Project 1 (24T2)

Anomaly temperature detection (12 marks)

Background: The seriousness of global warming is underscored by its potential to disrupt ecosystems, exacerbate extreme weather patterns, accelerate sea level rise, and threaten food security and human well-being worldwide. Anomaly detection aims to examine specific data points and detect rare occurrences that seem suspicious because they're different from the established pattern of behaviors. Anomaly detection is a crucial technique in various industries and domains where detecting unusual patterns or behaviors can lead to significant benefits, such as preventing fraud, ensuring safety, optimizing processes, and saving costs.

Problem Definition: You are given a dataset of temperature collected in major Australian cities from 1995 to 2020 (only January). Each record in the dataset consists of a city name, a date (month day year), and a temperature value (*Fahrenheit*). A sample input file has been provided. Your task is to utilize MRJob to detect anomalies from the statistics for each city based on the following steps:

- For each city, calculate the monthly average temperature in each year (monthly average for short).
- For each city, calculate the overall average temperature from 1995 to 2020 (overall average for short).
- For each city, calculate the difference by which the monthly average exceeds the overall average.
- Report all the temperatures such that the monthly average exceeds the overall average for that city in that month over a given threshold τ (*Celsius degree*). Note: you need to convert *Fahrenheit* temperature value to *Celsius* temperature value using the formula **°C = (°F - 32) × 5/9.**

Output Format: The output should contain three fields: the city name, the year, and the gap, in the format of "<the city name>**\t<**the year>**,<**the difference>". The difference value should be at *Celsius* degree. The results should be sorted by city name alphabetically first and then by year in descending order. Given the sample input file and the threshold τ=0.5, the result should be like:

```
"Brisbane"      "1999,0.749597423510456"
"Brisbane"      "1998,0.6292270531400924"
"Brisbane"      "1995,0.9486714975845226"
"Melbourne"     "1999,4.324074074074073"
"Melbourne"     "1998,1.236111111111108"
```

One more test case is provided as well, and the value of τ is set to 0.3.

Code Format: The code template has been provided. Your code should take three parameters: the

input file, the output folder on HDFS, and the threshold value τ. We will also use more than 1 reducer to test your code. Assuming τ=0.3 and using 2 reducers, you need to use the command below to run your code:

```
$ python3 project1.py -r hadoop testcase.txt -o hdfs_output --jobconf myjob.settings.tau=0.3 --jobc
```

Note: You can access the value of τ in your program like "**tau = jobconf_from_env('myjob.settings.tau')**", and you need to import **jobconf_from_env** by "**from mrjob.compat import jobconf_from_env**" (see the code template).

Submission

Deadline: Monday 24th June 11:59:59 PM

If you need an extension, please apply for a special consideration via "myUNSW" first. You can submit multiple times before the due date and we will only mark your final submission. To prove successful submission, please take a screenshot as the assignment submission instructions show and keep it to yourself. If you have any problems with submissions, please email yi.xu10@student.unsw.edu.au.

Late submission penalty

5% reduction of your marks for up to 5 days, submissions delayed for over 5 days will be rejected.

Marking Criteria

- You must complete this assignment based on MRjob and Hadoop. Submissions only contain regular Python techniques will be marked as 0.
- You cannot simply emit all key-value pairs from mappers and buffer them in memory on reducers to do the task, and such a method will receive no more than 4 marks
- Submissions that cannot be compiled and run on Hadoop in the Ed environment will receive no more than 4 marks
- Submissions can be compiled on ED and run on Hadoop. => +4
- All the difference values in the output are correct. =>+1
- The order in the output is correct. =>+1 (**Note:** You only need to guarantee the order within each reducer output)
- The output format is correct. => +1
- Submissions correctly implement the combiner or in-mapper combing. => +1
- Submissions correctly implement order inversion (i.e., using special keys). => +1
- Submissions correctly implement secondary sort. => +1
- Submissions can produce the correct result using one MRStep. => +1
- Submissions can produce the correct result with multiple reducers. => +1 (**Note:** You do not

need to include 'mapreduce.job.reduces' in JOBCONF since the number of reducers will be received from the command)