Investigating the 2019-2021 Kaggle survey data

Module Name: COMP2271
Date: 30/03/2022
Submitted as part of the degree of BSc Natural Sciences to the
Board of Examiners in the Department of Computer Sciences, Durham University

1 Introduction

The Kaggle survey asks participants involved in the computer science industry a series of questions relating to their work and background, while also keeping track of personal information such as country of origin and age. Some of the questions are multiple choice, while others offer multiple selection, and a few offer the choice of text input.

We are given the results from the years 2019, 2020, and 2021 (in separate CSV files), which amounts to over 60,000 responses in total. We want to investigate the data across all three years, specifically exploring the situation of women in computer science worldwide. In order to do this, we will need to merge the CSV files into one large dataset, helping us to process and plot the results more efficiently. The accompanying Python code is provided, which we will reference throughout.

2 Merging the datasets

The solution to this issue exists in the accompanying file "Problem 2.py". Note that when we merge the datasets, we will only include questions that exist in all three surveys. In each of the datasets, multiple choice questions are represented as a single column, with either the selected choice in the corresponding cell, or an empty NaN value if the participant did not answer.

Multiple selection questions however span several columns, with each column representing a different option. In each of these columns, the cell is left empty if the participant did not select that option, or contain the option in text if they did select that option.

2.1 Handling textual responses

The first problem we run into here when inspecting the datasets is that the 2019 responses also come with an additional file of text responses to some questions. This is because in the 2019 survey, respondents were able to enter a textual response for multiple selection questions if they selected the option "Other". This was not made available in the 2020 and 2021 surveys.

Considering that text responses can be very random and inconsistent in nature, and that they are in some sense irrelevant since they correspond to selecting "Other" and not one of the main options, we can essentially disregard these and consider only the multiple choice/selection responses from the 2019 data. Lines 4-6 of the code therefore load the separate CSV files into pandas data frames.

2.2 Changing the columns of the data frames

In each of the data frames, the column names don't correspond to the questions themselves, but rather the question numbers, e.g. "Q5". The actual questions exist in the first row of the data frames. In order to successfully join the data frames together, we need the column names to be exactly the same for common questions in all three data frames. Therefore, our first job is to replace the column names with the first row (the actual questions), and then drop the first row, in each data frame. Lines 8-18 handle this.

Another important thing we notice is that some of the questions are worded slightly differently across the three surveys, but essentially ask the same thing. For example, in the 2019 survey, there is a question asking "Have you ever used a TPU (tensor processing unit)?". In the 2020 and 2021 surveys however, this is worded as "Approximately how many times have you used a TPU (tensor processing unit)?". Therefore, we need to change some of the column names in each of the three data frames so that they match up with each other. This is done in lines 36-63, 89-118, and 145-174 of the code.

2.3 Handling different options across the surveys

Another issue we encounter is that for many of the multiple selection questions, the options change across the three years. Many of the options exist in all three surveys, but some only exist in one or two. In order to ensure fairness in the merged dataset, we will only include options that were given in all three surveys.

However, some options are actually merged into one in different years. For example, for the question "Which of the following integrated development environments (IDE's) do you use on a regular basis?", the 2019 survey offers the option "Visual Studio / Visual Studio Code". In the 2020 and 2021 surveys, this option is split into two separate options as "Visual Studio" and "Visual Studio Code". We therefore need to merge the answers from the two options in the 2020 and 2021 surveys into a singular option of "Visual Studio / Visual Studio Code".

To do this, we simply take the full column of each of the two options, and use string concatenation on them as shown in lines 20-27 of the code [1]. This will essentially create a new column which will have empty values if the individual did not select either of the two options, and will contain

some string if the individual selected at least one of the two options. We will see it is unimportant what the string values are when we clean the data. We have to do this for several columns across all three surveys, for example in lines 30-33 and 73-78 of the code.

2.4 Concatenating the data frames

Before we concatenate the data frames, we insert an extra column into each of the data frames, corresponding to the year in which each individual took the survey. Lines 177-179 show the implementation of this. Finally, we take the intersection of the three data frames with respect to the columns, meaning the new data frame we produce will keep only the columns (i.e. questions) which were in all three of the edited data frames, and keep all individuals from all of the edited data frames. This is shown in line 182. Once this is done, we just write the new data frame to a new CSV file. Now that we have produced a new data frame containing the answers from all three years, we need to clean it in order to actually investigate the data.

3 Cleaning the data

We clean the data in the Python file ”Problem 3.py”. The first thing we can do is filter out any individuals who took less than two minutes in answering the survey. We were told that the 2020 dataset had already done this, so it makes sense to make things fair across all three years by doing so for the merged dataset. Line 6 of the code explains how we accomplish this.

3.1 Renaming columns

A few of the column names are too long in length and can be expressed in fewer words. For example, the column ”What is your gender? - Selected Choice” can simply be renamed to ”Gender”. Likewise, ”What is your age (# years)?” can be changed to ”Age”. This will make it easier for us to access these columns when we want to visualize and investigate the data, since the column names will be more concise. Lines 9-18 rename some of the columns in the data frame.

3.2 Dealing with levels in categorical columns

When we observe the value counts in the multiple choice columns, we find several issues relating to the levels (i.e. options) of the corresponding questions.

3.2.1 Repeated levels

Similar to the issue we encountered with some of the multiple selection questions when merging the three datasets, we see that in a few of the multiple choice columns, options which are the same are worded differently across the three years. For example, when we observe the value counts for the ”Gender” column, we find that there are instances of ”Male”, ”Man”, ”Female”, and ”Woman”.

Obviously, ”Male” and ”Man” are essentially the same option, as are ”Female” and ”Woman”. Hence, we can just replace all instances of ”Man” and ”Woman” with ”Male” and ”Female” respectively in the ”Gender” column, demonstrated in lines 46-48 of the code. We observe the same issue for the ”Current role” column, in lines 29-35.

3.2.2 Poorly worded levels

In a small number of the multiple choice columns, a few of the levels are worded poorly or are too long. For example, in the ”Country” column, we have levels of ”United Kingdom of Great Britain and Northern Ireland”, ”Viet Nam”, and ”Iran, Islamic Republic of...”, which can all be replaced with ”United Kingdom”, ”Vietnam”, and ”Iran” respectively (lines 22-27). We employ a similar method to remove the word ”years” from all the levels in the ”Coding experience (years)” column (lines 88-94). Similar to renaming columns, this will make it easier for us to access individuals with these levels when we want to plot data.

3.3 Dealing with missing values in multiple choice columns

In almost all of the multiple choice columns, we find a large number (over 5,000 in many cases) of missing (NaN) values. We need to figure out some way to replace these NaN values, however there are a varying number of ways to do this for each column and it depends on the number of NaN values and if there is an obvious replacement for them.

3.3.1 When there is an obvious replacement

In most columns, there will be an obvious replacement for the NaN values, whether it be a level that already exists, or one that we can define ourselves. For example, in the column ”Approximately how many times have you used a TPU (tensor processing unit)?” there are about 8,000 NaN values, and it would clearly make the most sense to just replace these with the existing level of ”Never”. Line 103 shows how we do this.

Furthermore, in the column ”Primary tool to analyze data” there are over 16,000 NaN values (meaning approximately 25% of the participants did not answer this question). We could replace them with the existing level of ”Other”, however this does not make the most sense. Most of the individuals who did not respond to this question may not actually analyze data as part of their job. They could be software engineers or maybe students. Hence, it makes more sense to replace the NaN values with a new level of ”Do not analyze data”. Lines 82-86 explain this.

3.3.2 When the number of NaN values is large

In five of the columns, there are approximately 25,000 NaN values, which corresponds to over 35% of individuals not responding. This feels like a major problem which could be hard to deal with, however there is one underlying reason for why there are so many missing values.

All of these columns only apply to people who are employed. For example, one of the columns is ”Number of employees at your company”, which clearly cannot be answered by students. If we filter out any individuals that are not students, we see that the only response in all of these columns is a missing value, of which there are over 15,000.

So, although there is an alarming number of NaN values in these columns, about 60% of them are because of students to which the question is not applicable. Hence, we can replace all missing values in these columns with a new level called ”I do not know / not employed”. We then do a similar

procedure for the other four columns. Lines 40-44, 50-58, 60-63, 66-73, and 75-80 show the implementation on all five columns

3.4 Handling the multiple selection questions

As previously mentioned, the multiple selection questions span several columns for each question. When we merged the three datasets together, we set it up so that each multiple selection column would have a missing value if an individual did not select that option, and would have anything else if they did.

There is no real way to combine the columns for each question into a single one, however with the way we merged the datasets, it is actually fairly simple to clean the columns. We first save all the names of the multiple selection columns to a list, while simultaneously adding them to a dictionary along with a new, tidier name for each column. This is shown in lines 113-119.

Then, for all of the multiple selection columns, we simply replace all NaN values with a 0, and any non-NaN values with a 1, which we see in line 123. After doing so, we simply rename all of the multiple selection columns to their tidier version, and that ends the cleaning process of our data.

Now that we have a fully cleaned dataset, we can start investigating the data and visualizing it through the use of the Seaborn library.

4 Investigating Data Science Frameworks

In this section, we will reference the Python file "Problem 4.py".

Something interesting which we can investigate is the main frameworks (i.e. languages and visualization libraries) used by experienced (5+ years of programming) data scientists. This will give us a good insight into the best frameworks out there. We want to plot the top 5 (most users) in each category from each year. To help us do this, we filter the cleaned dataset by the relevant conditions (only data scientists and more than 5 years of programming experience), and create three new datasets from that for each year. This is demonstrated in lines 7-16 of the code.

Then, we make lists of the columns we need for each category (lines 21-26). Note that we need lists because languages and visualization libraries were offered as multiple selection questions in the surveys. Once we have done this, we call a function with each category-year pair which will plot a Seaborn barplot of the top 5 languages/libraries in each year (lines 62-70).

The function (lines 28-60) works by creating a Pandas series from a dictionary storing sorted [2] counts of the number of 1s for each relevant option (not including "None" and "Other") in the given year. It then uses Seaborn to create a barplot with the top 5 counts on the y-axis and their corresponding names on the x-axis. The following subsections interpret the results.

4.1 Programming languages

Fig. 1 and Fig. 2 show that the top three languages (Python, SQL, R) have not changed between 2019-2021, however there appears to be a recent surge in the number of Java and JavaScript users, as they have both overtaken Bash. Note also that the number of Python and SQL users has increased significantly more than any of the other languages between 2019-2021, indicating a larger uptick in popularity.

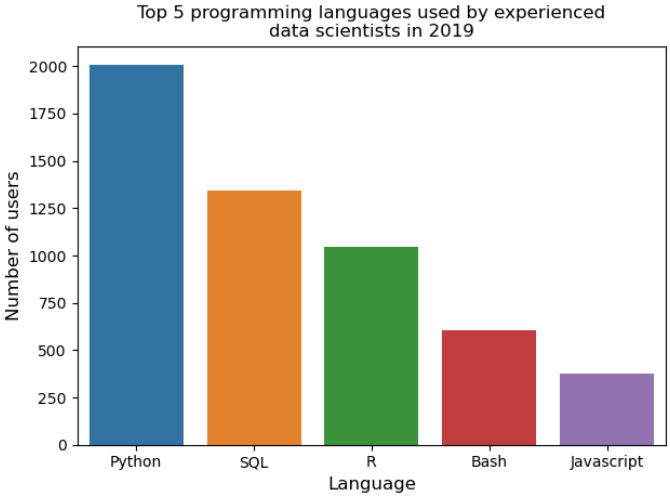

Fig. 1. Top 5 languages among experienced data scientists in 2019

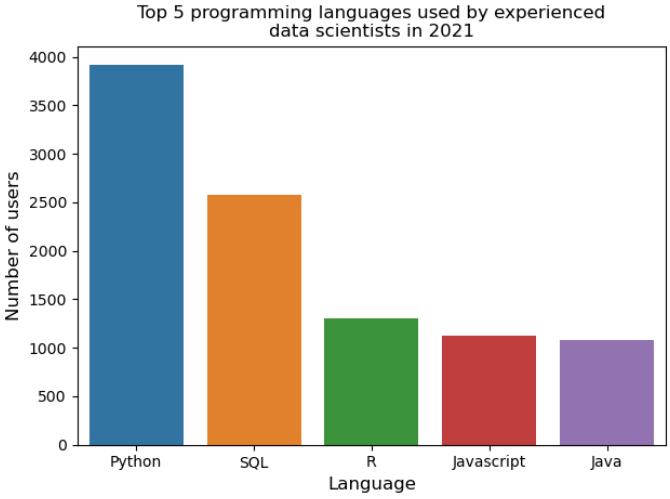

Fig. 2. Top 5 languages among experienced data scientists in 2021

4.2 Visualization libraries

Similar to the case of programming languages Fig. 3 and Fig. 4 show that the top two libraries (Matplotlib and Seaborn) have not changed over the last three years, and the number of users of each seems to have increased far more than other libraries. We also see that Plotly has become more popular recently than Ggplot, whereas Shiny has remained as the fifth-most used library and has not had a significant increase in users.

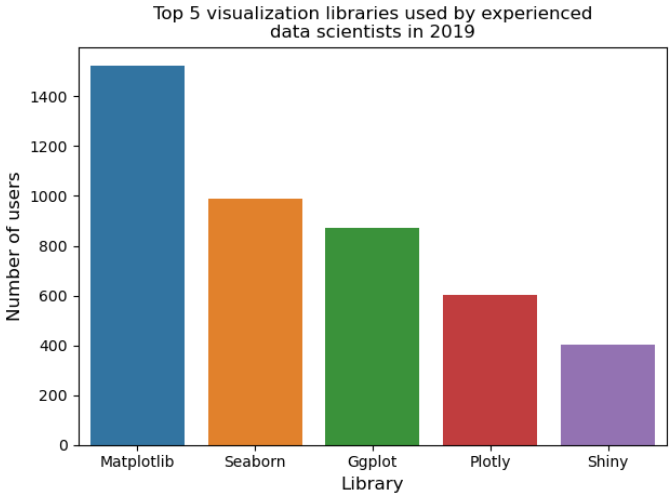

Fig. 3. Top 5 libraries among experienced data scientists in 2019

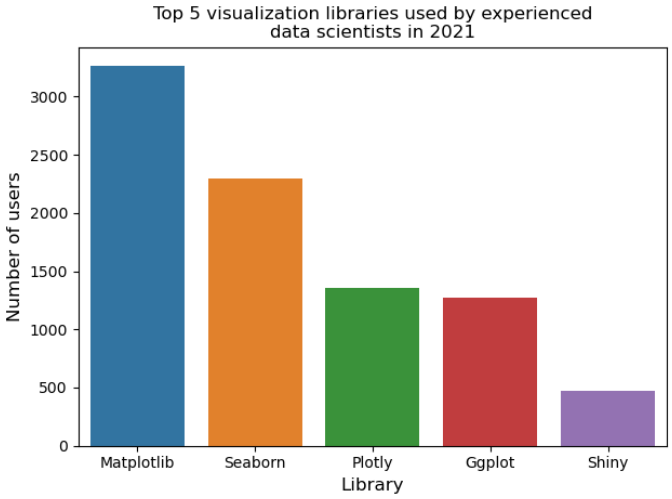

Fig. 4. Top 5 libraries among experienced data scientists in 2021

5 Investigating the world-wide situation of women in computer science

In this section, we will reference the Python file "Problem 5.py". To explore the situation of women in computer science, we will use Seaborn on the cleaned Kaggle dataset to produce a series of plots. Note that we will first filter the dataset to only include individuals whose gender was "Male" or "Female" (line 7), to make it easier to plot by gender.

5.1 Disparity in numbers

As we see in Fig. 5 (plotted using lines 9-13), there is a drastic difference in the number of men and women involved in the computer science industry. Despite a small increase for women between 2019-2021, there was an even larger increase for men between 2020-2021, potentially indicating some disparity in the opportunities offered to men and women in computer science.

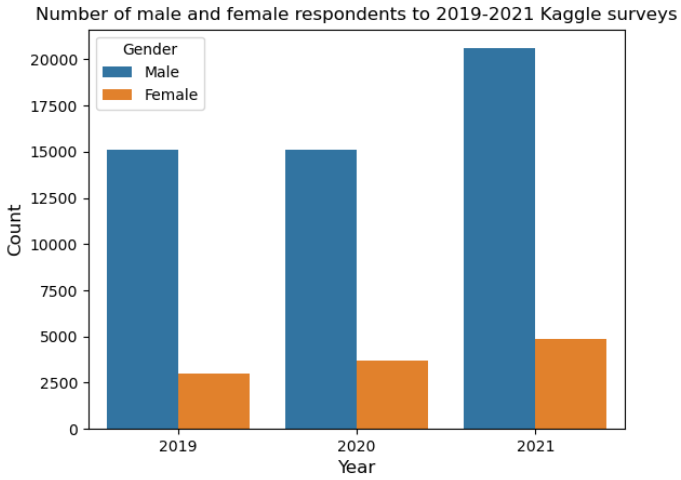

Fig. 5. Number of male and female respondents to 2019-2021 Kaggle surveys

5.2 Recent surge in women entering computer science

As Fig. 5 alludes to, it appears as though more women have entered the computer science industry. To investigate this more, we can plot the age distribution of men and women from the dataset. The age column contains categorical data of age ranges however, and not numerical data of actual ages. So we first need to transform the age column into numerical data by replacing all intervals with a value in that interval (lines 33-34). We then use a Seaborn histplot with bins corresponding to the previous intervals (lines 35-37).

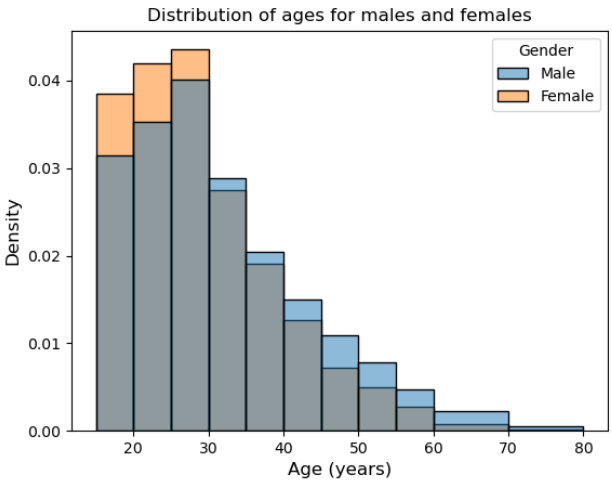

Fig. 6. Age distribution for males and females from the 2019-2021 Kaggle surveys

As we see in Fig. 6, younger people make up a much higher percentage of the female population than they do in the male population. This shows that men have been working in the computer science industry longer than women, meaning the entry of women into computer science may be a more recent trend. This idea is reflected in Fig. 7 (plotted in lines 43-54 using a similar numerical transformation), which

explains how women in general have less programming experience than men. Overall, this tells us that more and more women are finding an interest in the industry, which could lead to less of a disparity in the overall numbers of men and women.

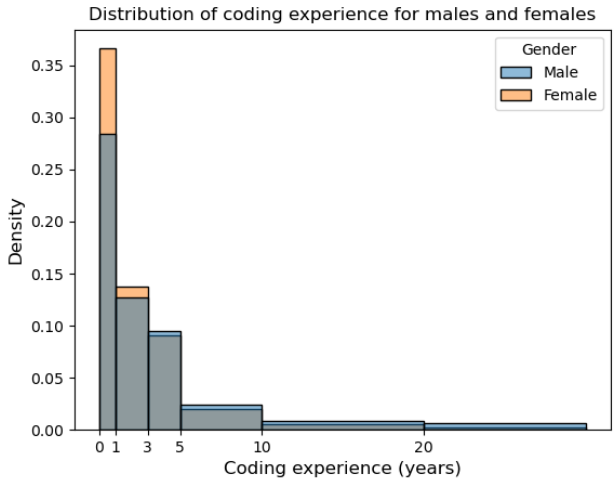

Fig. 7. Coding experience distribution for males and females from the 2019-2021 Kaggle surveys

5.3 The pay gap

The pay gap between men and women is a hot topic not just in the computer science industry, but really in any profession. We can investigate this from our dataset by plotting a histogram of the "Current yearly compensation ($USD)" column. Similar to before, we transform the answers into numerical data (we change all responses of "Not compensated" to -1 and temporarily remove them from the data) in lines 56-63. We then produce a plot showing the distribution of compensation for men and women on a logarithmic scale (lines 64-70), shown by Fig. 8. We clearly see that the distribution for men is more dense at higher values than that for women. This does not necessarily mean that there is inequality of pay between the two genders however. Much of the disparity could be due to the fact that a larger proportion of the female population in computer science is made up of younger people and people who are less experienced programmers, hence these people would, in general, be paid less.

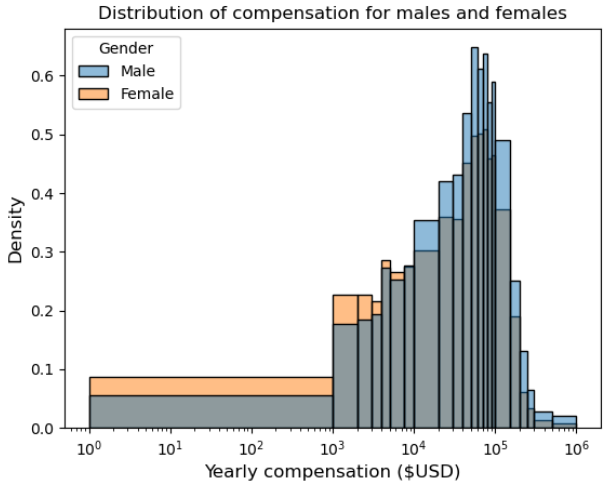

Fig. 8. Distribution of yearly compensation in $USD for men and women from the 2019-2021 Kaggle surveys

To explore deeper into the issue of the pay gap, we can plot the distribution of compensation for men and women by their "Current role", to see if they are being paid similar amounts for the same role. We filter the dataset to only include data scientists and analysts in lines 132-137. We then create a Seaborn facet grid in lines 138-139, and map separate histograms of compensation for each role by gender onto the facet grid (lines 141-144), shown in Fig. 9. We see that the distributions are more equal when we filter by job, though men appear to still be favored, which again could just be a result of the previously discussed reasons, or maybe prejudice. We cannot really tell from the data we have, however what we can say is that more women are definitely taking an interest in computer science.

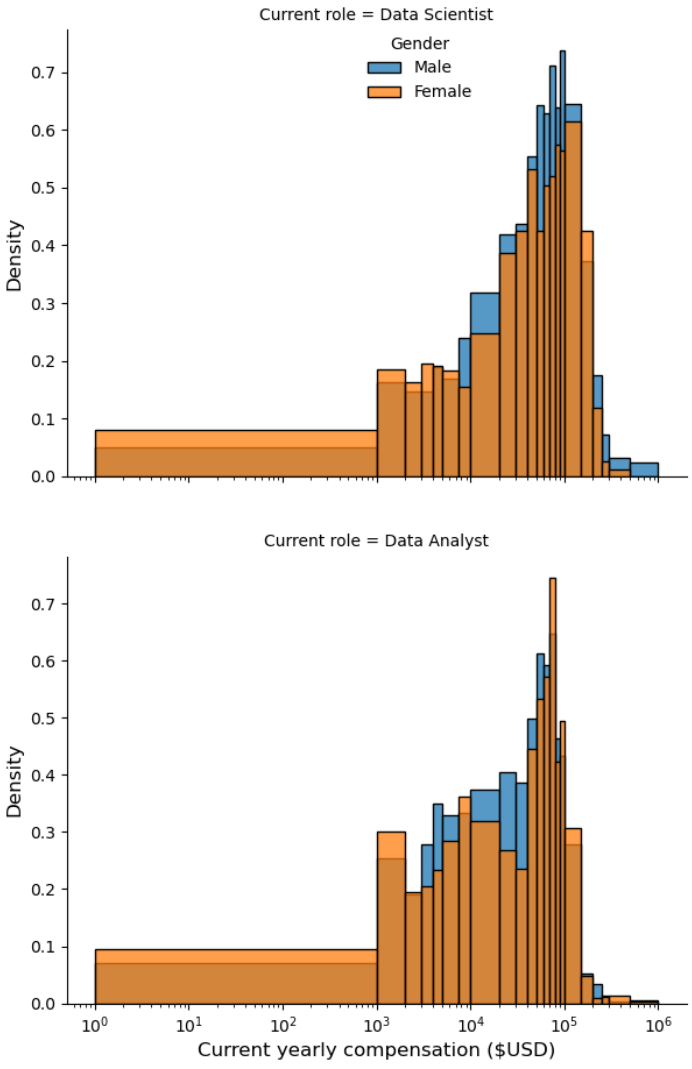

Fig. 9. Distribution of yearly compensation in $USD for male and female data scientists and data analysts from the 2019-2021 Kaggle surveys

REFERENCES

[1] https://www.statology.org/pandas-combine-two-columns/
[2] https://www.tutorialsteacher.com/articles/sort-dict-by-value-in-python