MIE 524 Data Mining - Fall 2023
Assignment 1: Spark

This assignment is designed to allow students to get familiar with Spark and its Python interface (PySpark).

- Programming language: Python (Google Colab Environment)
- Due Date: September 25, 2023, 11:59PM EST

Marking scheme and requirements: Full marks will be given for (1) working, readable, reasonably efficient, documented code that achieves the assignment goals, (2) for providing appropriate answers to the questions in a Python notebook (named `MIE524_A1.ipynb`) committed, together with any associated output files, to the student's assignment repository, and (3) attendance in the post-assignment lab quiz.

Please note the plagiarism policy in the syllabus. If you borrow or modify any multiline snippets of code from the web, you are required to cite the URL in a comment above the code that you used. You do not need to cite tutorials or reference content that demonstrate how to use packages – you should certainly be making use of such content. *The use of generative AI solutions to generate code or text answers to the assignment questions is not allowed.*

What/how to submit your work:

1. All your code should be included in a notebook named `MIE524_A1.ipynb` that is provided in the cloned assignment repository.
2. Commit and push your work to your GitHub repository in order to submit it. Your last commit and push before the assignment deadline will be considered to be your submission. You can check your repository online to make sure that all required files have actually been committed and pushed to your repository.
3. A link to create a personal repository for this assignment is posted on Quercus.

Attribution: Q3 is inspired by contents from Stanford's CS247.

This assignment has 4 points in total and the point allocation is shown below:

- Q1: 1 points
- Q2: 1 point
- Q3: 2 points

Note: Partial points may be given for partial answers. Points will be deducted for missing or incomplete answers. Points could also be deducted for styling.

PART 1

Q1. Word Count in Spark

After successfully completing the first few steps provided in the notebook, you are now ready to work on the `plotsummaries.txt` data file, which contains plot summaries of more than 16 thousand books extracted from Wikipedia. You can find more information about this dataset here.

Task

Write a Spark program which outputs the number of words that start with each English letter (not including letters with accents and foreign language letters). This means that for every letter, we want to count the total number of (non-unique) words that start with a specific letter.

In your implementation, ignore the letter case, i.e., consider all words as lowercase. Also, you can ignore all words that start with a non-alphabetic character. If you encounter words broken as a result of new lines, e.g. "pro-ject" where the segment after the dash sign is on a new line, no special processing is needed and you can safely consider it as two words.

Output

The output of your application should be a text file named `Q1.txt` that contains one line per letter sorted alphabetically in the following format: `<letter>,<count>`
Important: Make sure that output text file (`Q1.txt`) is also included in the submission repository (i.e., you will need to download the created text file from Google Colab and push it to your GitHub's repository).

PART 2:

This second part of this assignment (Q2 and Q3) will focus on the Oxford Covid-19 Government Response Tracker (OxCGRT) dataset for the United States. Similar to the dataset you have seen in the lab, this dataset contains information on US states (instead of countries). The data and the documentation can be found here.

Q2. Computing Index Score with Spark

Write a Spark program to compute the Government Response Index Score described in the next sub-section for each state in the United States for each month in 2020 and 2021.

Indicator	Max. value (N_j)	Flag? (F_j)
C1	3 (0, 1, 2, 3)	yes=1
C2	3 (0, 1, 2, 3)	yes=1
C3	2 (0, 1, 2)	yes=1
C4	4 (0, 1, 2, 3, 4)	yes=1
C5	2 (0, 1, 2)	yes=1
C6	3 (0, 1, 2, 3)	yes=1
C7	2 (0, 1, 2)	yes=1
C8	4 (0, 1, 2, 3, 4)	no=0
E1	2 (0, 1, 2)	yes=1
E2	2 (0, 1, 2)	no=0
H1	2 (0, 1, 2)	yes=1
H2	3 (0, 1, 2, 3)	no=0
H3	2 (0, 1, 2)	no=0
H6	4 (0, 1, 2, 3, 4)	yes=1
H7	5 (0, 1, 2, 3, 4, 5)	yes=1
H8	3 (0, 1, 2, 3)	yes=1

Table 1: Maximum value and existence of flag for each indicator.

Algorithm

The government response index, $index_t$, is a proxy for how strict the policies imposed by the government are for a period t in a state. This index is computed as the average of the individual sub-indices, one for each of the 16 indicators as listed in Table 1.

$$index_t = \frac{1}{16}\sum_{j=1}^{16} I_{j,t}$$

The sub-index I_j for each indicator j is a function based on the following four parameters:

- the maximum value of the indicator (N_j)
- whether that indicator has a flag ($F_j = 1$ if the indicator has a flag variable, or 0 if the indicator does not have a flag variable)
- the recorded policy value on the ordinal scale ($v_{j,t}$)
- the recorded binary flag for that indicator ($f_{j,t}$)

$$I_{j,t} = 100\frac{v_{j,t} - 0.5(F_j - f_{j,t})}{N_j}$$

Since we are computing on a monthly basis, the indicator value for a month is then the majority value during the month. If there are ties, then we take the larger value (1 for flags). When no data is available for the entire period, you can safely assume the minimal value for that indicator.

Indicator	$v_{j,t}$	$f_{j,t}$	N_j	F_j	$I_{j,t}$
C1	2	1	3	yes=1	66.67
C2	no data	no data	3	yes=1	0.00
C3	2	0	2	yes=1	75.00
C4	2	0	4	yes=1	37.50
C5	0	null	2	yes=1	0.00
C6	1	0	3	yes=1	16.67
C7	1	1	2	yes=1	50.00
C8	3	N/A	4	no=0	75.00
E1	2	0	2	yes=1	75.00
E2	2	N/A	2	no=0	100.00
H1	2	0	2	yes=1	75.00
H2	3	N/A	3	no=0	100.00
H3	2	N/A	2	no=0	100.00
H6	2	0	4	yes=1	37.50
H7	2	1	5	yes=1	40.00
H8	2	1	3	yes=1	66.66

Table 2: An example for computing the government response index.

In the example given in Table 2, the score should be 57.18.

Output

The output of your application should be a text file named `Q2.txt` that contains one line per region (in alphabetical order) and period (from earliest to latest) in the following format:

```
<RegionName>,<Period(MM-YYYY)>,<GovernmentResponseIndex>
```

Important: Make sure that output text file (`Q2.txt`) is also included in the submission repository (i.e., you will need to download the created text file from Google Colab and push it to your GitHub's repository).

Q3. Association Rules

Association Rules can be used to discover interesting relations between government policies. This information could then be used to make policy recommendations.

Evaluation of item sets: Once you have found the frequent itemsets of a dataset, you need to choose a subset of them as your recommendations. A commonly used metric for measuring significance and interest for selecting rules for recommendations is **Confidence** defined as follows:

Confidence (denoted as $\operatorname{conf}(A \to B)$): *Confidence* is defined as the probability of occurrence of B in the basket if the basket already contains A:

$$\operatorname{conf}(A \to B) = \Pr(B|A),$$

where $\Pr(B|A)$ is the conditional probability of finding item set B given that item set A is present.

Task

a) Let's now only look at the first day of each month in our dataset (so we don't get too many duplicated policies assuming that a set of policies is applied for at least a month). Write a program in Spark that implements the A-priori algorithm to find policies which are frequently imposed together. Fix the support to s = 100 and find itemsets of size 2 and 3 (i.e. policies pairs/triplets that occur together at least 100 times are considered frequent).

b) For the pairs of items (X, Y) such that the support of $\{X, Y\}$ is at least 100, compute the *confidence* scores of the corresponding association rules: $X \Rightarrow Y$, $Y \Rightarrow X$. Sort the rules in decreasing order of confidence scores and list the top 5 rules in the writeup. Break ties, if any, by lexicographically increasing order on the left hand side of the rule.

c) For all triples (X, Y, Z) such that the support of $\{X, Y, Z\}$ is at least 100, compute the confidence scores of the corresponding association rules: $(X, Y) \Rightarrow Z$, $(X, Z) \Rightarrow Y$, $(Y, Z) \Rightarrow X$. Sort the rules in decreasing order of confidence scores and list the top 5 rules in the writeup. Order the left-hand-side pair lexicographically and break ties, if any, by lexicographical order of the first, then the second item in the pair.

Output

You will create two text files for this problem as listed below.

a) no output required

b) a text file named `Q3_b.txt` that contains 10 lines (5 lines for each rule) in the following format:

```
<X>,<Y>,<ConfidenceScore>
...
<Y>,<X>,<ConfidenceScore>
...
```

c) a text file named `Q3_c.txt` that contains 15 lines (5 lines for each rule) in the following format:

```
<(X,Y)>,<Z>,<ConfidenceScore>
...
<(X,Z)>,<Y>,<ConfidenceScore>
...
<(Y,Z)>,<X>,<ConfidenceScore>
...
```

Important: Make sure that output text files (`Q3_b.txt` and `Q3_c.txt`) are also included in the submission repository (i.e., you will need to download the created text file from Google Colab and push it to your GitHub's repository).