a data frame with all columns, and the needed columns need to be selected. By inputting a list of columns' name, we can choose what columns will be saved.

Data analysis:
By now, there will be a data frame ready to be analysed.

1. Fit a linear regression:

Function fit_LR_plot(comparison_file):
This function will help fit a linear regression model for budget and box office to find the relationship between them. And it can verify if there is a linear relation. And the result will be a graph.

2. Mean analysis:

Function get_mean_and_sum_by_column(comparison_file, column_name):
This function can help get mean and sum by column name. There are 2 column names can be used, which are Year and Genre. We can get mean and sum amount for Year and Genre.

3. Plot Pie Chart:

Function plot_pie(ProductionBudget_or_WorldwideGross):
Function is created for plot pie chart. It will plot budget or box office amounts by genres.

4. Plot Bar Chart:

Function plot_bar_chart(comparison_file, column_name):

Function is created for plot bar chart. It will plot budget or Box Office counts by genres and years to see the number of movies.

How to run:

The whole program needs to be run in Jupyter Hub.

Before running the programming, you can have a brief look at the results in the Jupyter Hub that has been gotten already.

If you are going to rerun the program, please follow the below procedures.

Note: Program codes are in the same order as below, so you may try to run it by clicking each cell. For step 7, it will take around 20 mins to complete as the function will request for 17,000 URLs.

Part A Data scraping:

1. Call these library:

```
#call requests package and BeautifulSoup package to get the html and extract data from html
import requests
from bs4 import BeautifulSoup
import concurrent.futures
#call csv package to save as csv file
import csv
```

2. Run functions below:

```
def get_page_urls_array(link, maxpage):
```

```
def get_html(page_url):
```

```
def extract_ranking_from_html(raw_html):
```

```
def get_BoxOffice_ranking_parallel(urls, column_num):
```

```
def get_csv_header(dict_list):
```

```
def create_csv(filename, dict_list, header):
```

3. Call functions:

In jupyter Book, I have created variables and called the function, you can just run the cell below:

```
Boxoffice_first_link = "https://www.the-numbers.com/box-office-records/domestic/all-movies/cumulative/all-time"#use box
total_pages_BO = 16901#total page number is 16901
all_Boxoffice_links = get_page_urls_array(Boxoffice_first_link, total_pages_BO)#get all links of boxoffice
column_num = 7#7 columns' information to be extracted.
boxOffice_result = get_BoxOffice_ranking_parallel(all_Boxoffice_links, column_num)#call function get_BoxOffice_ranking_
print(len(boxOffice_result))

#Save the data I get into CSV
boxoffice_csv_header=get_csv_header(boxOffice_result)
create_csv("Boxoffice_Metadata.csv", boxOffice_result, boxoffice_csv_header)
```

```
Budget_first_link = "https://www.the-numbers.com/movie/budgets/all"#use budget first page as base link.
total_pages_BG = 6065 #total page number is 6065
all_Budget_links = get_page_urls_array(Budget_first_link,total_pages_BG)#get all links of budgets
column_num = 6 #6 columns' information to be extracted.
budget_result = get_BoxOffice_ranking_parallel(all_Budget_links, column_num)#call function get_BoxOffice_ranking_parall
print(len(budget_result))

#Save the data I get into CSV
budget_csv_header=get_csv_header(budget_result)
create_csv("Budget_Metadata.csv", budget_result, budget_csv_header)
```

The files will be automatically saved in your Jupyter folder.

4. Run this function:

```
def get_movie_after2000(boxOffice_result):
    """To filter out the movies released before 2000, based on Year column."""
    boxOffice_after2000 = []
    for information in boxOffice_result:
        if int(information["Year"]) >= 2000:
            boxOffice_after2000.append(information)
    return boxOffice_after2000
```

5. Call the function to get data of movies released after 2000 and save the csv file:

```
movie_after2000 = get_movie_after2000(boxOffice_result)
print(len(movie_after2000))

#Save the movies released after 2000 into CSV
movies_after2000_csv_header=get_csv_header(movie_after2000)
create_csv("ranking_after2000.csv", movie_after2000, movies_after2000_csv_header)
```

6. Run functions below:

```
def get_genres_box_links(boxOffice_result):
```

```
def get_summary(parsed_html, results, url):
```

```
def get_Genres_time_language(genres_links):
```

7. Call these function: this might take a bit long, roughly 20 mins to scrape data from around 17,000 URLs:

```
genres_links = get_genres_box_links(movie_after2000)#use base url and glue with movie_href, result is a list with all l
genres_and_overseas_info = get_Genres_time_language(genres_links)#use links get genres, duration, releasing date, movie
print(len(genres_links))  #length of links
print(len(genres_and_overseas_info))  #length of results based of links

#Save this data into csv file
final_csv_header=get_csv_header(genres_and_overseas_info)
create_csv("Detail_infor.csv", genres_and_overseas_info, final_csv_header)
```

Once Part A is completed, you will find below files in your folder or Jupyter folder. And then, we can move to part B.

Existing CSV Files: "Boxoffice_Metadata.csv", "Budget_Metadata.csv", "ranking_after2000.csv", "Detail_infor.csv".

Part B Data transformation:

8. Call these library for tidying up data:

```
# Use these library to tidy up data and transform data.
import pandas as pd
import numpy as np
import re
```

9. Run below functions:

```
def import_file(filename):
```

```
def delete_and_convert_to_int(df_name, column_names_list, sign):
```

```
def get_joined_file_on_href(left_file, right_file):
```

```
def select_columns(columns_name_list):
```

```
def divide_number_and_add_column_compare(comparison_file):
```

10. Call functions by running below:

```
#Import 4 files getting from Part A
Boxoffice_Metadata_filename = 'Boxoffice_Metadata.csv'
Boxoffice_df = import_file(Boxoffice_Metadata_filename)

Budget_Metadata_filename = 'Budget_Metadata.csv'
budget_df = import_file(Budget_Metadata_filename)

ranking_after2000_filename = 'ranking_after2000.csv'
movie_after2000_df = import_file(ranking_after2000_filename)

Detail_info_filename = 'Detail_infor.csv'
detail_info_df = import_file(Detail_info_filename)
```

```
#remove $ sign and convert 'ProductionBudget', 'DomesticGross', 'WorldwideGross' in budget_df to int
df_name = budget_df
column_names_list = ['ProductionBudget', 'DomesticGross', 'WorldwideGross']
sign = "$"
budget_df = delete_and_convert_to_int(df_name, column_names_list, sign)
```

```
#remove $ sign and convert 'DomesticBox Office', 'InternationalBox Office', 'WorldwideBox Office' in movie_after2000_df
df_name = movie_after2000_df
column_names_list = ['WorldwideBox Office']
sign = "$"
movie_after2000_df = delete_and_convert_to_int(df_name, column_names_list, sign)
```

```
#left join budget_df, movie_after2000_df, detail_info_df by href together.
joined_file1 = get_joined_file_on_href(budget_df, movie_after2000_df)
joined_file2 = get_joined_file_on_href(joined_file1, detail_info_df)
```

```
#select needed column
columns_name_list = ['Year', 'Genre:', 'ProductionBudget', "WorldwideGross"]
comparison_file = select_columns(columns_name_list)
```

```
comparison_file = divide_number_and_add_column_compare(comparison_file)
```

Once Part B is done, you will have a file named comparison_file, which will be used for analysing.

Part C Data analysis:

11. Call these packages:

```
#call these function will be needed for linear regression.
import numpy as np
import matplotlib.pyplot as plt  # To visualize
import pandas as pd  # To read data
from sklearn.linear_model import LinearRegression
```

12. Run function for linear regression:

```
def fit_LR_plot(comparison_file):
```

13. Call function to plot the linear result:

```
fit_LR_plot(comparison_file)
```

Note: you will get a plot with dots and line.

14. Run function for getting sum and mean:

```
def get_mean_and_sum_by_column(comparison_file, column_name):
```

15. Call this function:

```
genre_mean = get_mean_and_sum_by_column(comparison_file, "Genre")
```

```
Year_mean = get_mean_and_sum_by_column(comparison_file, "Year")
```

Note: you will get 2 tables of sum and mean amounts.

16. Run for package and function below:

```
import matplotlib.pyplot as plt

def plot_pie_chart(comparison_file, ProductionBudget_or_WorldwideGross):
```

17. Call functions:

```
plot_pie_chart(comparison_file, "WorldwideGross")
```

```
plot_pie_chart(comparison_file, "ProductionBudget")
```

Note: you will get 2 pie charts.

18. Running for function plot_bar_chart:

```
def plot_bar_chart(comparison_file, column_name):
```

19. Call the above function:

```
plot_bar_chart(comparison_file, "Genre")
```

```
plot_bar_chart(comparison_file, "Year")
```

Note: we will get 2 bar charts showing numbers of movies by Year or Genre.

Further Study:

If I am going to do further study based on the existing dataset, I would like to analyse box office by country. I can analyse which genre movie are the most popular and least popular by using box office data. And I may make a model to predict if the new movie is going to be popular or not. But it needs more knowledge about coding, mining and movies.