Project 4: Epub

CS 5473: Data Mining Fall 2022

Instructor: Dr. Mohammad Imran Chowdhury

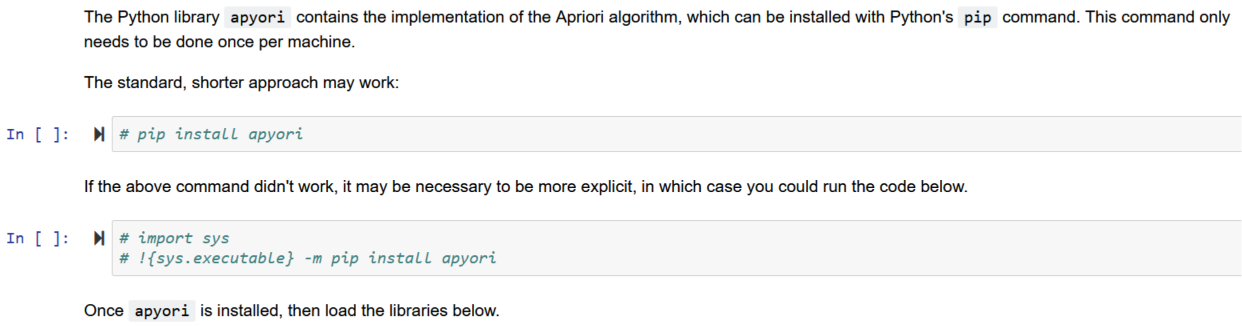
Total Points: 75

Due: 11/03/2022 11:59 PM

In this project, I invite you to do the following:

1. Import and prepare the dataset `Epub.csv`.
2. Apply the apriori algorithm to the data.
3. List the rules in a readable table.
4. Plot the rules.

Task 1: Import and prepare the dataset `Epub.csv` **(15 points)**



The Python library `apyori` contains the implementation of the Apriori algorithm, which can be installed with Python's `pip` command. This command only needs to be done once per machine.

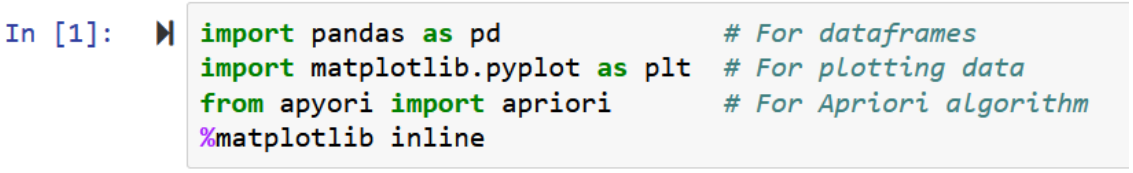
The standard, shorter approach may work:

```
In [ ]:  # pip install apyori
```

If the above command didn't work, it may be necessary to be more explicit, in which case you could run the code below.

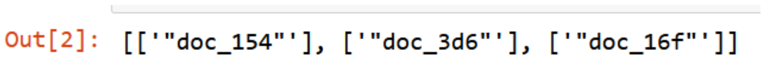
```
In [ ]:  # import sys
         # !{sys.executable} -m pip install apyori
```

Once `apyori` is installed, then load the libraries below.



```
In [1]:  import pandas as pd                  # For dataframes
         import matplotlib.pyplot as plt      # For plotting data
         from apyori import apriori           # For Apriori algorithm
         %matplotlib inline
```

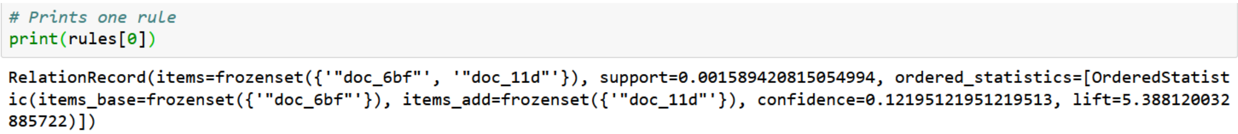
After that load the Epub dataset provided to you as **'data/Epub.csv'** file into the Jupyter Notebook, your code should open the dataset and converts it to list format, which is necessary for the `apriori()` function. You can name the list variable as transactions. Here is the output of the first three (03) itemset with the command transactions[:3]. Your output should match with mine.



```
Out[2]:  [['"doc_154"'], ['"doc_3d6"'], ['"doc_16f"']]
```

Task 2: Apply the **apriori** algorithm to the data. **(10 points)**

Call **apriori()** on transactions data. As parameters **apriori()** can take the minimum support, minimum confidence, minimum lift and minimum items in a transaction. Only the pairs of items that satisfy these criteria would be returned. For example:



```
# Prints one rule
print(rules[0])
```

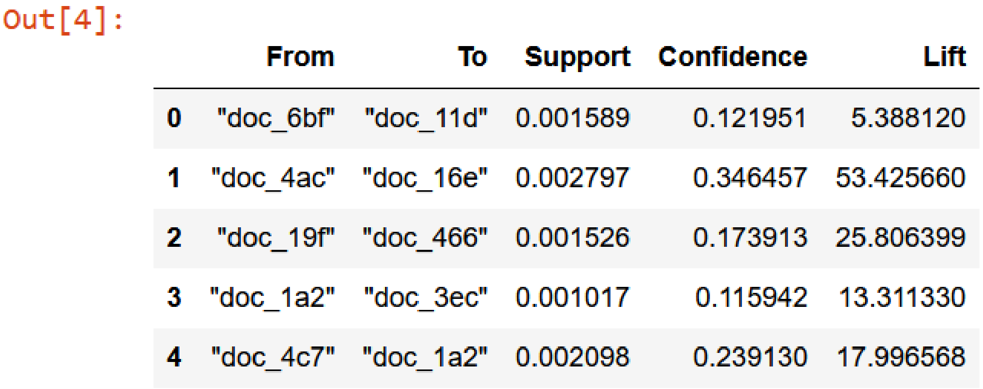
```
RelationRecord(items=frozenset({'"doc_6bf"', '"doc_11d"'}), support=0.001589420815054994, ordered_statistics=[OrderedStatist
ic(items_base=frozenset({'"doc_6bf"'}), items_add=frozenset({'"doc_11d"'}), confidence=0.12195121951219513, lift=5.388120032
885722)])
```

Note that here rules is the **apriori()** object on transactions data.

Task 3: List the rules in a readable table. **(25 points)**

The printed rule above is not very clear. You've to convert it to a more readable format. You'll add a **From** and **To** field to the DataFrame, to indicate a rule's antecedent and consequent respectively. Hence for a rule of the form **A->B**. The **From** will contain **A** and **To** will contain **B**. We'll also add the **Support**, **Confidence** and **Lift** corresponding to each rule in the DataFrame.

The output should be as follows for the first 5 rows: **(10 points)**



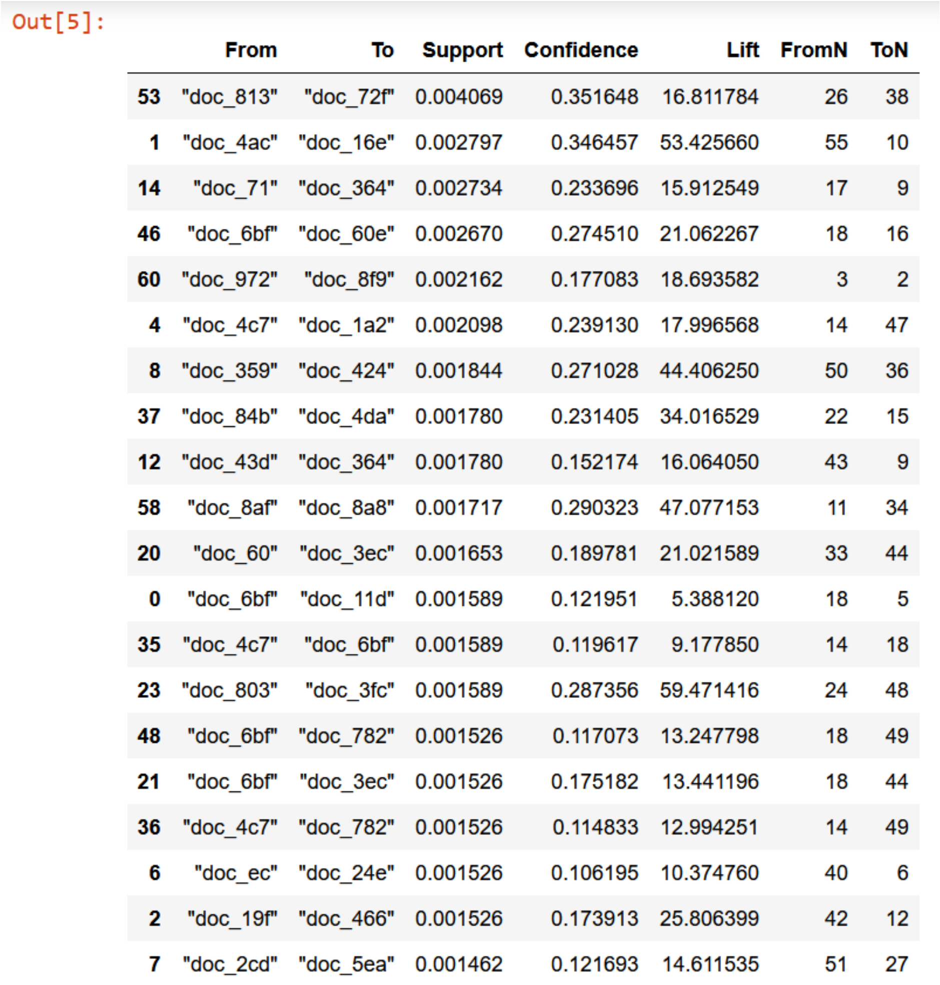
Out[4]:

	From	To	Support	Confidence	Lift
0	"doc_6bf"	"doc_11d"	0.001589	0.121951	5.388120
1	"doc_4ac"	"doc_16e"	0.002797	0.346457	53.425660
2	"doc_19f"	"doc_466"	0.001526	0.173913	25.806399
3	"doc_1a2"	"doc_3ec"	0.001017	0.115942	13.311330
4	"doc_4c7"	"doc_1a2"	0.002098	0.239130	17.996568

Next, List Rules with N's. Here you have to do the following: **(15 points)**

- Pick top rules sorted by Support, then **(3 points)**
- List of all items, then **(3 points)**
- Creates a mapping of items to numbers, then **(3 points)**
- Maps the items to numbers and adds the numeric **'FromN'** and **'ToN'** columns, then **(3 points)**
- Displays the top 20 association rules, sorted by Support **(3 points)**

The output should be as follows:



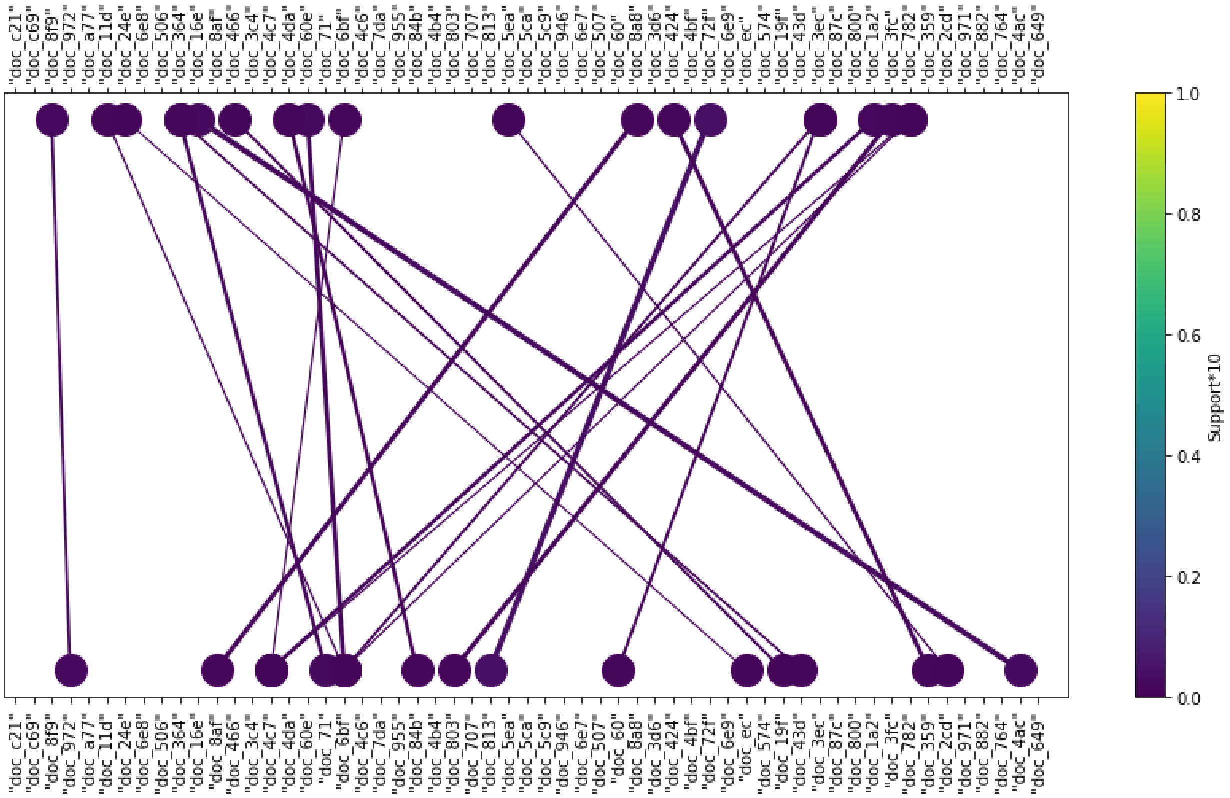
Out[5]:

	From	To	Support	Confidence	Lift	FromN	ToN
53	"doc_813"	"doc_72f"	0.004069	0.351648	16.811784	26	38
1	"doc_4ac"	"doc_16e"	0.002797	0.346457	53.425660	55	10
14	"doc_71"	"doc_364"	0.002734	0.233696	15.912549	17	9
46	"doc_6bf"	"doc_60e"	0.002670	0.274510	21.062267	18	16
60	"doc_972"	"doc_8f9"	0.002162	0.177083	18.693582	3	2
4	"doc_4c7"	"doc_1a2"	0.002098	0.239130	17.996568	14	47
8	"doc_359"	"doc_424"	0.001844	0.271028	44.406250	50	36
37	"doc_84b"	"doc_4da"	0.001780	0.231405	34.016529	22	15
12	"doc_43d"	"doc_364"	0.001780	0.152174	16.064050	43	9
58	"doc_8af"	"doc_8a8"	0.001717	0.290323	47.077153	11	34
20	"doc_60"	"doc_3ec"	0.001653	0.189781	21.021589	33	44
0	"doc_6bf"	"doc_11d"	0.001589	0.121951	5.388120	18	5
35	"doc_4c7"	"doc_6bf"	0.001589	0.119617	9.177850	14	18
23	"doc_803"	"doc_3fc"	0.001589	0.287356	59.471416	24	48
48	"doc_6bf"	"doc_782"	0.001526	0.117073	13.247798	18	49
21	"doc_6bf"	"doc_3ec"	0.001526	0.175182	13.441196	18	44
36	"doc_4c7"	"doc_782"	0.001526	0.114833	12.994251	14	49
6	"doc_ec"	"doc_24e"	0.001526	0.106195	10.374760	40	6
2	"doc_19f"	"doc_466"	0.001526	0.173913	25.806399	42	12
7	"doc_2cd"	"doc_5ea"	0.001462	0.121693	14.611535	51	27

Task 4: Plot the rules. **(25 points)**

Plot each pair of items in the rule. If a rule is A->B, then the item A is in the bottom row of the plot (y=0) and B is in the top row (y=1). The color of each line indicates the support of the rule multiplied by 100 (support*100). The width of each line is controlled by the confidence of each rule.

The output should be close to as follows:



The submission grading rubric is as follows (points out of 75 total):

Project element	**Points**
Task 1	15
Task 2	10
Task 3	25
Task 4	25

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 file. Submit the compressed file to Blackboard.

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.