xTech @ bip.

Task 3
Coding Application

Introduction & setup

This is a brief overview of the execution of Task 3.
Below, in addition to a small guide on how to prepare the working environment, are explained in detail how the task has been done and the reasons which led to the development of such a solution. Lastly, an analysis of the results and performances obtained by the different versions of the script is reported.

Python (version: *3.8.10*) has been chosen as the programming language for the execution of the task. This choice was mainly dictated by the fact that this language has a lot of modules for reading and analyzing files in "*.csv*" format.
In addition to the interpreter it's necessary to insert in this folder the '*Hr5m.csv*' file, downloadable at the following link.

IMPORTANT: *The downloaded file is in '.ZIP' format. To be able to get the script to work properly you need to manually extract the ".csv" file from that archive and insert it into this folder*.

This additional operation was made necessary because it was not possible to load the entire folder via the delivery form on VGen due to the size of the file.
If the file is not present, the following message will be displayed and the execution will be interrupted:

```
ATTENTION: Hr5m.csv file not found.
```

In addition to the language, it's recommended to install the Pandas module. For simplicity is recommended installation via *pip* from PyPI:

```
pip install pandas
```

This operation is recommended but *not strictly necessary*! The code in fact is designed to work even with the csv module, included by default in the *Python Standard Library*.
However, the use of Pandas for the management of large "*.csv*" files allows a significant increase in performance (see the *Performance* section for more details).

Once configured the environment you can run the script via terminal with the following command:

```
python task3.py
```

Output

Since the output format is not particularly complex, the result of the computation is directly printed in the terminal.
If the environment has been set correctly the result of the computation will be the as follow:

```
Starting...
Reading '.csv' file [time]
Iterate through data [time]

CALCULATING AVERAGE SALARY PER AGE (upper limit excluded) (*)
20-30: 119952.93
30-40: 120055.40
40-50: 119997.22
50-60: 119960.60
[time]

(*) In the age range 50-60 the upper limit of 60 years has been included in the calculation.

CALCULATING AVERAGE SALARY PER REGION
Midwest: 120023.60
South: 119990.11
West: 119952.66
Northeast: 119989.09
[time]

Operation completed in [time].
```

In addition to the results of the computation, for each operation performed, the time taken to perform it (expressed in seconds) is reported.

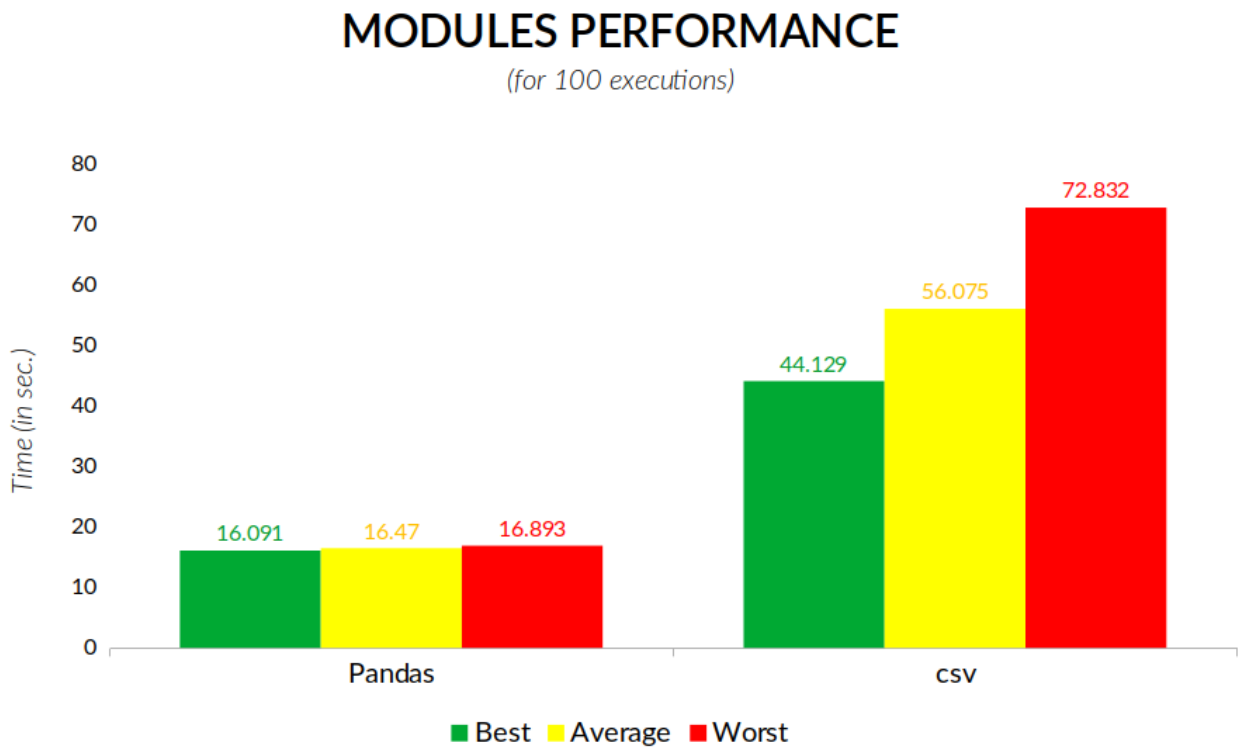
ANOMALIES: In case the pandas module has not been installed, the following message will be shown on top of the output of the script:

```
ATTENTION: No module named 'pandas'. Check out the documentation for more information.
```

This notice reminds the user that the Pandas module is not installed and that the current execution is using the csv library. It should be stressed again that this does not involve any kind of problem (except for a longer execution time). On the contrary, the message does not appear if the module is correctly installed.

Performance

Below is an analysis of the performance of Pandas and csv modules for interaction with "*.csv*" files.



As you can see from the chart, Pandas is clearly better than csv in terms of performance. To this must be added the fact that Pandas offers greater constancy, with a gap of just 0.8 seconds between its worst and best performance, while for its competitor the difference is almost 30 seconds.
These differences are due to the fact that the two modules operate on the file content differently:

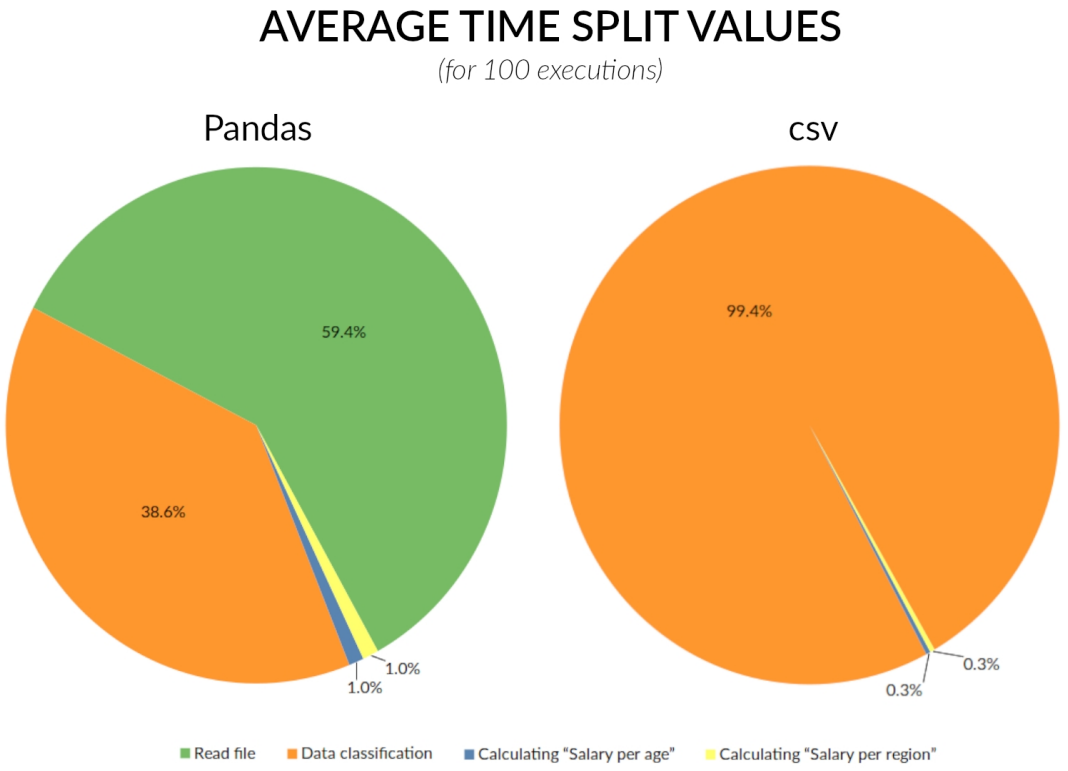
```
[...]
62   import csv
[...]
72   with open('Hr5m.csv') as data:
73       csv_data = csv.DictReader(data)
[...]
```

After opening the file, the csv module converts the entire contents of that file: each row (or record) of the file is converted to a dictionary (one of the most efficient data structures in Python in terms of access to information) and added to a list.

```
[...]
19   import pandas
[...]
29   df = pandas.read_csv('Hr5m.csv', usecols = ['Age in Yrs.', 'Salary', 'Region'])
[...]
```

Pandas instead, starting from the input file, creates an object containing records (on which successively it will go to iterate) composed from the single fields of interest for the task that must be executed (in the case of this task only columns "*Age in Yrs.*", "*Salary*" and "*Region*" are enough). This results in a considerable gain in time because the script will read only 3 of the more than 30

fields in the original file.
Finally, a look at how the execution times of the two modules are divided:



With Panads almost all of the work consists in reading the data from the file and their subsequent classification. Reading the file is the most expensive operation in terms of time, but it's also the one in which takes place a first "selection" of information (read only the fields of interest) and this saves time during teh data classifying. It's interesting to note that while performing two iterations on the information (first reading and then analysis), the execution times are drastically lower than the competitor.
For csv the thing is different, in fact in this module the file is opened and read immediately in a single solution (opening the file takes a few milliseconds but reading requires much more). As you can see from the graph, this operation is extremely time consuming and this is due to the fact that the entire file is analyzed in its entirety instead of being divided into useful and not useful information.
No significant difference in salaries calculation, as both Pandas and sv process the information in the same format and subsequent operations on the latter are not affected by either module.
It should however be taken into account that, while csv is a "lighter" module, designed to provide the developer a "ready-to-use" tool for interaction with "*.csv*" file (this is why it's included in Python by default) and is designed for simple operations on files of small size, Pandas has been designed for data analysis operations and is therefore optimized to perform more complex tasks even on files of considerable size.