UNIVERSITY OF MALAYA

EXAMINATION FOR THE DEGREE OF MASTER OF DATA SCIENCE

ACADEMIC SESSION 2019/2020 : SEMESTER II

WQD7005 : Data Mining

June 2020

INSTRUCTIONS TO CANDIDATES :

Answer **ALL** questions (50 marks).

Name : Nurullainy Binti Mat Rashid

ID : 17036591

Link to codes and data:

<https://github.com/Nurullainy/Data-Mining-Final-Exam>

Link to video : <https://drive.google.com/drive/folders/1hMqLYw5ubSpMRsPvyJwjDG8MuV3GO_Qt?usp=sharing>

(This question paper consists of 5 questions on 3 printed pages)

Mini-assignment (50 marks)

Instructions: Work individually, submission via Spectrum.

1. You are required to make a user-agent that will crawl the WWW (your familiar domain) to produce dataset of a particular website.

* the web site can be as simple as a list of webpages and what other pages they

link to

* the output does not need to be in XHTML (or HTML) form

a multi-stage approach (e.g. produce the xhtml or html in csv format)

**Topic: Web Crawling Data of TV Shows and Movies at Online Internet Movie Database (IMDb)**

IMDb is an online database of information related to films, television programs, home videos, video games, and streaming content online – including cast, year released, ratings, production crew, plot summaries, trivia, fan and critical reviews.

I want to analyze data of TV Show and movie from IMDb. The data can be extracted from this website : https://www.imdb.com/search/title/?year=2017

**1.1 Importing Python Libraries**

First of all, I will do the following step:

1) Import requests module and BeautifulSoup from bs4

2) Assign the address of the web page to a variable named ‘url’.

3) Request the server the content of the web page by using get( ), and store the server’s response in the variable ‘response’.

4) Print HTTP status code and a small part of response‘s content by accessing its text attribute (response is now a Response object).

A screenshot of a social media post

Description automatically generated

**1.2 Using Beautifulsoup Module To Parse The HTML Content**

Parsing HTML document and extract the 50 div containers

1) Import the BeautifulSoup class creator from the package bs4.

2) Parse response.text by creating a BeautifulSoup object, and assign this object to page\_soup.

3) The 'html.parser' argument indicates that we want to do the parsing using Python’s built-in HTML parser.

A screenshot of a social media post

Description automatically generated

Class attribute has two values; 1) lister-item and 2) mode-advanced.

This combination is unique to these div containers. Use the find\_all() method to extract all the div containers that have a class attribute of *lister-item mode-advanced* and assign it to variable *movie\_container*. It will return a ResultSet object which is a list containing all the 50 divs

A screenshot of a social media post

Description automatically generated

**1.3 Extracting The Data For A Single Movie**

Now I’m selecting one movie container (let say the first container) to extract 9 attributes that I am interested with for next data mining purposes:

A screenshot of a social media post

Description automatically generatedA screenshot of a social media post

Description automatically generated

The 9 following attributes are as follows:

1. The name of the TV show or movie
2. The year of release
3. Runtime of each TV show or movie
4. Genre of TV show or movie
5. Revenues from the movie released
6. The IMDB rating
7. The number of votes from user
8. Stars of the TV show or movie (name of director and main cast)
9. Hyperlink to the TV show or movie

**1.4 Extracting Information For All The Tv Shows And Movies In A Single Page**

1. Declare list of variables to have something to store the extracted data in.
2. Loop through each container in a web page (the variable which contains all the 50 movie containers).
3. Extract the data points of interest only if the container is True

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

Let’s check the data collected using Pandas library. Then print the 10 movie\_container in first web page

A screenshot of a social media post

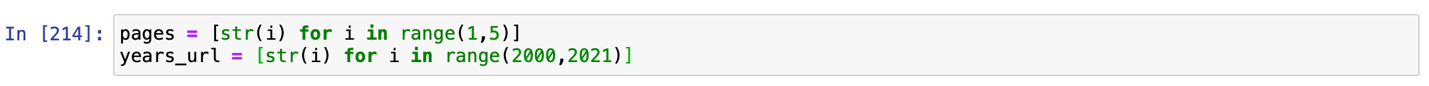
Description automatically generated

A close up of a piece of paper

Description automatically generated

**1.5 Extracting data for all TV shows and movies from multiple pages from year 2000 – 2020**

1. Create a list called *pages*, and populate it with the strings corresponding to the first 4 pages.
2. Create a list called *years\_url* and populate it with the strings corresponding to the years 2000 - 2020



**1.6 Controlling the crawl-rate**

Controlling the rate of crawling is important for the website that I will be scraping. If I not controlling the rate of crawling, much less likely to get my IP address banned.

Need to avoid activity disruption of the scraped website by allowing the server to respond to other users’ requests too.

Control the loop’s rate by using the *sleep()* function from Python’s time module. *sleep()* will pause the execution of the loop for a specified amount of seconds. To mimic human behavior, I’ll vary the amount of waiting time between requests by using the *randint()* function from the Python’s random module. *randint()* randomly generates integers within a specified interval.



Since the web scraping is more than 10 pages, it would be nice to find a way to monitor the scraping process as it’s still going. The greater the number of pages, the more helpful the monitoring becomes.

For my script, I’ll make use of this feature, and monitor the following parameters:

1) The *frequency (speed) of requests*, to ensure our program is not overloading the server

2) The *number of requests*, so I can halt the loop in case the number of expected requests is exceeded

3) The *status code of our requests*, to make sure the server is sending back the proper responses

To get a frequency value, I divide the number of requests by the time elapsed since the first request.

1) Set a starting time using the *time()* function from the time module, and assign the value to start\_time.

2) Assign 0 to the variable requests which to be use to count the number of requests.

3) Start a loop, and then with each iteration:

- Simulate a request

- Increment the number of requests by 1

- Pause the loop for a time interval between 8 and 15 seconds

- Calculate the elapsed time since the first request, and assign the value to elapsed\_time

- Print the number of requests and the frequency

If I set my request to 100 requests, my output return will look a lengthy and a bit untidy as the output accumulates. To avoid that, I’ll clear the output after each iteration, and replace it with information about the most recent request. I use the *clear\_output()* function from the IPython’s core.display module. Then set the wait parameter of clear\_output() to True to wait with replacing the current output until some new output appears.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

I set the loop limit to 100 but the requests stop at request number 84. This indicates that I have collected all TV shows and movies data from year 2000 – 2020.

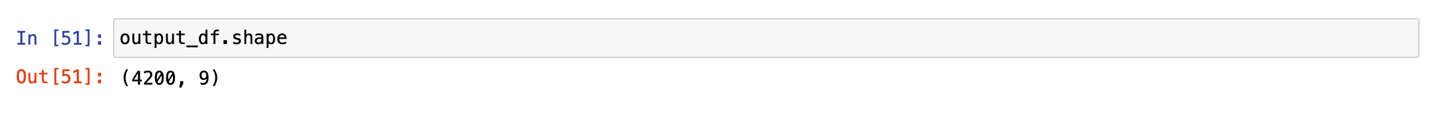
Let’s check the data collected using Pandas library. Then print the 10 movie\_container in first web page



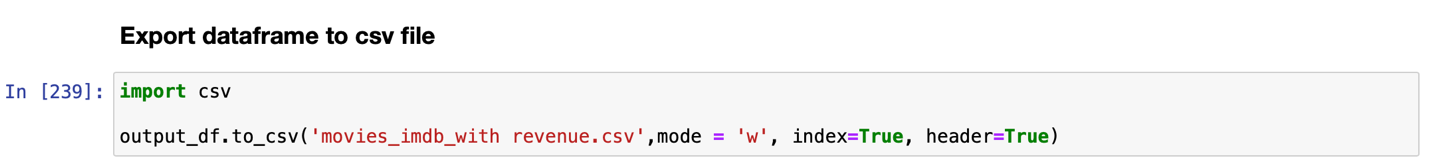
A screenshot of a cell phone

Description automatically generated

Total number of TV shows and movies collected are 4200 titles.



Export dataframe to csv file



Snapshot of exported csv file (movie\_imdb\_with\_revenue) dataset

A screenshot of a computer

Description automatically generated

(10 marks)

1. Draw snowflake schema diagram for the above dataset. Justify your attributes to be selected in the respective dimensions.

(10 marks)

1. You are required to write code to create a decision tree (DT) model using the above dataset (Question 1). In order to achieve the task, you are going to cover the following steps:

* Importing required libraries
* Loading Data
* Feature Selection
* Splitting Data
* Building Decision Tree Model
* Evaluating Model
* Visualizing Decision Trees

(10 marks)

1. You are required to write code to find frequent itemsets using the above dataset (Question 1). In order to achieve the task, you are going to cover the following steps:

* Importing required libraries
* Creating a list from dataset (Question 1)
* Convert list to dataframe with boolean values
* Find frequently occurring itemsets using Apriori Algorithm
* Find frequently occurring itemsets using F-P Growth
* Mine the Association Rules

(10 marks)

1. It will be appeared in week 14.

( 10 marks)

Submissions:

The student is expected to submit answers to each question individually, and submit the document in PDF format. The student can include online materials, screenshots, videos and/or codes (ipynb format) to support your answer