**CCT College Dublin**

**Repeat Assessment Cover Page**

*To be provided separately as a word doc for students to include with every submission*

| **Module Title:** | *Repeat MSC\_DA\_Integr\_CA2\_Sem1* Programming for DA |
| --- | --- |
| **Assessment Title:** | *MSC\_DA\_Integr\_Repeat\_Sem1* |
| **Lecturer Name:** | **Marina Iantorno, David McQuaid, Dr. Muhammad Iqbal, Sam Weiss** |
| **Student Full Name:** | Adam Goszczynski |
| **Student Number:** | sba21182 |
| **Assessment Due Date:** | *6th August 2023* |
| **Date of Submission:** | *6th August 2023* |

Repeat CA2 Programming for DA

We must consider specific data sources and the accompanying needs to compare and contrast data processing and data aggregation libraries and approaches. I'll give a general overview of libraries and techniques used for data processing and aggregation in the provided task.

1. **Data Processing**

Cleaning, converting, and preparing raw data for analysis is what data processing entails. The following are examples of common data processing libraries and techniques:

a) Pandas:

Pandas is a popular Python data manipulation and analysis library. It includes DataFrame and Series data structures, which make data processing more straightforward and efficient. Pandas provides a variety of operations for filtering, combining, reshaping, and dealing with missing data. It works nicely with other libraries such as NumPy and Matplotlib and is well-suited for organised tabular data.

b) Dask:

Dask is a flexible parallel computing package that enables for the processing of datasets bigger than memory by distributing computations across numerous cores or computers. It includes DataFrame and Array structures that are similar to Pandas and NumPy interfaces, making the switch from Pandas to Dask for large-scale data processing simple. Dask is appropriate for big data applications since it can handle out-of-core and distributed processing.

c) Modin:

Modin is a newer library that seeks to overcome Pandas' memory and processing restrictions. It makes use of Dask's distributed computing capabilities to deliver faster and more scalable data processing. Modin has a similar to Pandas API, therefore work created for Pandas can be readily ported to Modin with minor changes.

1. **Data Aggregation:**

The process of summarising and reducing data in order to gain useful insights. It is often utilised in statistics computation, group-wise procedures, and data summarization. Among the relevant libraries and methodologies are:

* **NumPy** is a key Python package for numerical computing. It provides sophisticated array-based operations and mathematical calculations. The aggregation functions in NumPy (for example, sum, mean, min, max) quickly compute statistics across arrays and support vectorized operations. It works nicely with Pandas DataFrames and is useful for simple aggregating operations.
* **SQL** (Structured Query Language): SQL is a domain-specific language used for managing and querying relational databases. Not used in this assignment.
* Function like **groupby()**
* Tools such as **matplotlib, seaborn, and plotly** aid in the visualisation of aggregated data in interesting ways.

Justification for the Libraries/Techniques Selected:

The libraries and techniques used for data processing and aggregation are determined by the individual data sources, data size, and analytical needs. For this analysis, it was necessary to use mathematical and statistical libraries another way it would be difficult to finalize the outcome. Here are some justifications for the libraries/techniques chosen:

**Pandas**: Pandas is a popular data processing tool due to its ease of use and comprehensive range of features. It is appropriate for structured data such as CSV files and Excel sheets. Pandas delivers rapid and efficient processing capabilities for tiny datasets.

**NumPy**: NumPy is a Python library that is useful for array-based numerical computations and simple data aggregation applications. It is a fundamental library for many people.

Other libraries used:

**scipy.stats as stats**  SciPy for scientific and technical computing

**statsmodels.stats import weightstats**  Statsmodels for statistical tests

**json**  Importing the json library for working with JSON data

**time**  Importing the time library for time-related functionalities

**textblob import TextBlob**  Importing the TextBlob class from the textblob library to

work with API.

Assignments include Machine Learning algorithms and libraries described in the attached Jupiter Notebook.

In this assignment, Pandas and Numpy were used among others. Modin, fail and turned out that the Pandas library is faster, which will be discussed in the next section. Given the stable performance and already proven integrated libraries, I decided to use Pandas because the collections they provided were small enough to use this method rather than Dask or Modin. In case the data would exceed the computer's memory, I could use the Medin library, which can use the hard drive in case of insufficient RAM. Pandas also met all the requirements I needed to perform the analysis, such as:

* Data Cleaning: Pandas includes routines for dealing with missing data, imputation, and data type conversions, making data cleaning chores simple.
* Data Transformation: Pandas supports data filtering, sorting, reshaping, and combining, allowing users to alter data in a variety of ways.
* Pandas enable group-wise operations and statistical aggregation, allowing for speedy and efficient data summarization.
* Pandas provide extensive time series features, making it a popular choice for time-based data analysis.
* Pandas interfaces with other libraries like as NumPy, Matplotlib, and SciPy to enhance its data analysis capabilities.

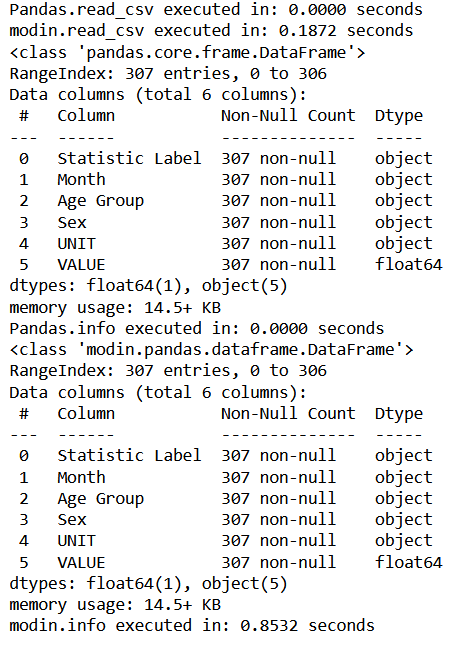
In this case, Pandas was my choice because is a mature and feature-rich library that specialises in the memory handling of small to medium-sized datasets. It is adaptable and frequently utilised in the data analysis.

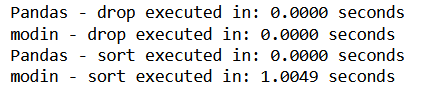
Pandas also allows you to upload different types of data files. Json, csv and excel formats were used in this analysis.

Modin, on the other hand, is intended to support Pandas constraints when dealing with datasets that are larger than memory. It can significantly boost data processing performance, especially on multi-core devices.

**Testing & Optimisation:**

Python provides built-in modules such as time and timeit, which allows measuring the time taken for code to execution. In our analysis, I did check function executing in Pandas and Modin. The results tipped the scales in favor of Pandas and we can see them in the snippets below :





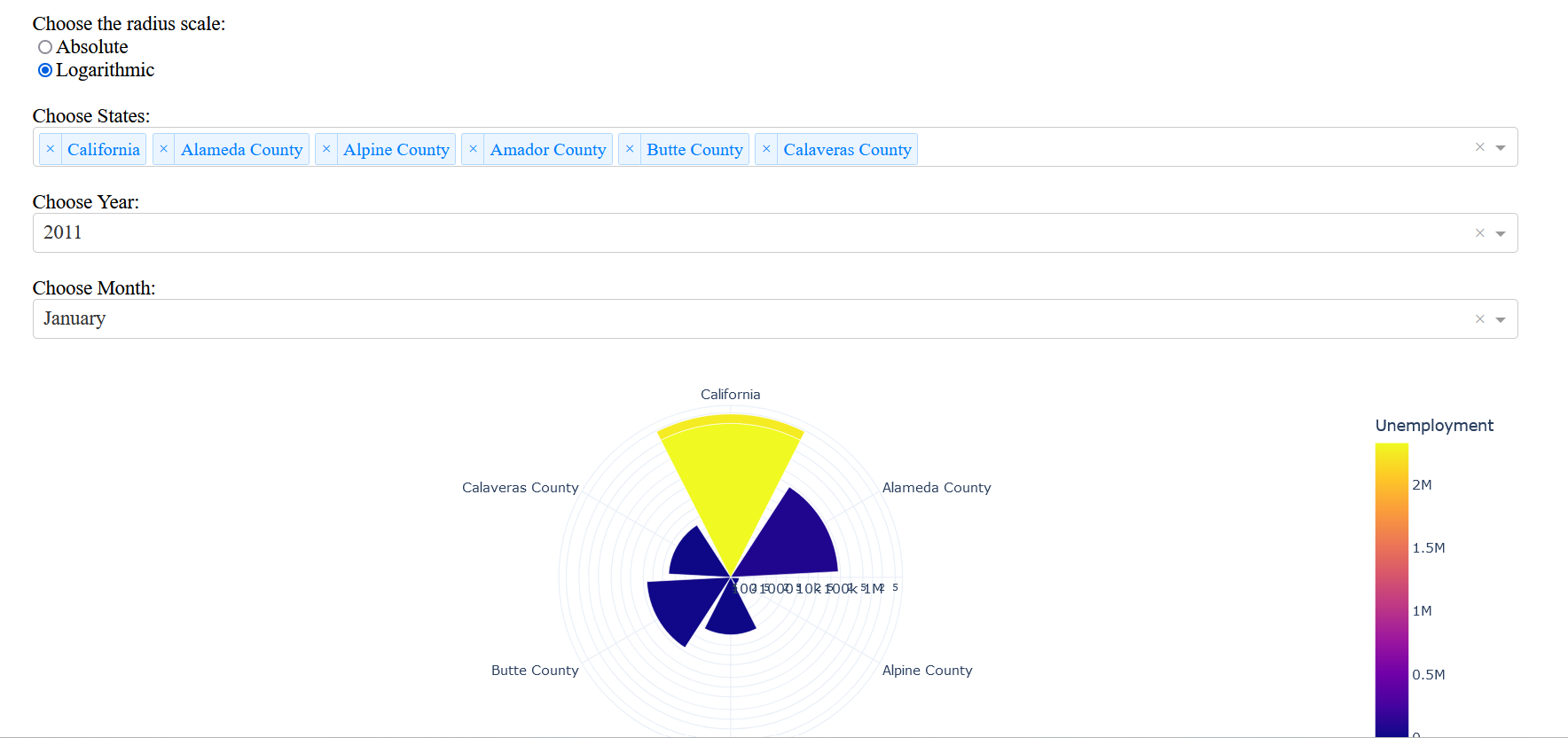
In addition, to optimize the code, I tried to use functions so as not to duplicate the code unnecessarily and use the same code only if necessary. Also, data frames that were no longer needed were removed from memory. After the upload, variables that were not needed were also removed to free up memory.

To verify that the code works as intended, some snippets were checked and visualization tools were used. As it is not a complicated program but an analysis, there was no need to use methods such as error handling. With more serious errors, Pandas itself would return an error that allowed the code to be supported. The github option was also used to have a control version, and the bark would let you to revert to more desirable outcomes if something went wrong. Furthermore, the code was written in short snippets to allow for error detection on the fly. The results were checked by using functions like **.head()**, **print()** statements or for example function which was created **def** explore\_data**(data)**. These snippets were later transformed into functions to optimise operation.

Control version address [HERE](https://github.com/Adamgosz/CA2Repeat) .

**Dash**

*Dash is running on* [*http://127.0.0.1:8050/*](http://127.0.0.1:8050/)



References

1. *Visualization with Python* (no date) *Matplotlib*. Available at: https://matplotlib.org/ (Accessed: August 06 2023).
2. *An introduction to seaborn#* (no date) *An introduction to seaborn - seaborn 0.12.2 documentation*. Available at: https://seaborn.pydata.org/tutorial/introduction (Accessed: August 06 2023).
3. *About Us* (no date) *NumPy*. Available at: https://numpy.org/about/ (Accessed: August 06 2023).
4. *About Pandas* *pandas*. Available at: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.aggregate.html (Accessed: August 06 2023).
5. *Learn* (no date) *scikit*. Available at: https://scikit-learn.org/stable/ (Accessed: April 13, 2023).
6. *Scale the python tools you love* (no date) *Dask*. Available at: https://www.dask.org/ (Accessed: 06 August 2023).
7. *Scale your pandas workflow by changing a single line of code#* (no date) *Scale your pandas workflow by changing a single line of code - Modin 0.23.0+0.g6a5416c7.dirty documentation*. Available at: https://modin.readthedocs.io/en/stable/ (Accessed: 06 August 2023).
8. Chawla, A. (2022) *Five killer optimization techniques every pandas user should know*, *Medium*. Available at: https://towardsdatascience.com/five-killer-optimization-techniques-every-pandas-user-should-know-266662bd1163 (Accessed: 06 August 2023).
9. *Dash example index* (no date) *Dash*. Available at: https://dash-example-index.herokuapp.com/ (Accessed: 06 August 2023).

Source of data

1. <https://www.cso.ie/en/releasesandpublications/ep/p-mue/monthlyunemploymentjuly2023/data/>
2. <https://data.gov.ie/dataset/nsa55-mean-hourly-earningsweekly-earnings-and-weekly-paid-hours?package_type=dataset>
3. <https://catalog.data.gov/dataset/local-area-unemployment-statistics-laus/resource/1dc85695-4216-4298-bf1f-4e78f88ee936>