**Movie Box Office Data Mining Project –Documentation**

**Introduction**

For this project to analyze and predict the performance of the future movie’s office box from the model, we program a small set of API using Python to do data collection mainly from <http://the-numbers.com> and <https://www.wikipedia.org/> and <http://www.imdb.com/>, as well as data preprocessing.

**Programming Language**: Python

**Dependency**: Python version above 2.7, Anaconda version above 2.5.0.

It is better to include all program files in the same folder for processing

|  |
| --- |
| Movie\_data\_fetch.py |
| Purpose |
| 1. Fetch movie information from website <http://www.the-numbers.com> in the range of specified years 2. Movie data will be stored in directories arranged by years with HTML format 3. The years are available since 1915, however the information is less complete. For our project, we fetch data from 1990 to 2015 |
| Library used |
| 1. urllib2 – for getting html source code as string 2. re – for using regular expression to fetch wanted data |
| Functions: |
| 1. void get\_movie\_info (start\_year, end\_year, dir\_path)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | **start\_year** | **String** | The start year for fetching | | **end\_year** | **String** | The end year for fetching | | **dir\_path** | **String** | Location to store html file, ex: ‘./years’ means the directory ‘years’ under present directory. | |
| Mechanism: |
| 1. This program is the first one of all to execute 2. First, it gets the links of years from the specified start\_year to end\_year, which shows all the movies listed by release date. 3. For each link of the year, the program will fetch the html source code of all the movies information in the link and save it into the dir\_path year by year. 4. The default start and end year are 1990 and 2015. The default dir\_path is “./years/”, which means the sub-folder “years” at the present folder. |

|  |
| --- |
| Data\_to\_csv.py |
| Purpose |
| 1. Fetch information from the html source code got from the Movie\_data\_fetch.py and save to csv file, in which the delimiter is Tab (\t). 2. File columns include:  |  |  |  | | --- | --- | --- | | **Column Name** | **Source** | **Type** | | Release Date | The numbers or Wikipedia | Date | | Movie Name | The numbers | String | | MPAA Rating | The numbers | String | | Genre | The numbers | String | | Budget | The numbers | Float | | Domestic Box Office | The numbers | Float | | International Box Office | The numbers | Float | | Budget with inflation | The numbers | Float | | Domestic Box Office with inflation | The numbers | Float | | International Box Office with inflation | The numbers | Float | | Weekly Box Office1 | The numbers | Float | | Theater number1 | The numbers | Int | | Weekly Box Office per theater1 | The numbers | Float | | Change1 | The numbers | Int | | Weekly Box Office2 | The numbers | Float | | Theater number2 | The numbers | Int | | Weekly Box Office per theater2 | The numbers | Float | | Change2 | The numbers | Int | | Weekly Box Office3 | The numbers | Float | | Theater number3 | The numbers | Int | | Weekly Box Office per theater3 | The numbers | Float | | Change3 | The numbers | Int | | Weekly Box Office4 | The numbers | Float | | Theater number4 | The numbers | Int | | Weekly Box Office per theater4 | The numbers | Float | | Change4 | The numbers | Int | | During Summer | based on calculation from release date | String | | In Holiday | based on calculation from release date | String | | Franchise | The numbers | String | | Source | The numbers | String | | Production Company | The numbers or wikipedia | String | | Distribution Company | The numbers or wikipedia | String | | Director | The numbers | Int | | Producer1 | The numbers | Int | | Producer2 | The numbers | Int | | Producer3 | The numbers | Int | | Producer4 | The numbers | Int | | Producer5 | The numbers | Int | | ScreenWriter1 | The numbers | Int | | ScreenWriter2 | The numbers | Int | | ScreenWriter3 | The numbers | Int | | Actor1 | The numbers | Int | | Actor2 | The numbers | Int | | Actor3 | The numbers | Int | | Actor4 | The numbers | Int | | Actor5 | The numbers | Int | | Actor6 | The numbers | Int | |
| Library Used |
| 1. os – operation on file 2. re – regular expression on string 3. pandas – use DataFrame to process data 4. dateime – datetime operation 5. urllib2 - for getting html source code |
| Functions: |
| 1. void fetch\_movie\_by\_year(start\_year, end\_year, file\_name)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | **start\_year** | **String** | The start year for fetching | | **end\_year** | **String** | The end year for fetching | | **file\_name** | **String** | The name of output csv file | | **dir\_path** | **String** | Location to store html file, ex: ‘./years’ means the directory ‘years’ under present directory. |  1. String get\_release\_date(source, each\_year, movie\_name)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | source | **String** | Html source code | | each\_year | **String** | The present processing year | | movie\_name | **String** | Movie |  1. String change\_format(date)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | date | **String** | The string of date of this movie |  1. String check\_if\_in\_summer(date\_stamp)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | date\_stamp | **datetime** | The datetime of release date of this movie |  1. String check\_if\_in\_holiday(date\_stamp)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | date\_stamp | **datetime** | The datetime of release date of this movie |  1. String find\_nearest\_day(date\_stamp)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | movie\_day | **datetime** | The datetime of release date of this movie |  1. String find\_mpaa(source)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | source | **String** | Html source code |  1. String find\_genre(source)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | source | **String** | Html source code |  1. String find\_production\_company\_name(source, movie\_name)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | source | **String** | Html source code | | movie\_name | **String** | Movie name |  1. String find\_distri\_company\_name(movie\_name)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | movie\_name | **String** | Movie name |  1. String find\_director(source)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | source | **String** | Html source code |  1. String find\_franchise(source)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | source | **String** | Html source code |  1. String find\_film\_source(source)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | source | **String** | Html source code | |
| Mechanisms |
| 1. The program is the second phase of all. 2. Before starting, the start and end year and file\_name should be specified. The default start and end year are 1990 and 2015; the default string of file\_name is ‘./movies\_1990-2015.csv’ which is given with value the start and end year. 3. Mainly it uses Dataframe object from Pandas library to save each columns of the movies representing a 2D array and then write to output file 4. We have dict objects representing the inflation of each year compared to 2015, and the date of Labor Day and Thanksgiving for each year from 1990 to 2015 by default.      1. On the-numbers website, the first available date for weekly box office is 1990-01-05 and the next available ones are continuous with the range of 7 days, so we add all of the date into a list until the end of 2015, which is 2015-12-30      1. For each file gotten from the first program in each year’s folders, this program process all of them one by one to get the wanted information to columns with the re library to do regular expression searching. 2. The missing value will mainly be ‘NA’ 3. The movie with the domestic box office after considering inflation less than 10000000 will be ignored.     9. The distribution company is fetch from Wikipedia, and the percentage of missing value is higher. |

|  |
| --- |
| Categorization.py |
| Purpose |
| To categorize columns to Integer for data mining  The columns include:   1. MPAA Rating 2. During Summer 3. In holiday 4. Source 5. Franchise (based on total box office) 6. Production Company 7. Distribution Company 8. Director (based on total box office) 9. Producer1-5 (based on total box office) 10. ScreenWriter1-3 (based on total box office) 11. Actor1-6 (based on total box office) |
| Library used |
| pandas – the use of Dataframe |
| Functions |
| 1. void categorization(input, output, twenty, fifty)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | input | **String** | The file path and name of input csv file | | output | **String** | The file path and name of output csv file | | used\_profit | **Int** | The standard of net profit of a movie to determine the tier of the data field |  1. void add\_to\_dict(the\_dict, target, net\_profit, false\_string)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | the\_dict | **dict** | The dict which will be used to append the box office of the present processing movie for the same name of the processing column | | target | **String** | target string value to be transformed to nth-tier | | net\_profit | **Float** | the net profit of this movie | | false\_string | **string** | the string of unusable value like ‘NA’ or ‘’ |  1. void categorize\_based\_on\_profit(target\_profit, the\_dict, target, false\_string, column\_name, the\_index, data\_frame)  |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | target\_profit | **Float** | The standard of profit used to judge the tier of a given target | | the\_dict | **dict** | The dict which will be used to append the box office of the present processing movie for the same name of the processing column | | target | **String** | target string value to be transformed to nth-tier | | false\_string | **String** | the string of unusable value like ‘NA’ or ‘’ | | column\_name | **String** | the column name the target string belongs | | the\_index | **Int** | the index of the row in which the target | | data\_frame | **Dataframe** | the movie data-frame as a input parameter | |
| Mechanisms |
| 1. This is the thrid step all of program 2. The input file should be the output file of the second program. 3. Column for MPAA rating will be transformed to number:   Not Rated=0, G=1, PG=2, PG-13=3, R=4, NC-17=5   1. Column during\_summer will be transformed to number:   Yes=1, No=0   1. Column for in\_holiday will be transformed to number:   In holiday=1, No=0   1. Column for source will be transformed to number:   “Original Screenplay”=0, others=1   1. All other columns are transformed to number from 1 to 4 based on the net profit of a movie, the default used profit is 50000000 |

|  |
| --- |
| Get\_imdb\_page.py |
| Purpose |
| According to the movie name listed on CSV file, find the movie’s webpage that contains plot summary and save the html source code into local machine |
| Library used |
| urllib2 – read the html source code of the given link  re – regular expression for string operation  json – data column fetching when searching on Google  mechanize – mimicking a real browser to send requests and avoid being blocked  pandas – dataframe operation  os - |
| Functions |
| void get\_imdb\_page( input\_file, output\_path)   |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | input\_file | **String** | The file path and name of input csv file | | output\_path | **String** | The output path to store movie’s html source code | |
| Mechanism |
| 1. This is the fourth step of all programs. 2. First it will load the CSV file into as a Datafream object 3. For each movie name in the csv file, it will search the movie’s id (for example, Titanic is ‘tt0120338’) in IMDB website either by IMDB search engine or by Google search API with JSON format. 4. After getting the movie id, it then starts to get the html source code of plot summary page of the movie. 5. For each movie, save to output\_path as .html file |

|  |
| --- |
| Get\_imdb\_plot.py |
| Purpose |
| From the output\_path of Get\_imdb\_page.py, for each .html file, get the contents of plot summary of each movie and do text preprocessing. And then save the content of plot summary of each movie into a .txt file in a folder |
| Library used |
| os – mainly for loading files in a folder  re – regular expression for string operation |
| Functions |
| void get\_imdb\_plot(input\_path, output\_path)   |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | input\_path | **String** | The input path of the html source code | | output\_path | **String** | The output path for the .txt files | |
| Mechanism |
| 1. For each html file in input\_path, load the source code, get the plot summary and do some pre-processing. 2. Save the plot summaries into .txt file at the output path for each movie. |

|  |
| --- |
| Calculate\_text\_value.py |
| Purpose |
| Calculate the text value, and add the columns for first three high-scored words and their scores into the movie dataframe |
| Required Files |
| all\_first\_name.txt  all\_surname.txt  The name dataset is from <https://github.com/enorvelle/NameDatabases> |
| Library used |
| textblob – for calculating text value  os – mainly for loading files in a folder  pandas – using dataframe  sys – setting default encoding  math – needed by textblob |
| Functions |
| void calculate\_text\_value(plot\_input\_path, input\_file, output\_file)   |  |  |  | | --- | --- | --- | | **Parameter Name** | **Parameter Type** | **Description** | | plot\_input\_path | **String** | The input path of the plot summary of movies | | input\_file | **String** | The input file path of .csv which is used to be loaded as a Dataframe | | output\_file | **String** | The output file\_path of .csv which has the added first three high text and value. | |
| Mechanism |
| 1. First add three text columns and three score columns with default values. 2. For each plot summaries txt file, send the plot summaries of all movies to token database for analysis. 3. After the database is completed, a for loop will go through all movies and get their first three high-scored texts and values with the filtering function to ignore the unimportant words like human names, pronouns, preposition and conjunctions, then add them to the six new-added columns in the movie Dataframe. 4. Output the modified movie Dataframe as the output\_file path. |