2023/24

HIGH PERFORMANCE COMPUTATIONAL INFRASTRUCTURE

SALARIES AND JOBS IN DATA-RELATED CAREERS

SUMMIYA ALAM
2340875

1. Introduction

The rise of big data has led to a revolution in the methods of storing, managing, and manipulating data. Developers and researchers have devised a range of techniques to manage mega or petabytes of data, or even larger volumes. Hadoop, an open-source platform, is designed for storing such vast amounts of data. Hadoop filing system (HDFS) takes care of data storage. MapReduce is programming paradigm, that is used for processing the data stored in HDFS. MapReduce deals with data that can essentially be divided into smaller distributed chunks. Due to parallel computing, along with the flexibility and reliability offered by MapReduce, numerous major corporations like Yahoo, Facebook, Amazon, Google, and others have been using this model (Karloff, Suri and Sergei Vassilvitskii, 2010).

The objective of this practical project is to utilize a case study as an effective demonstration of employing a large-scale data storage and processing platform in real-world scenarios. Our aim is to determine the number of AI professionals such *Data Scientists, Data Engineers and Data Architecture*, employed within the AI industry, categorized by their geographic locations and salaries. The problem statement and solution are organized in a way that supports the implementation of a MapReduce framework.

The rest of the document contains definition of problem statement and associated dataset. There is design and implement section followed by results and discussion.

1.1. Problem Statement

As a student pursuing Data Science and Analytics, the potential employment opportunities and salaries within the field of data science hold significant importance to me, as well as to anyone else with even a remote interest in this area of study or profession. Before explaining the dataset and its feature, it is important to formulate appropriate research question. The main goal or research question is

- **What is the distribution of different jobs employed within the AI industry, categorized by their geographical locations and salaries?**

Distribution, in this context, pertains to mathematical measures indicating the density of employees in the UK.

The objectives to answer above research questions are

- Identify all the employes who are earning in six figures or more annually.
- Filter out Data Scientists among different professions
- Find out the geographical regions of those high earning Data Scientist.
- Next, triangulate the ones that are working specifically in United Kingdom as a Data related worker with annual salary of six figures or more and find out few measures based on total count (count, percentage and Average).

As evident from objectives and research question, principal focus of this study is profession, location and salary of employees.

1.2. Associated Dataset

The focus of research in this study is *Jobs and salaries in Data Science* of employers employed in related professions and countries. The study relies on an open-source dataset obtained from Kaggle (Qaasim, n.d.),

focusing on salary patterns across various data-related positions from all over the world. The data presented in the dataset spans from years 2020 to 2023. The original dataset is a csv file with twelve parameters with almost 10,000 records. This may constitute a relatively smaller dataset compared to the capabilities of Hadoop framework. Owing to the limitations of personal laptop, files containing millions of records are exceedingly challenging to open, manage, manipulate and store. A lot of prediction, regression and linear analysis has been done on this dataset (Qaasim, n.d.). An entirely separate approach will be adopted in this project to look at data from different perspective. Hadoop and MapReduce will be applied to learn insights and identify the problem to answering the research question. Even though, currently the dataset is of few thousand observations, Hadoop and MapReduce would facilitate in offering scalability and efficiency by execution in a distributed environment, if the dataset is to be extended to more data points by adding employes from before 2020.

The data dictionary of *Jobs in Data* is provided in Table 1. This table contains the columns that are relevant to this project.

Name	Description	Domian
Job_category	Range of jobs related to AI industry such as 'Data Engineering', 'Machine learning analyst' etc.	Nominal
Job_title	Job titles related to Data category of work such as 'Data Scientist', 'Data Engineer' and 'Data Analyst'	Nominal
Salary_currency	Currency in which salary is paid	Nominal
Salary_in_usd	Annual gross salary after conversion to United States dollars	Integer
Employee_residence	Country of residence of employee	Nominal

Table 1: Data dictionary for 'jobs in Data'

2. Design and Implementation

2.1. Algorithm Design:

This section describes the complete design of proposed solution by utilising Hadoop and MapReduce to run the data in a distributed environment.

The key element behind the algorithm design is "*single-stage MapReduce job with multiple reducers*" (Jayalath, Stephen and Eugster, 2014). This refers to the design where we are using single MapReduce job but the output of mapper is passed through two different reducer jobs. This configuration, one mapper and two reducers, is highly effective in cases when the data is huge and it is required to distribute the load on multiple reducers**.** In this project, load distribution is not applicable as dataset is small. The use of two reducers in this context aims to efficiently allocate and distinguish the tasks so that the research question and objectives (section 1.1) can be clearly identified in design and implementation.

Figure 1 shows the basic flow of proposed design

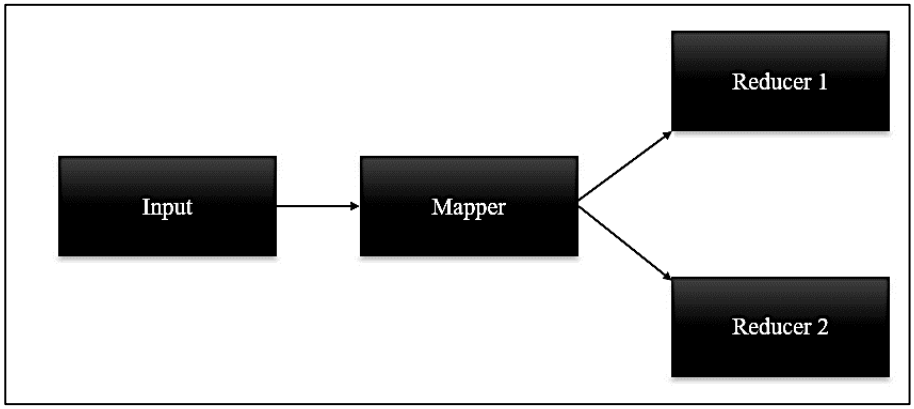

Figure 1: Basic Flow of proposed design

Functionality of Mapper and reducers are

- Data (csv file) is fed into the mapper. The file will be stored on Hadoop filing system called HDFS.
- Salary-mapper will produce the key value pairs by performing data splitting and delineation. In this study, the key-value pairs are Job_title and employee_residence based on a person's salary. If the yearly salary amounts to six figures or more, create the key-value pairs of Job_title and employee_residance. The key is job title and country of residence is the value. The key-value pairs will be shuffled and sorted to be fed into the reducer.
- Datascientist-reducer takes input of these shuffled and sorted key-value pairs. It will filter-out all the jobs, and generate an output for only Data Scientists working all over the world.
- UK-distribution-reducer will be executed with the salary-mapper to find all data related jobs in a targeted country. It will show the distribution of AI employees working in United Kingdom earning 6 figures or more annually. The output is percentage, count and average of workers in UK.

Above details are in Figure 2.

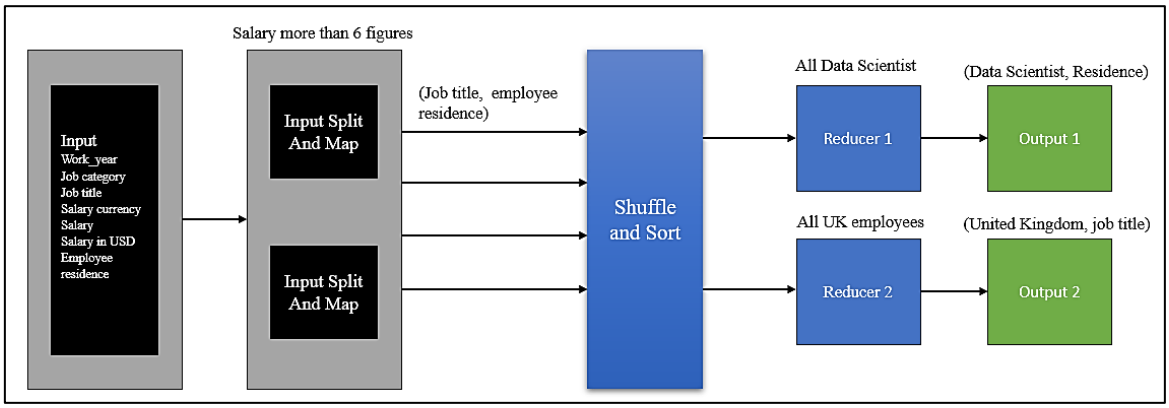

Figure 2: Detailed Design of Jobs in AI

In an alternative and simpler approach, the algorithm can be designed with a mapper and only one reducer. In this context, the tasks carried out by mapper and reducer will be assigned as follows

- The map function will remain the same. It will extract the salary_in_usd, job_title and employee_residence after splitting and delineating values. Key-value pairs will be generated based on salary. If the annual salary is in 6 figures or above, generate the key value pairs of job and residence. The key is Job_title, and the corresponding value is the country of residence.
- The difference is in reducer function. The reducer function will filter out the residence and generate a list of employes living in UK, working as a data Scientist and earning more than 6 figures. The distribution measures (count, average and percentage) generated by second reducer function can be incorporated in this single reducer.

Reflection on both methodologies

Both of the designs work well and can be easily implemented in current settings and dataset. However, first design is far more scalable. When the data grows, functional requirements also evolve. Instead of counting or average calculation, we might need to perform complicated tasks. In such scenarios, it is preferable to utilize a second reducer. We will be implementing one mapper with two reducers.

2.2. Implementation:

For implementation of design defined in section (2.1), python is used. Python 3.12 (64-bit) is required run python files for mapper and reducer. These can be executed in a distributed setting of Hadoop filing system (HDFS).

Figure 3 shows the python implementation of salary-mapper. The mapper function takes in the data from csv file, split() and strip() the value using comma and then fetch the values of salary, job_title, and country based on the location of their indices. Salary is salary_in_usd, job_title is job_title column, and country is employee_residence column. The subsequent part of mapper creates key value pairs of job_title and country, based on an "if" condition to exclusively produce pairs for employees with annual salaries of six figures or higher.

```python
import sys
import io


def process_input_mapper_job_country(line):
    #Remover white spaces from start and end
    values = line.strip()
    # Splitting the values by tab
    columns = values.split(',')

    # Assuming the job title is in the second column and country in the third column
    # Change the indices accordingly based on your dataset
    job_title = columns[1].strip()
    country = columns[10].strip()
    salary = columns[5].strip

    # job_title, country = line.split('\t')

    if (len(str(salary)) >= 6):
        print(f'{job_title}\t{country}')

def main():
    # Read from standard input with specified encoding
    input_stream = io.TextIOWrapper(sys.stdin.buffer, encoding='latin1')
    for line in input_stream:
        # map station and temperature data
        process_input_mapper_job_country(line)


if __name__ == '__main__':
    main()

```

Figure 3: Salary-mapper

The first reducer function DataScientist-reducer, presented in figure 4, receives the output of mapper as its input. It takes key value pairs from mapper-salary, performs required split() and strip() actions on each line. Subsequently, it traverses through each job_title, emitting the countries for all occurrences of "Data Scientist" job_titles. The final output is all the occurrences of data scientists in each regional location from the data.

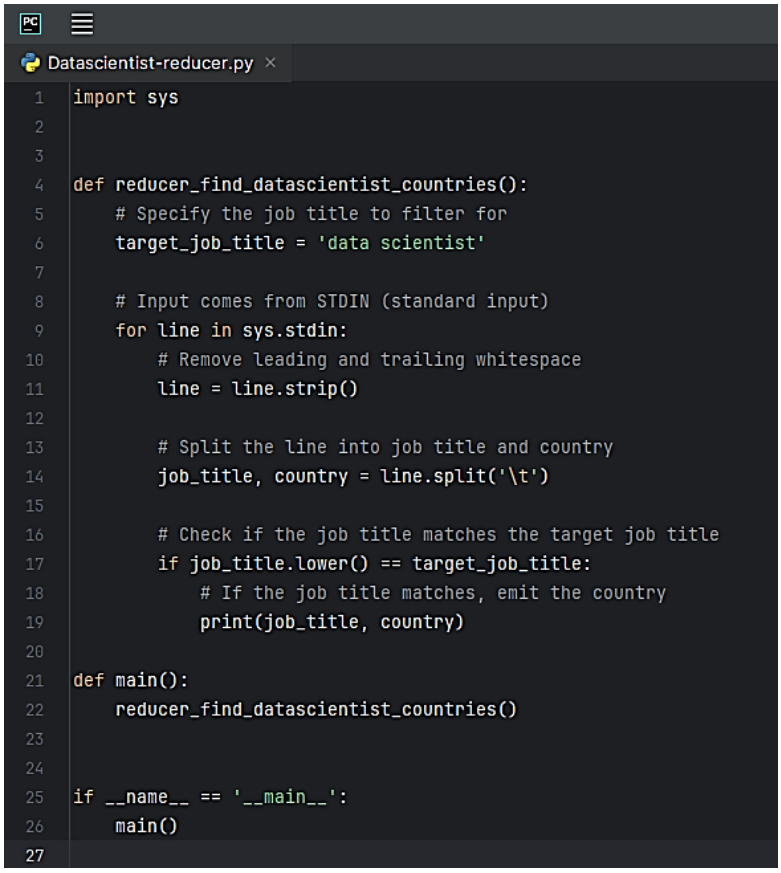

Figure 4: DataScientist reducer

The second reducer function, shown in figure 5, provides distribution measures for total number of AI professionals residing in United Kingdom. The code starts with receiving key value pairs from salary-mapper. Select a target country which is United Kingdom. Using for loop, which is subjected to a condition, iterate through all the key value pairs to acquire total number of AI professionals only present in UK. All the employes from other countries are filtered out. Use a counter variable to get the total number of AI professionals residing in UK. Instead of emitting key value pairs, this second reducer performs few mathematical calculations including the total count, mean of employees living in UK and proportion of UK professionals against other regional works. These measures are calculated using the total count of UK employees and the total number of records present in the data.

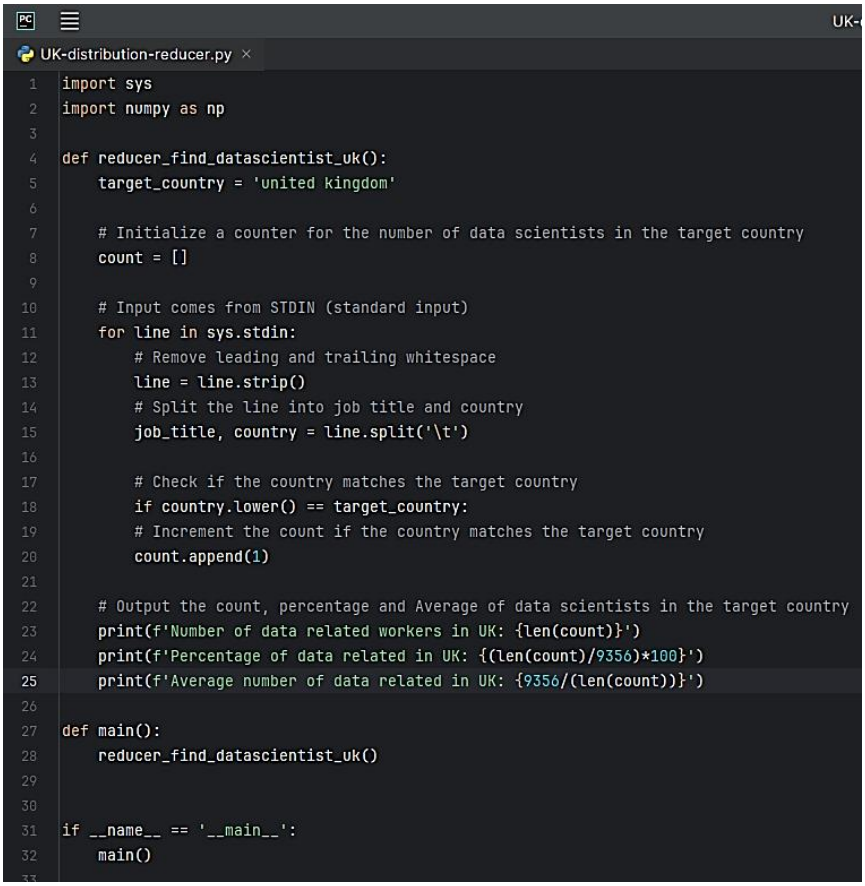

Figure 5: UK-distribution-reducer

Execution of MapReduce

The map and reduce functions can be executed in with HDFS using Linux commands. There are multiple ways to run these commands. Either one can directly use the Linux operating system, or use terminal for windows or MAC to access SSH Shell for remotely accessing the server to perform file operations using HDFS. For this project, Google Colab is used to set up Hadoop and execute the written code for mapper and reducer.

First step is to download the Hadoop in virtual Linux machine of google Colab. Since it's a compressed file, it needs to be unzipped before it can be utilized. Now copy those unzipped files to another directory /usr/local.

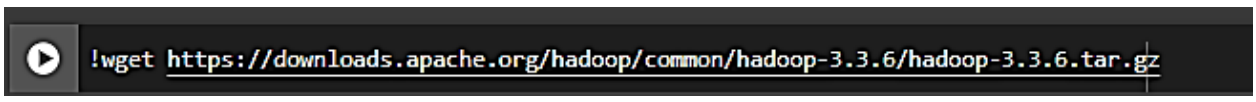

Figure 6: Hadoop download

Next step is to use java SDK as Hadoop is primarily written in Java. Since Google Colab comes with Java preinstalled, all that's needed is to find the SDK. Once located, one can set the Java path variable accordingly. Additionally, another global path variable is updated to include the java path. This increases the efficiency of program execution and provide smooth running of Hadoop without specifying the path again and again.

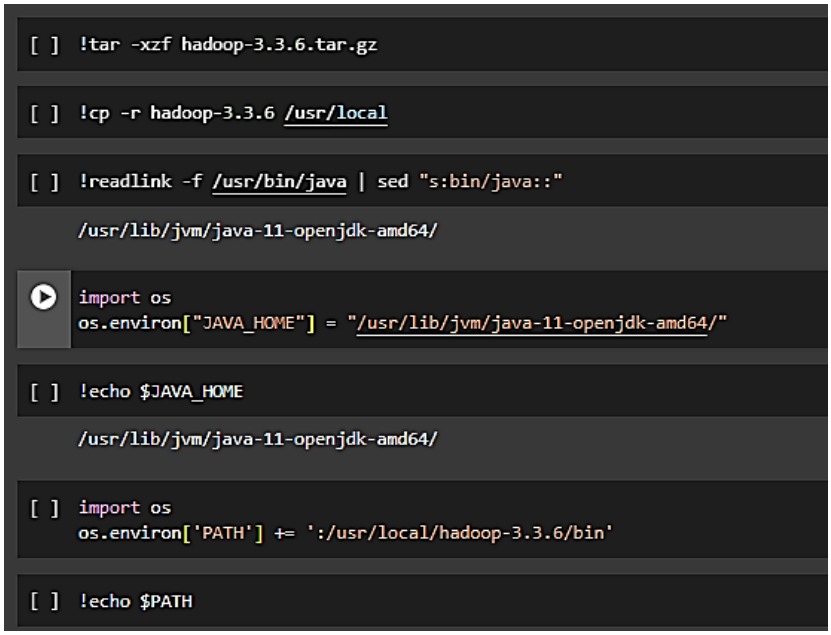

Figure 7: Path setting

After setting the environment, prepare for mapper reducer execution. Upload the CSV data file, along with the mapper and reducer function files, to the "content" folder in Google Colab. The final step remaining is to execute the Linux command for MapReduce functions. The command has following instances

- -files: it instructs the framework to combine mapper reducer files to execute them together
- -input: Here, we provide location of our input to MapReduce. For this project, it is the CSV file present in the content folder
- -output: This is where output is saved after the execution
- -mapper: Specify the mapper file uploaded in the content folder
- -reducer: Specify the reducer file uploaded in the content folder. First, I will run the mapper and reducer 1.

```
! hadoop jar /usr/local/hadoop-3.3.6/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar \
-Dmapreduce.job.reduces=2 \
-files /content/salary-mapper.py,/content/Datascientist-reducer.py \
-input /content/jobs_in_data.csv \
-output /content/output/test-16 \
-mapper 'python salary-mapper.py' \
-reducer 'python Datascientist-reducer.py'
```

Figure 8: salary mapper with Datascientist Reducer

After execution of mapper and reducer 1, I will run exactly the similar command with same mapper but different reducer.

```
! hadoop jar /usr/local/hadoop-3.3.6/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar \
-Dmapreduce.job.reduces=2 \
-files /content/salary-mapper.py,/content/Datascientist-reducer.py,/content/UK-distribution-reducer.py \
-input /content/jobs_in_data.csv \
-output /content/output/test-2 \
-mapper 'python salary-mapper.py' \
-reducer 'python UK-distribution-reducer.py'
```

Figure 9: salary-mapper with UK-distribution-reducer

The results are explained in next section.

3. Results, evaluation and discussion

3.1. Results

Upon completion of the code, it is time to execute it and observe the results. The resultant txt file is generated and saved in the output directory mentioned in the execution command. In our scenario, two output files are produced. One txt document is for first reducer output and the other file is output of second reducer.

Let's state the results based on objectives described in section 1.1.

Objective 1: Identify all the employes who are earning in six figures or more annually.

This is accomplished in the mapper as stated in section 2.2. The iterator generates all the key value pairs where salary is in 6 figures or more.

Objective 2: Filter out Data Scientists among different professions

This objective aims to filter out all the 'data scientists' among different professions or job_titles. After the mapper feeds the key value pairs of job_title and employeE_residence to the reducer 1, it goes through the list and identify all the Data Scientists. The final output is displayed in text file with all the data scientist as shown in Figure 10.

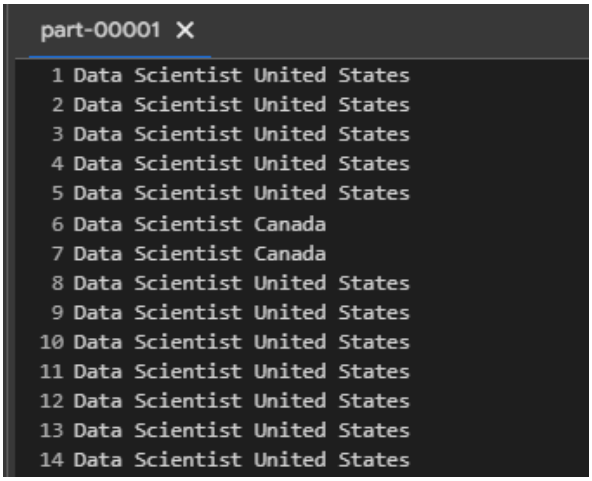

Figure 10: Data Scientists in different countries

Objective 3: Find out the geographical regions of those high earning Data Scientist.

This objective is fulfilled by generating key value pairs of job_title and employee_residence from the mapper. Each generated pair is high earning Data Scientist. Refer to Figure 10.

Objective 4: Next, triangulate the workers that are working specifically in United Kingdom as a Data related worker with annual salary of six figures or more and find out few measures based on total count (count, percentage and Average).

Second reducer generates this output. After receiving key value pairs from the mapper, it filters out the country. The counter for UK and the total rows in the csv files is used to find out related measures such as the average number and percentage of employees operating from UK. The results are shown in figure 11.

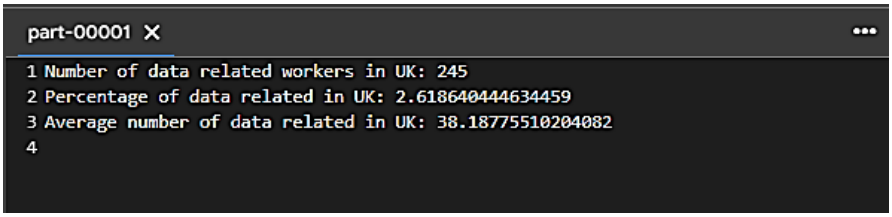

Figure 11: Count, Average and Percentage of Data Professionals

The research question is

What is the distribution of different jobs employed within the AI industry, categorized by their geographical locations and salaries?

The results generated from reducer 1 and reducer 2 (figures 10 and 11) answer the research question. Based on high salary, it first shows the number of Data Scientist and then all the AI works in UK. The calculated measures show the outcome based on location and salary. For each reducer job, two output files are generated

3.2. Evaluation of results

To evaluate the results, another implantation is carried out in R-language. The outcome of each objective is generated using R code and results are measured. The implantation details and evaluation are performed below

Objective 1: Identify all the employes who are earning in six figures or more annually.

The code below filters the salary and generates the job_title and location of each employee. The csv file is loaded through read,csv function of R as a dataframe. Using subset technique, the whole data frame is filtered to generate list of employees earning in 6 figures or more. Just like mapper, the subset variable (called high_earners) include two variables Job_title and employee_residence. These results are identical to the ones generated from the mapper as shown in the figure 12.

```r
# Filter individuals earning six figures or more and select only Country and Job columns
high_earners <- df[df$salary_in_usd >= 100000, c("employee_residence", "job_title")]

# Print filtered dataset
print(high_earners)
```

	employee_residence <chr>	job_title <chr>
2	United States	Data Architect
4	United States	Data Scientist
6	United States	Data Scientist
7	United States	Data Scientist
8	United States	Machine Learning Researcher
9	United States	Machine Learning Researcher
10	United States	Data Engineer
11	United States	Data Engineer
12	United States	Machine Learning Engineer
13	United States	Machine Learning Engineer

Figure 12: List of employees earning in six figures or more

Objective 2: Filter out Data Scientists among different professions

This task was carried out by the first reducer in previous implementation. In R, high_earners dataset containing job_title and employee_residence is further filtered down to identify all the Data Scientists in the list. Here, this data set is called “Data_scientists”. The code and output are shown in the figure 13.

```{r}
Data_Scientists <- high_earners[high_earners$job_title == "Data Scientist", c("employee_residence", "job_title")]

Data_Scientists
```

	employee_residence <chr>	job_title <chr>
400	United States	Data Scientist
401	United States	Data Scientist
414	Canada	Data Scientist
415	Canada	Data Scientist
422	United States	Data Scientist
423	United States	Data Scientist
424	United States	Data Scientist
430	United States	Data Scientist
434	United States	Data Scientist
435	United States	Data Scientist

Figure 13: Data Scientists among different professions

Objective 3: Find out the geographical regions of those high earning Data Scientist.

The third objective is achieved alongside the second objective in R. The filtered "Data_Scientists" dataset includes the geographical locations of employees as well. Evident from figure 13.

Objective 4: Next, triangulate the workers that are working specifically in United Kingdom as a Data related worker with annual salary of six figures or more and find out few measures based on total count (count, percentage and Average).

In R, I have reused high_earners dataset for this objective. In map reduce, this is the input from mapper function. Using dataset filtering techniques, all the employees from United Kingdom are separated in a variable called 'united_kingdom'. For the calculation of measure, I have used nrow() function to have the total count of employees in United Kingdom. To calculate percentage and average, I have used counts from original dataset and united_kingdom dataset using the nrow() function. The results are shown in Figure 14.

```{r}
count <- nrow(united_kingdom)
percentage <-  (count / df_count ) * 100
average <- count / df_count

paste("Total number of UK employees {count}", count)
paste("Percentage of employees working in UK {percent}", percentage)
paste("Average of employees working in UK {average}", average)
```

```
[1] "Total number of UK employees {count} 184"
[1] "Percentage of employees working in UK {percent} 1.96686264029931"
[1] "Average of employees working in UK {average} 0.0196686264029931"
```

Figure 14: Data Professionals living in UK.

The results of map reduce are calculated using another implementation in R. There are two distributions generated by Map-reduce implementation. While the results in R matches the mapper-reducer output, there is some inconsistency with calculated values.

There are two visual representations, showing job_titles in United Kingdom (the final output) and the countries where Data Scientists are working.

Below graph shows available jobs in AI profession

```r
library(ggplot2)

ggplot(united_kingdom, aes(x = job_title)) +
  geom_bar(fill = "gray", color = "black") +
  labs(title = "Distribution of Job Titles", x = "Job Title", y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

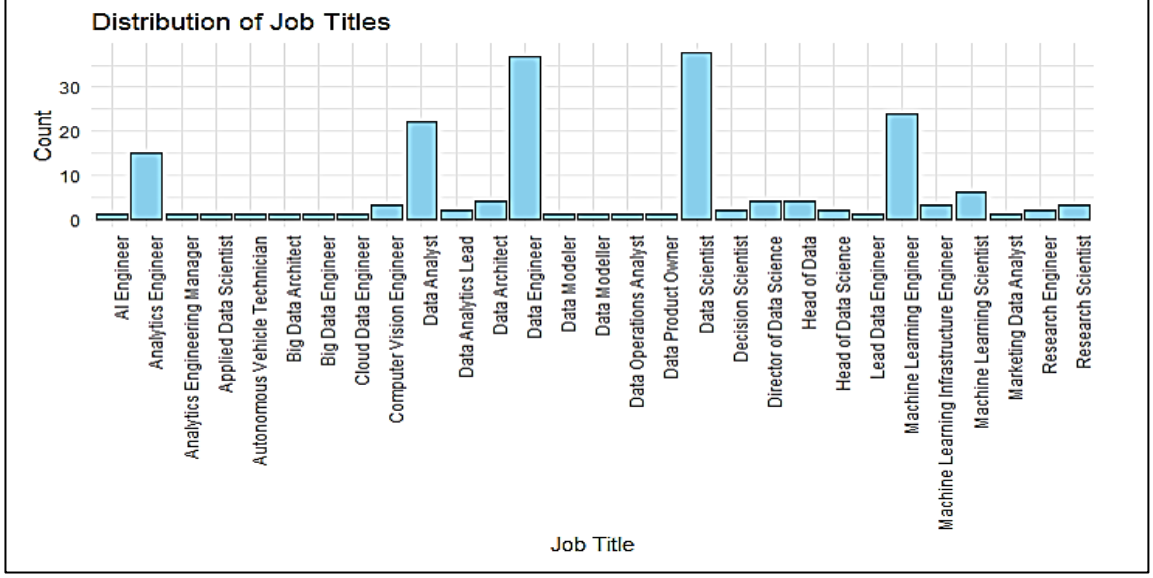

Figure 15: Different Jobs in AI profession

Next visualisation shows the areas with multiple AI professionals. United States has the highest number.

```r
library(ggplot2)

ggplot(Data_Scientists, aes(x = employee_residence)) +
  geom_bar(fill = "gray", color = "black") +
  labs(title = "Distribution of Countries", x = "Job Title", y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

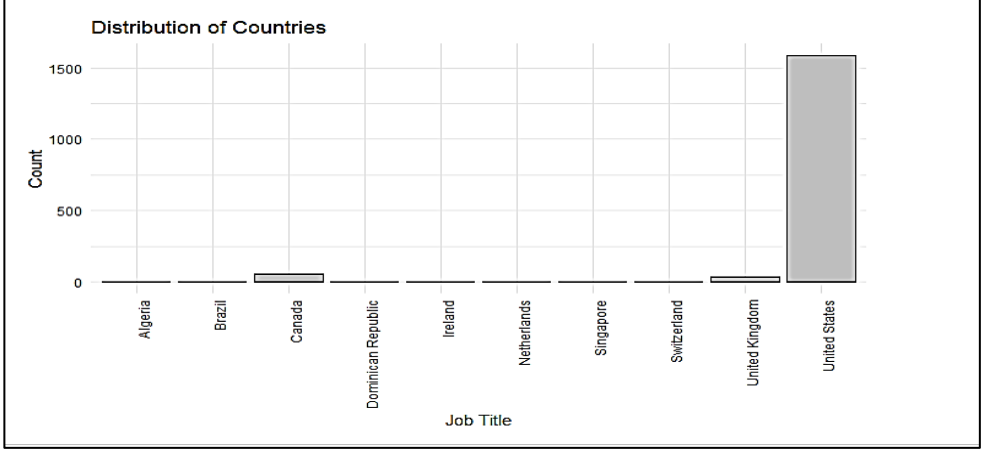

Figure 16: Countries with AI professionals

3.3. Discussion and Critical Reflection:

The implementation of MapReduce algorithm is carried out in this report. Overall, this implementation is a great learning experience that was crucial for the understanding of new concepts. Developing the research question, acquiring the dataset was relatively a smooth process without many problems. The design and implementation part were rather challenging. I experienced many issues and difficulties while dealing with the second part of the report. Though the development of research objective was comparatively easier but it was difficult to come up with the distributed solution for the problem. After a lot of pondering and exploration of dataset, it was possible to come up with an appropriate design. Similarly, the implementation phase was challenging.

In the initial design and implementation, I tried using second reducer to take input from the first reducer and create a chain of mapper, reducer 1 and reducer 2. It might be lack of knowledge on my part but I was unable to implement this design. The screenshot below shows multiple attempts to run mapper and 2 reducers concurrently but there were no results. This led me to change the design and implementation with single mapper working individually with 2 reducers. Following are the few commands I tried to run.

```
! hadoop jar /usr/local/hadoop-3.3.6/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar \
-Dmapreduce.job.reduces=2 \
-files /content/2mapper-job.py,/content/2reducer-job.py,/content/3reducer-distribution.py \
-input /content/jobs_in_data.csv \
-output /content/output/test-10 \
-mapper 'python 2mapper-job.py' \
-reducer 'python 2reducer-job.py' \
-output /content/output/test-10-secondreducer \
-mapper 'cat' \
-reducer 'python 3reducer-distribution.py'
```

```
[ ] !hadoop jar /usr/local/hadoop-3.3.6/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar \
-Dmapreduce.job.reduces=2 \
-files /content/2mapper-job.py,/content/2reducer-job.py,/content/3reducer-distribution.py \
-input /content/jobs_in_data.csv \
-output /content/output/test-13 \
-mapper 'python 2mapper-job.py' \
-reducer 'python 2reducer-job.py' \
-reducer 'python 3reducer-distribution.py'
```

```
[ ] !hadoop jar /usr/local/hadoop-3.3.6/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar \
-Dmapreduce.job.reduces=2 \
-files /content/2mapper-job.py,/content/2reducer-job.py,/content/3reducer-distribution.py \
-input /content/jobs_in_data.csv \
-output /content/output/test-14 \
-mapper 'python 2mapper-job.py' \
-reducer 'python 2reducer-job.py' \
-mapper 'python 3reducer-distribution.py' \
-reducer 'python 3reducer-distribution.py'
```

Figure 17: Failed commands for MapReduce

Google Colab is used for execution of MapReduce with Hadoop. Though the terminal implementation would have been easier for me, but due to VPN issues, I switched to Google Colab.

The evaluation of results is carried out using R code. It is always encouraged to use visualisations; the outcome of both reducers is visually represented in section 3.2. Both implementations are producing same results except the some calculated values. For this dataset, the processing speed is similar for both MapReduce and R, owing to limited dataset. The contrast would have been much clearer with a larger dataset. Another issue with my approach is the generation of two output files, each displaying different calculated fields in the MapReduce procedure. This work can be improved upon by using slightly bigger dataset. The functionality can be enhanced by employing two separate sets of mapper and reducer functions linked together to generate more intricate questions and answers.

4. Conclusion

This report contains the practical implementation of MapReduce Algorithm using Hadoop. It is an excellent platform for storage and processing of big data. For this project, an open-source dataset is utilized to address the research query concerning AI professionals in the United Kingdom, classified by their salary levels. A distributed design is created with one map function and two reducer functions. Implementation of these functions is carried out in python. The execution with Hadoop filing system (HDFS) is done in Google Colab.

The outcomes are generated corresponding to each objective, ultimately fulfilling the final research question. The evaluation is carried out by complete implementation of solution in R. The results are identical in both of the cases. The future work includes using bigger dataset, improvement in design by

implementing two sets of mapper and reducer function and including more variables to develop complex research questions.

References

Jayalath, C., Stephen, J. and Eugster, P. (2014). From the Cloud to the Atmosphere: Running MapReduce across Data Centers. IEEE Transactions on Computers, 63(1), pp.74–87. doi:https://doi.org/10.1109/tc.2013.121.

Karloff, H., Suri, S. and Sergei Vassilvitskii (2010). A model of computation for MapReduce. ACM - SAIM symposium on Discrete Algorithm, pp.938–948. doi:https://doi.org/10.5555/1873601.1873677.

Qaasim, H. (n.d.). Jobs and Salaries in Data Science. [online] www.kaggle.com. Available at: https://www.kaggle.com/datasets/hummaamqaasim/jobs-in-data/data.

Appendix

The appendix material is submitted as a zipped folder with following files

1. Jobs_in_data.csv
2. salary_mapper.py
3. Datascientist-reducer.py
4. UK-distribution-reducer.py
5. Evaluation.RMD