COMP20008 - 2018 - SM2 - Project Phase 1

Release Date: 11:59am Monday, 13^{th} August 2018

Due Date: 11:59am Friday, 31^{st} August 2018

Submission is via the LMS

Please, make sure you get a submission confirmation email once you submit your assignment. Otherwise, it will be considered as a late submission.

Phase 1: Warmup - Python Exercises (20 marks, worth 20% of subject grade)

In this phase, you will practice your Python wrangling skills with a publicly available dataset. The dataset is obtained through the TMDB (The Movie DB) API. It contains information on movies featured in the Full MovieLens Dataset and released on or before July 2017. The main features of the Movies Metadata file include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.

You will be working with the following dataset in this phase:

- Movies_tmdb.csv: It has a set of movie records (approx. 45,000), released **on or before July 2017**. Note that this dataset is quite large, and you may find it beneficial during development, to first test your code on a smaller sample of this data.

Libraries to use are Pandas and Matplotlib. You will need to write Python 3 code and work with Series and DataFrames discussed in workshop week 2 and data cleaning and basic visualisations covered in workshop weeks 3-4. If you are using other packages, you must provide an explanation in your code about why it is necessary.

Import Required Python Libraries and Load the Data

Please write here all the Python libraries you will be using! Also load the dataset (.csv) in a dataframe object.

In [ ]:

```
#import ....
import pandas as pd
movies_df =  pd.read_csv("Movies_tmdb.csv", low_memory=False)
```

1 Understanding the Dataset (3 Marks)

1.1 Print the number of movies, number of attributes/columns, column names and datatypes. The output of this step should look like **(2 Marks)**

```
***
Q1.1
Number of movies: #
Number of attributes/columns: #
Column names: #
Column datatypes: #
***
```

where # is the values/strings you find.

In [ ]:

```
### answer Q1.1
```

1.2 In this assignment, we won't be using all the features (i.e. columns) which are included in the csv file, so create a new dataframe with the following columns: **(1 Marks)**

title	*genre*	*release_date*	*runtime*	*budget*	*revenue*	*original_language*

popularity	*vote_average*	*vote_count*	*adult*	*production_countries*

You must keep the order of the columns as provided above. Output of this question should be printing the first TWO rows (i.e. movies) from the new created dataframe in the following format:

```
***
Q1.2
The first two rows from the filtered movies dataframe are:
#
#
***
```

where each # represents one movie row.

In [ ]:

```
### answer Q1.2
```

2 Missing Values and Data Types (5 Marks)

2.1 Most of the columns in the movies dataframe have object datatype, let's convert the **"popularity"** column to float64 datatype, **"title"** column to string and **"adult"** column to boolean. **(1 Mark)**

The output of this step should print the datatypes of all columns in the movies dataframe after the conversion. You should follow the following format:

```
***
Q2.1 Datatypes after conversion:
#
***
```

where # should be the datatypes of the dataframe columns. Note: You don't have to create a new dataframe for this question, instead you can use the same dataframe which you created in Q1.2.

In [ ]:

```
### answer Q2.1
```

2.2 Now, we will deal with the missing values as a preprocessing step before performing any further analysis. Let's first print the total number of missing values for each column separately. Following this, you should print the percentage of movies with incomplete data in any of its attributes (i.e. missing values). Note: A movie is considered incomplete record if it has a missed value in at least one of its features. **(2 Marks)**

Note: missing values might be 0, nan, or empty cell.

```
***
Q2.2 Number of missing values per attribute:
col_1: x
col_2: x
...
col_n: x
***
% of movies with incomplete data: #
***
```

Replace col_1,col_2 ... col_n with the columns' names, x with the calculated values, and # with the calculated percentage.

In [ ]:

```
### answer Q2.2
```

2.3 Write code that will add a new column called "runtime_non_missing" to the movies dataframe. The values in the new column should be copied from the "runtime" column and replaces all missing values in this column with the average of non-missing values for that column. **(2 Marks)**.

The output of this question should print the average calculated value in the following format:

```
***
Q2.3 Missing values in 'runtime' column are replaced with:
#
***
```

Where # is the calculated value.

Do you think it will be better to replace the missing values in the "runtime" column with the median instead of the average? Yes/No - Why?

In [ ]:

```
### answer Q2.3
```

```
### answer Q2.3 justification
```

3 Cleaning Dataset (8 Marks)

3.1 In this question, you will deal with the datetime datatype. The question has three parts as following: **(4 Marks)**.

Dealing with data formats is an essential step in the data wrangling pipeline. One of the issues is that data entry might be inconsistent. For example, by looking at the "release_date" column, you will find two different formats for the date value: '%m/%d/%Y' and '%Y-%m-%d'. Write code which converts the "release_date" into datetime datatype and consider reading both formats correctly. The final datetime format should be '%m/%d/%Y'.

Another issue is the wrong values for some of these dates, for example some movies have the "release_date" after July 2017. However, in the description of the dataset, it says the collected movies released on or before July 2017. To deal with this issue, write code which removes any suspicious records (i.e. any movie which has a "release_date" after July 2017).

In this assignment, we are not interested in analysing movies released before 1990. Therefore, as a preparation for our next questions, we only want to keep movies with release date between Jan 1990 and July 2017 (inclusive, i.e. Jan 1990 $\leq$ release_date $\leq$ July 2017) in the movies dataframe. So write code to delete all movies released outside this interval or has a nan/empty "release_date". You should display the number of the records (i.e. movies) in the final filtered movies dataframe.

The output of this question should be in the following format:

```
***
Q3.1
The number of movies with release date between Jan-1990 and July-2017: #
***
```

Where # is the calculated number.

Note: The resulting dataframe will be used to answer the remaining questions.

In [ ]:

```
### answer Q3.1
```

3.2 You might have noticed that the data of the genres column is represented as a list of dictionaries. Let's change this format into an easier one. Write code to convert the values of genres column into a list of strings instead of a list of dictionaries, keeping only the value of the "name" attribute. For example, the value [{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id': 10751, 'name': 'Family'}] should be ['Animation','Comedey','Family']. The newly converted values should be stored in an extra new column called "genres_cleaned". **(4 Marks)**.

You should display the first 5 rows of the movies dataframe after adding this new column. The output of this question should be in the following format:

```
***
Q3.2
The first 5 rows after adding the "genres_cleaned" column are:
#
***
```

where # is the first 5 rows in the movies dataframe.

In [ ]:

```
### answer Q3.2
```

4 Basic statistics, summary and grouping (10 Marks)

4.1 Write code that calculates the median and average of non-missing values in the budget column for movies released between 2000 and 2010 (inclusive, i.e. 2000 $\leq$ release_year $\leq$ 2010). **(2 Marks)**.

Your code should print out the results with the following format:

```
***
Q4.1: Movies budget (2000-2010)
Median: #
Average: #
***
```

where # is the calculated values rounded to 1 decimal place.

In [ ]:

```
### answer Q4.1
```

4.2 Write code that returns a "popular_movies" dataframe with the most popular movie for each year since 2000. This means the dataframe will contain 18 movie, one for each year from 2000 till 2017). The dataframe should also contain the following columns: "title", "release_date", "runtime" , "original_language", "popularity". Also, the dataframe should be sorted by the "popularity" values in **descending** order. **(2 Marks)**.

Your code should print out the popular_movies dataframe in the following format:

```
***
Q4.2: Most popular movies (2000-2017):
```

title	release date	runtime	language	popularity
#				
#				
#				
.				
.				
.				
#				

***

where each # represents one row in the popular_movies dataframe.

In [ ]:

```
### answer Q4.2
```

4.3 In this question, we will be working with the "vote_average" and "vote_count" columns. Write the code that returns the "title", "vote_average", "vote_count" of the 10 lowest voted average movies with at least 400 voters (i.e. "vote_count)". The 10 movies should be displayed in ascending order by the "vote_average" values. **(2 Marks)**.

Your code should print out the lowest voted average movies in the following format:

***
Q4.3: The 10 movies with the lowest vote average are:

title	vote average	vote count
#		
#		
#		
.		
.		
.		
#		

***

where each # represent one of the 10 movies.

In [ ]:

```
### answer Q4.3
```

4.4 Write code to display the count of movies for the top three movie production countries since Feb-2005 (i.e. Feb-2005 $\leq$ release_date). **(4 Marks)**.

Your code should print out the result in the following format:

***
Q4.4: Top 3 Movie Production Countries since Feb-2005:

Country	Count of Produced Movies
x	y
x	y
x	y

***

where x represents the country name and y is the count of movies produced by this country.

In [ ]:

```
### answer Q4.4
```

5 Visualization (13 Marks)

5.1 Boxplots **(2 Marks)**.

Draw a plot consisting of two boxplots. One boxplot to show the distribution of revenue for adult movies. One boxplot to show the distribution of revenue for other non-adult movies. Note: You should not include movies with zero-revenue in the box-plot.

In [ ]:

```
### answer Q5.1
```

5.2 Histogram **(2 Marks)**

Draw a bar plot showing month of year (x-axis) versus total number of movies released on that month (y-axis). Each bar will represent the total number of movies released on a specific month across all years.

Is there any relation between the month of the year and the total number of movies? Yes/No - Explain?

In [ ]:

```
### answer code Q5.2
```

```
### answer justification 5.2
```

5.3 Scatter plot **(3 Marks)**

In this question, we will analyze the relation between few columns in the movies dataset. Draw four plots with the following four scatter/line plots:

```
1- Non-zero revenue movies (x-axis) versus number of genres.
2- Non-zero revenue movies (x-axis) versus release year.
3- Non-zero revenue movies (x-axis) versus runtime.
4- Non-zero revenue movies (x-axis) versus vote average.
```

Pick one of the four plots and justify/explain the relation between the two attributes. You should mention whether the relation is positive, negative or random. Did you expect this type of relation, Yes/No? Why?

In [ ]:

```
### answer Q5.3
```

```
### answer justification 5.3
```

5.4 Pie chart **(2 Marks)**

Create a pie chart showing the number of movies for each genre. For example, if a movie is classified as both comedey and action then the count for each of the action and comedey slices should be increased by 1. Each slice of the pie should have a different colour and contain a percentage number listing its relative size. Also, each slice of the pie should have a label next to it indicating which genre it corresponds to.

In [ ]:

```
### answer Q5.4
```

5.5 Parallel co-ordinates **(4 Marks)**

In this question, we will the parallel co-ordinates plot to visualize the trend/relation between some of the features in the movies dataset. You should write the code that implements the following steps:

1. Delete all movies (rows) with missing values in any of the following features: budget, runtime, popularity, vote_average, and revenue.
2. Then for each of the features, normalise its values to lie within the range [0-1] (0 to 1 inclusive). Use the the following formula for normalising a feature:

$newvalue = (oldvalue - min)/(max - min)$ where min is the minimum value for the feature, max is the maximum value for the feature, newvalue is the normalised value for the feature and oldvalue is the old (un-normalised value).

3. Using these normalised features, compute the mean budget, runtime, popularity, vote_average, and revenue for each year.
4. Finally, draw a parallel co-ordinates plot, each line correspondes to a different year. The ordering of the features for the plot should be budget(leftmost), runtime, popularity, vote_average, and revenue(rightmost).

Colour the movies with $release\_year < 2000$ in red, $2000 \geqslant release\_year < 2010$ in green and $release\_year \geqslant 2010$ in blue. Provide a legend mapping colours to day type.

From the plot, can you see any relation between the revenue and popularity features? Yes/No- Explain?

In [ ]:

```
### answer Q5.5
```

```
### answer justification 5.5
```

Marking scheme

Correctness (39 marks): For each of the 5 questions a mark will be allocated for level of correctness (does it provide the right answer, is the logic right), according to the number in parentheses next to each question. Note that your code should work for any data input formatted in the same way as Movies_tmdb.csv. E.g. if a random sample of 20,000 records was taken from Movies_tmdb.csv, your code should provide a correct answer if this was instead used as the input.

Correctness will also take into account the readability and labelling provided for any plots and figures (plots should include title of the plot, labels/scale on axes, names of axes, and legends for colours where appropriate).

Coding style **(1 Mark)**: Mark will be allocated for coding style. In particular the following aspects will be considered:

- Formatting of code (e.g. use of indentation and overall readability for a human)
- Code modularity and flexibility. Use of functions or loops where appropriate, to avoid redundant or excessively verbose definitions of code.
- Use of python library functions (you should avoid reinventing logic if a library function can be used instead)
- Code commenting and clarity of logic. You should provide comments about the logic of our code for each question, so that it can be easily understood by the marker.

Submission Instructions

Via the LMS, submit a jupyter notebook containing the code. Make sure you get a submission receipt via email. If you didn't get a receipt via email, this means we didn't receive your submission and it will be considered as late submission.

Other

Extensions and Late Submission Penalties: If requesting an extension due to illness, please submit a medical certificate to the lecturer. If there are any other exceptional circumstances, please contact the lecturer with plenty of notice. Late submissions without an approved extension will attract a penalty of 10% of the marks available per 24hr period (or part thereof) that it is late. E.g. A late submission will be penalised 2 marks if 4 hours late, 4 marks if 28 hours late, 6 marks if 50 hours late, 8 marks if 73 hours late, 10 marks if 106 hours late, etc.

Phase 1 is expected to require 20-24 hours work.

Academic Honesty

You are expected to follow the academic honesty guidelines on the University website https://academichonesty.unimelb.edu.au (https://academichonesty.unimelb.edu.au)

Further Information

A project discussion forum has also been created on the subject LMS. Please use this in the first instance if you have questions, since it will allow discussion and responses to be seen by everyone. The Phase 1 project page will also contain a list of frequently asked questions.

In [ ]: