Coursework 1 pdf

1. Data preparation & understanding

1.1 Python code to prepare and understand the data

See data.py file

1.2 Original data set

See dataset.csv file

Data source: HESA. (2023). Estates management by academic year and HE provider. Online. https://www.hesa.ac.uk/data-and-analysis/estates/data.csv

License: CC-BY-40

1.3 Prepared dataset

See dataset_prepared.csv file

1.4 Explanation of code for preparation & understanding

The first action carried out to prepare the dataset was creating a pandas DataFrame. This was done using the read_csv function. Because the dataset.csv file contains some lines giving general information about the data such as the title, data source, license and more, it is necessary for the dataframe to be created without these lines. Therefore, the argument skiprows was added to the read_csv function to ignore these lines.

Once the dataframe was created, some general functions for information on the dataframe were used within the understand_df function. This includes the functions shape, columns, head, tail, dtypes. These functions allowed me to get a general understanding of the dataset including its size, the columns present and an idea of the information within each column. I could see that the colums 'Country of HE provider' and 'Region of HE provider' could present issues if there was some whitespace around the values for these columns. Therefore, I printed out the unique values within these columns to check this. The printed information showed that this was not an issue.

The dtypes function was also especially useful to start to give an idea of what issues there may be with the dataframe. The dtypes function showed that the 'Value' column was categorised as a string whereas I expected its type to be an integer based on the first and last five rows. This suggested there were null values and/or categorical values in the column which would present a problem. To confirm this I ran the isnull.sum function to see if there were null values in the data set and which columns they were in. This showed me that all the null values are in the 'Value' column.

I have inferred that these null values are present where the Higher Education Institution did not collect or provide data for that metric. Because all the values are independent from each other, they can be categorised as Missing Completely At Random (MCAR). Because my dataset is so large with over 246,000 rows and 18% of the rows have missing values, I decided to delete the rows with missing data. Deleting these rows won't have an impact on any of the other rows. This can be seen in the clean_data function. I also printed the isnull.sum function again to confirm there were no longer any remaining nulls in the dataset.

After removing the rows with null values, my dataset is still relatively large with 229,000 rows. I feel as though such a huge amount of data will be difficult to process when creating my data visualisation

dashboard. Therefore, I decided to narrow down the data I will use. I decided to remove any data from before the 2018/19 academic year and also only use data from HEIs in England. This significantly filtered by data resulting in a dataset of only 89,000 which will be much easier to use in my web app. This was done using the functions remove_years and keep_england_data.

The next step I took to continue preparing the data was to improve the 'Table' column. From the HESA website where the dataset was downloaded from, the data is split into five tables based on the 'Class' of the data as shown in Table 1 below When the dataset is downloaded, there is no column for 'Class', there is only a column for the 'Table' value. When it's time to create the dashboard, I ascertain as though it will be more useful and intuitive for me to have a 'Class' column rather than a 'Table' one. Therefore, I decided to rename the 'Table' column to 'Class' and replace all the values for Table for their corresponding 'Class'. The code to do this can be seen in the functions rename_columns and rearrange_columns. The function 'rename_functions' creates a new column called 'Class' and replaces the values in the 'Table' column with the corresponding class names and then removes the 'Table' column from the dataframe. The results in a 'Class' column that is after the 'Value' column which wasn't intuitive. Therefore the rearrange_column function was used to move the 'Class' column to the index that the 'Table' column was previously in.

Table 1 'Table' Column in raw dataset and corresponding 'Class' column in prepared dataset

Table	**Class**
Table-1	Building and spaces
Table-2	Energy
Table-3	Emissions and waste
Table-4	Transport and environment
Table-5	Finances and people

After this, I checked the dtypes of the dataset once again, but the 'Value' column was still object suggesting that removing the null values hadn't fully cleaned the data. I tried to change the data type to be float but this returned an ValueError: could not convert string to float: '86.8%'. Therefore, in the understand_df function I included some lines of code to show which rows had non-numeric values in the 'Value' column. This showed that there were a few issues.

I could see that some rows included the percentage in the 'Value' column, but this was unnecessary as the 'Category marker' lets the user know the value is a percentage. Therefore I used the str.rstrip('%') function to remove the percentage sign in the convert_value_column function.

There were a few other categories of non-numerical values in the dataframe. The first was in the category marker called 'Scope 3 carbon emissions from waste'. With this, the method of collecting scope 3 emissions from waste was rated across a scale of 'Basic', 'Medium' or 'Detailed'. Because this is a three-point scale, I converted the categorical ratings to numerical ratings to make the data easier to analyse. This can be seen in the convert_to_numeric function. Similarly, there were some 'Yes' and 'No' values related to environmental management, reporting, and Fairtrade accreditation. I also decided to change the Yes/No values to numerical values. The 'Fairtrade accreditation existence' category has Yes/No values but also 'Working towards accreditation'. Therefore, I added another numerical value that corresponds to this. Again, this can be seen in the convert_to_numeric function. The numerical values used for the ratings can be seen in Table 2.

Table 2 Categorical rating values and corresponding numerical values

Categorical value	**Numerical value**

Scope 3 carbon emissions	
Basic	1
Medium	2
Detailed	3
Environmental management and reporting & Fairtrade accreditation existence	
No	0
Working towards accreditation	0.5
Yes	1

The final type of categorical data in the 'Value' column is related to the type of external verification used to evaluate the environmental management system. The values for this category marker include 'ISO14001', 'Eco Campus Platinum', 'Eco Campus Silver' and more. Because this is not a rating system, it is not appropriate to convert this to equivalent numerical values. Therefore, I made the decision to keep these values as they are.

The next thing I then decided to do was to explore the data. Because the values for each category are going to be vastly different as each category is a different measurement, I decided to first group the dataset by category. The first plot I did was a boxplot to see the distribution of the data and identify any outliers. The resulting figure (Figure 1) shows my initial plot where each subplot is a different category. The figure shows that the box plot weren't completely useful as it suggested there was a great amount of outliers.

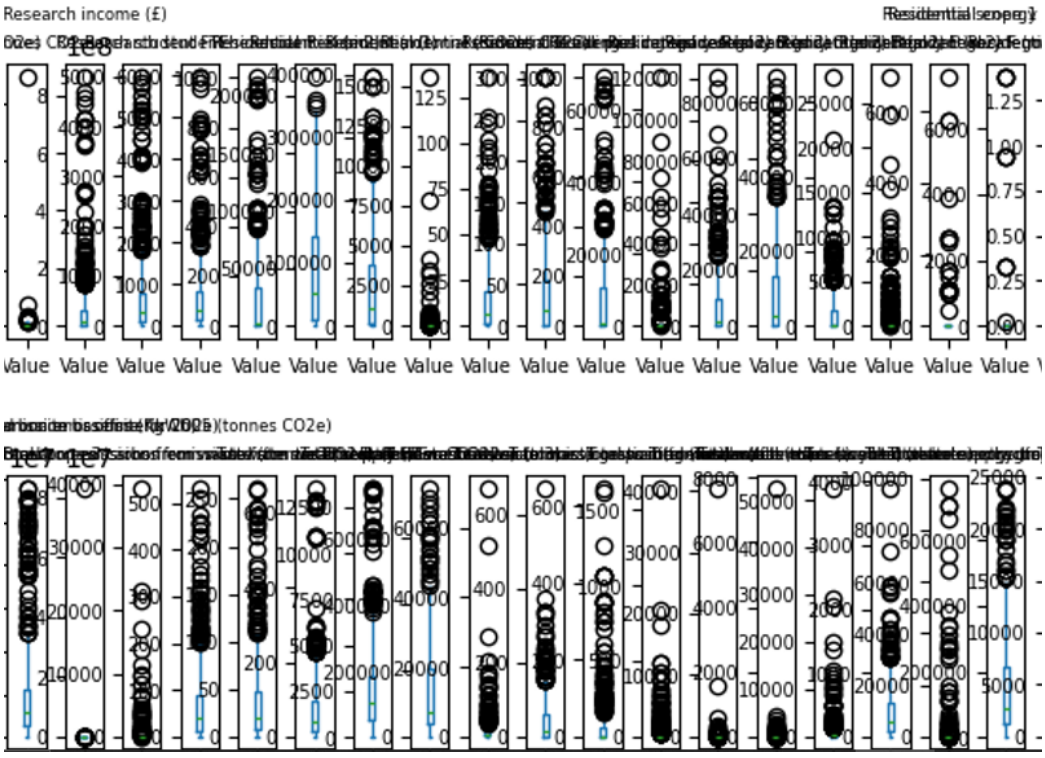

Figure 1 Selection of boxplots initially obtained for various categories within the dataset

I decided to explore other types of way to visualise the data distribution as I didn't think the boxplot gave me enough information visually as there are so many values overlapping. I came across seaborn stripplots which draws a categorical scatterplot with jitter to avoid overlapping data. With this I was able to view the data a lot better and also modified my code so that multiple figures were created and so that each subplot can be seen clearly. These figures can be seen in the folder called figures.

A cropped version of one of the figures can be seen in Figure 2 below. The first plot on the left shows the distribution of values for Liquefied petroleum gas (kWh) values. This shows that the majority of values are around 0 kWh. There are some data points that are higher between 0-200,000 kWh.

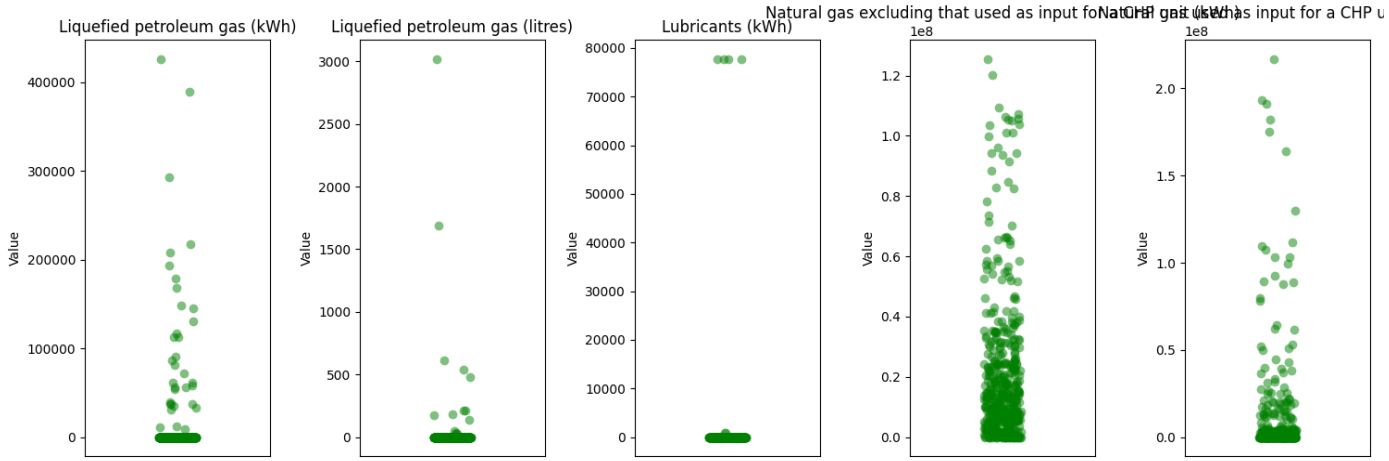

Figure 2 Stripplots obtained for 5 categories showing the distribution of values of within that category. Each dot represents a data point within that category

However there are 3 key outliers above 250,000 kWh. On the other hand, the fourth subplot from the left is a lot more distributed with a large number of points varying between 0-0.4 kWh with less points ranging from 0.4-1 kWh. Finally, the lubricants subplot show that these data points show very little distribution all around 0 kWh and with about 5v outliers at 80,000 kWh.

Because there is so much variation in the variation across categories of data, it will be very difficult to determine and remove which data are outliers. Additionally, because HEIs in England are so inevitably going to be so different in their environmental output, I do not think it will be appropriate to deem any value points as outliers. Because of these reasons, I have decided that I will not be removing any data points after visualising the data with stripplots.

In summary, I have cleaned and prepared my dataset for use in the creation of my webapps. I have filtered the data by removing the data from the first three years and only keeping the data of HEIs based in England. I removed the null values and removed percentage signs in the Value column. I also converted some of the categorical rating into numerical ratings to make the data easier to analyse. Finally, after visualising my data with stripplots I decided to not deem any values as outliers and have not removed any of them.

2. Product definition

2.1 Product overview

For staff and students of Higher Education Institutions (HEIs) as well as researchers who want to compare the environmental impact of HEIs in England. The HEI Environmental Information Dashboard is a data visualisation dashboard that allows user to easily compare and rank a wide range of environmental metrics. The Dashboard uses a REST API that will allow others to use the data in their software. Unlike impact ranking tables which have a methodology to determine environmental impact of HEIs, our product will allow easy visualisation of raw HEI data from the 2018/19 academic year and allows for exploration of correlations and patterns.

2.2. Persona

Reece Liplin

age: 38
residence: Didsbury, Manchester
education: Geology PhD
occupation: Professor at Manchester Metropolitan Univeristy
marital status: Married, with 2 children

I need accessible HEI environmental impact data for my current research project on whether education is more environmentally friendly that other industries

Reece spends some parts of her day lecturing Environmental Science MSc students and other parts conducting research into climate impact across various areas

Comfort With Technology

INTERNET

SOFTWARE

MOBILE APPS

SOCIAL NETWORK

Criteria For Success:

Data is where a lot of my research into climate impact starts. Without the data, I cannot make any concrete conclusions. I need to be able to access data and use it in my research easily. I want to be able to identify trends and correlations. I don't want to spend a lot of time trying to make the data I need to use manageable.

Needs

- Have environmental data for individual HEIs
- Be able to easily produce graphs for the environmental data
- Be able to filter the data based on a variety of factors
- Be able to apply multiple filters at once

Wants

- Simple interface and easy to use app
- Download graphs produced and change colour scheme
- Apply statistical analysis on data within app

Values

- Scientific integrity
- Reproducible actions

Fears

- Loss of data and work
- Overcomplicated interface
- Inefficient web app resulting in time consuming analysis

MOBILITY LABS

3. Tools & Techniques

3.1 Source code control

Github repository: https://github.com/ucl-comp0035/comp0035-cwi-4jjnaomi

3.2 Use of AI

I used both GitHub Copilot and ChatGPT in my data preparation code.

I mainly used ChatGPT when it came to creating plots for explore the data. The type of data plotting I wanted to do went beyond the material covered in this course, for example using seaborn stripplots. Also, because I had so many subplots, I needed to use matplotlib extensively to create multiple different figures with subplots. Therefore, I used ChatGPT to develop the code used the in the functions create_data_visualisation, separate_data_by_class and create_sub_plots. I explained to ChatGPT. To do this, I initially gave ChatGPT the following prompt:

> *"I have a column named 'Class' and I have 5 values in this column: 'Building and spaces', 'Energy', 'Emissions and waste', 'Transport and environment', 'Finances and people'. Each class has a variety of values in the 'Category' column. First, I would like to separate the datasets so that only rows with the same 'Class' are together. Then for each 'Class', I would like to group the data by 'Category' and make a figure where there are subplots for each 'Category' with a boxplot showing the 'Value' distribution. What is the code for this?"*

Where the resulting code had aspects I didn't understand, I asked ChatGPT to explain what the various arguments meant. At times, the code ChatGPT gave me resulted in an error. When this happened, I copied the error into ChatGPT and asked it to redo the code to avoid the error.

I gave ChatGPT various follow up prompts to get the result I wanted. Some examples of some follow up prompts I gave were:

> *"This code gets me a very large figure with 1 column of subplots. I want there to be a separate figure for each class and for each figure to have 4 columns."*
>
> *"Change the code so that instead of showing each figure, each figure is saved with an appropriate name."*
>
> *"Change the code so there is a separate subplot for each category."*
>
> *"Redo the code so that there is a maximum of 16 subplots per figure. Where a 'Class' has more than 16 'Category' values, make multiple figures."*

The other main way I used ChatGPT was to make my split my code into smaller functions as I was working with three massive functions for the majority of the time I was working on this coursework. To do this I copied my code into ChatGPT and gave the following prompt:

> *"I have the following code but my functions are too long. Help me split my functions into smaller and more manageable ones."*

ChatGPT did a great job of splitting my functions, but I had to make some changes once I'd copied the code it gave me into my IDE. This is because it changed some of the core code within the functions, therefore my code didn't work the way I wanted it to. I also had to change some of the name of the functions as they didn't align with what the code within the function was doing.

The code that ChatGPT gave me did not have any comments na the functions did not have any docstrings. Therefore, I used GitHub Copilot to add docstrings to all the functions and add comments

to help a third party understand what the code is doing. Like ChatGPT, Copilot attempted to change my code where I didn't want it to do so. I didn't accept these changes. I only accepted the docstrings and comments as these were of very high quality and much better than anything I could have come up with myself.