Project 3: Baby Names

CS3753/CS5163: Data Science Summer 2023

Instructor: Dr. Mohammad Imran Chowdhury

Total: 100 Points

Due: 07/11/2023 11:59 PM

In this project, I invite you to do the following tasks using the Python libraries such as **Pandas** and **Matplotlib** covered in class.

Task 1 (20 points): Load the Baby Names Dataset provided to you as the ``**names.zip**" file into the Jupyter Notebook. Note that this notebook requires Python 3.6 or higher. After extracting and loading the Baby Names dataset zip file into the Jupyter Notebook, the output should look as follows: **(5 points)**

```
In [3]: import zipfile
        zipfile.ZipFile('names.zip').extractall('.')

In [4]: import os
        os.listdir('names')

Out[4]: ['names',
         'NationalReadMe.pdf',
         'yob1880.txt',
         'yob1881.txt',
         'yob1882.txt',
         'yob1883.txt',
         'yob1884.txt',
         'yob1885.txt',
         'yob1886.txt']
```

Note: All these .txt files contain comma-separated values (CSV). That is each record in the individual annual files has the format "name,sex,number," where the name is 2 to 15 characters, the sex is M (male) or F (female), and "number" is the number of occurrences of the name.

Now, for each year in 1880-2014, load the corresponding CSV file 'names/yobXXXX.txt' as DataFrame, create a new column "year" with all elements set to loop variable, then concatenate all DataFrames into a single one named as ``**allyears**". After issuing the ``allyears.head()" and ``allyears.tail()" command the output should be as follows: **(15 points)**

```
In [11]: allyears.head()
```

Out[11]:

	name	sex	number	year
0	Mary	F	7065	1880
1	Anna	F	2604	1880
2	Emma	F	2003	1880
3	Elizabeth	F	1939	1880
4	Minnie	F	1746	1880

```
In [12]: allyears.tail()
```

Out[12]:

	name	sex	number	year
33039	Zykeem	M	5	2014
33040	Zymeer	M	5	2014
33041	Zymiere	M	5	2014
33042	Zyran	M	5	2014
33043	Zyrin	M	5	2014

Task 2 (20 points): Compute the number of times that each name was used, separately for boys and girls. This involves, first working with the **``allyears"** DataFrame from **Task 1** and grouping the DataFrame by ``sex" and ``name". Then calling the **sum()** method on the ``number" column, and finally calling the **unstack()** method on the result by the ``sex" column again to get the number of times that each name was used, separately for boys and girls. Name the resultant DataFrame as **``totals_bysex"**. If you print the first five rows then the output should be as follows:

```
totals_bysex.head()
```

Out[17]:

sex	F	M
name		
Aaban	NaN	72.0
Aabha	21.0	NaN
Aabid	NaN	5.0
Aabriella	10.0	NaN
Aadam	NaN	196.0

Task 3 (30 points): Identify the unisex names where the ratio between the boys and girls totals is between 1-to-4 and 4-to-1. This task has three (03) steps. These are as follows:

(a) First, you need to compute the total number of times that each name was used, both for boys and girls.

This involves summing up the ``F" and ``M" columns of the DataFrame **``totals_bysex"**. That is you need to call the **sum()** method on the **``totals_bysex"** DataFrame and save the result as the **``totals_both"** DataFrame. The first five rows of **``totals_both"** DataFrame should be as follows: **(5 points)**

```
In [19]: totals_both.head()

Out[19]: name
         Aaban         72.0
         Aabha         21.0
         Aabid          5.0
         Aabriella     10.0
         Aadam        196.0
         dtype: float64
```

(b) Second, you need to calculate the **ratio**.

This can be achieved by dividing the column ``F" of the DataFrame **``totals_bysex"** by the column ``M" of the DataFrame **``totals_bysex"**. The first five rows of the **``ratio"** DataFrame should be as follows: **(5 points)**

```
In [21]: ratio.head()

Out[21]: name
         Aaban       NaN
         Aabha       NaN
         Aabid       NaN
         Aabriella   NaN
         Aadam       NaN
         dtype: float64
```

(c) Third, you need to use the following unisex name formula on the ``totals_both" DataFrame to select the only rows from the ``totals_both" DataFrame that follows the given unisex name formula:

(ratio > 0.25) & (ratio < 4)

After that, you need to sort the ``totals_both" DataFrame to get the top ten (10) unisex names. This involves calling the **sort_values()** method on the ``totals_both" DataFrame. After applying the given unisex name formula and the **sort_values()** method, if you print the first ten (10) rows i.e., the top ten (10) unisex names from the ``totals_both" DataFrame, then the output should be as follows: **(20 points)**

```
totals_both.head(10)

Out[23]: name
         Willie    593888.0
         Jordan    479434.0
         Taylor    416096.0
         Leslie    376587.0
         Jamie     350262.0
         Angel     301425.0
         Lee       291691.0
         Jessie    274931.0
         Marion    259549.0
         Dana      243517.0
         dtype: float64
```

Task 4 (30 points): Plot **popularity vs. year** for the top ten (10) unisex names

To match exactly with my output, you have to work on the **subplot()** and **plotname()** methods as discussed in class. The **plotname()** method works on the ``allyears_indexed" dataset.

In class, we have also discussed how to get the ``allyears_indexed" dataset. That is, you need to apply the **set_index()** method on the ``allyears" dataset of **Task1**. The arguments of the **set_index()** method are **[sex, name, year]**. And then you need to call the **sort_index()** method on it. This is also clearly shown in the lecture slides on Pandas. The ``allyears_indexed" dataset

should look as follows: **(10 points)**

```
In [14]: allyears_indexed
```

Out[14]:

sex	name	year	number
F	Aabha	2011	7
		2012	5
		2014	9
	Aabriella	2008	5
		2014	5
...	...	...	...
M	Zytavious	2009	7
		2010	6
	Zyvion	2009	5
	Zyyon	2014	6
	Zzyzx	2010	5

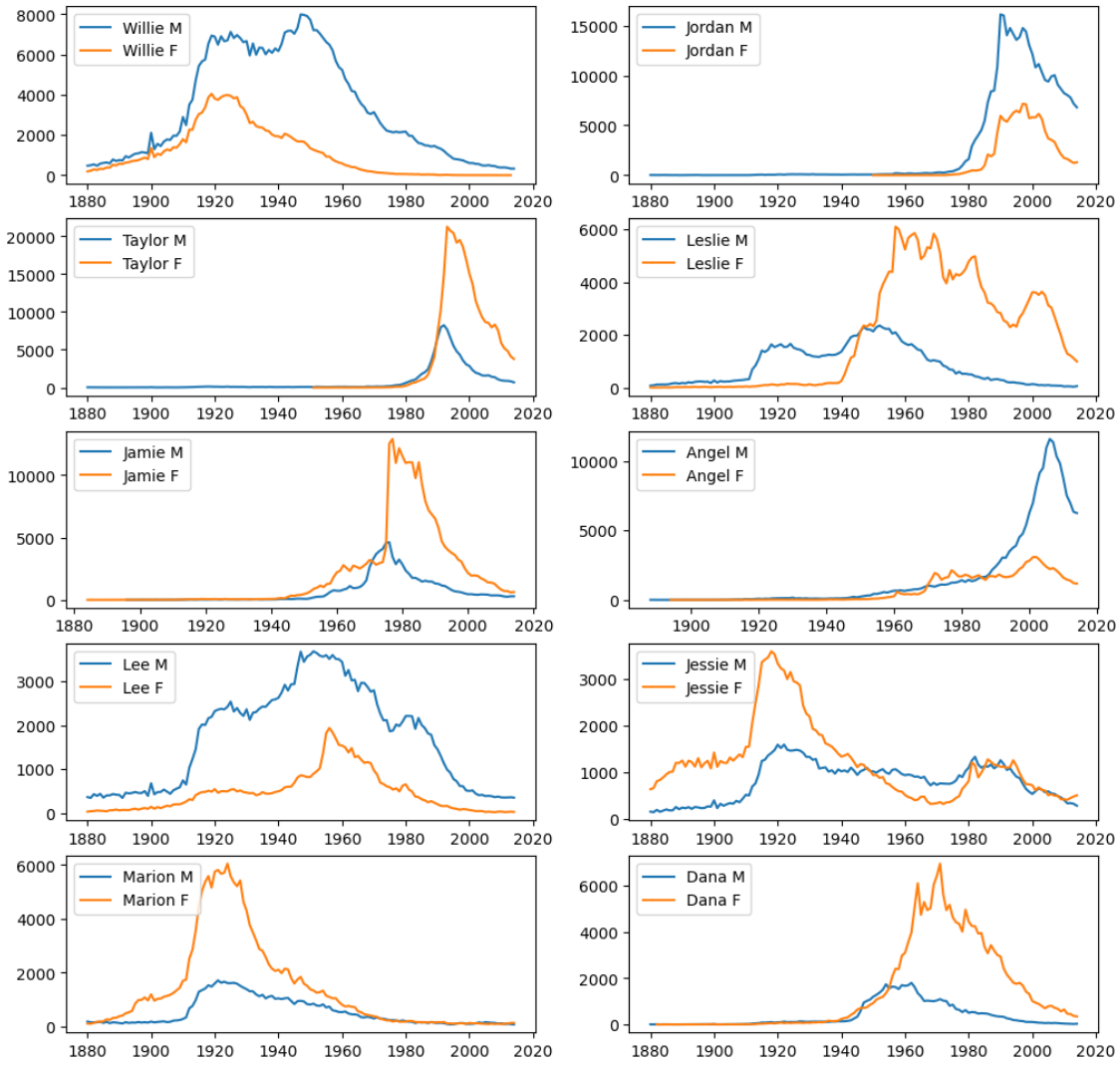
1825433 rows × 1 columns

Note that the **plotname()** method takes two parameters **sex** and **name**. The values for the **name** parameter should be the top ten (10) unisex names found in **Task 3 of part (c)**. For each top ten (10) unisex names, the value of the **sex** parameter will be ``M" and ``F" respectively. That is, you need to call the **plotname()** method twice to exactly match with my output given below. For example, unisex name **Willie**, the **plotname()** method will be called as follows:

plotname(`M', `Willie');
plotname(`F', `Willie');

After you plot the **popularity vs. year** for the top ten (10) unisex names, the output should be as follows: **(20 points)**

Make sure the legend, X-axis label, and Y-axis label match EXACTLY with the given final plot. You may get a different color. That's fine. But all information on the plot should EXACTLY match to get the full credits on this **Task 4**.



The submission grading rubric is as follows (points out of 100 total):

Project element	**Points**
Task 1	20
Task 2	20
Task 3	30
Task 4	30

Submission Instructions: Create a compressed file (.zip or .tar.gz files are accepted) with your all source files such as .ipynb files and data files. Generally speaking to complete Task1 through Task4, you just need one .ipynb file. But it's better to submit everything as a compressed zip file. Submit the compressed zip file to the "Project Submissions" area on the Blackboard course website.

Late submission policy: As described in the syllabus, any late submission will the penalized with 10% off after each 24 hours late. For example, an assignment worth 100 points turned in 2 days late will receive a 20 point penalty. Assignments turned in 5 or more days after the due date will receive a grade of 0.