Report

Data Preparation

Jack Sydenham - s3841816

To begin data preparation, I addressed the obvious issues in the data, which can cause immediate harm to the exploration/modelling phases, then moved to cleaning specific issues throughout the data, to ensure my data was properly ready for analysis.

Missing Values

Looking at the combined dataset, I immediately spotted a few missing values. After a quick search in the created csv file, I discovered that luckily, there are only five rows with missing values, so I can safely replace each NaN value with their correct values, based off the other data I have. Otherwise of course, they will be removed.

- First off, I replaced the NaN values of the Ford Focus with data based off other Ford Focus's in the dataset, so, 'Ford, Focus, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN' became , 'Ford, Focus, 30.61932203, 5.966101695, 94.03389831, 1497.169492, petrol, NaN, NaN, NaN, NaN'.

- I then repeated this same process for replacing the missing values (aside from the last four columns) in row 34 (Seat Ibiza) and row 56 (Toyota Aygo).

- Row 67 read only as 'Ford, , , , , , , , , ,'. Of course, this is not enough data to provide any value, so the row was removed.

- Row 77 was missing four values including manufacturer, and while we could assume that this is Peugeot based off other data, and fill in the blanks like before, it is far more likely that this is just a duplicate row of row 76 (the one before), as all given data, even the number of male owners, is the exact same as in row 76. For this reason, this row was also removed.

- Finally, to fill in the missing values in the last four rows, as they differ between inputs, the NaN values in the Male, Female, and Unknown owners' columns were replaced with the medians of those columns respectively. The Total was then calculated based off those values. To do this, the current owners columns was converted from type 'object' to type 'float64' so that the median operation could be performed on them.

Extra Whitespaces

Searching for spaces (' ') in the created csv file produces a few results, some values in the manufacturer and model columns on lines 216, 223, 226, 228, 229, 232, and 240 had extra spaces on either side of them. Since there are not supposed to be any spaces in either the Manufacturer or Model columns, I could safely convert the two columns from type 'object' to type 'string', then apply the str.strip() function, removing all whitespaces in the selected column.

did this for both Manufacturer and Model columns and the result is:

- On row 216, the manufacturer column reads ' MG '. After applying the strip() function, it becomes 'MG'.
- Row 223 has a similar result, with the whitespaces in the Manufacturer column being removed.

- On row 226, ‘A8 ‘ in the Model column becomes ‘A8’.
- On row 228, ‘ Citigo’ in the Model column becomes ‘Citigo’.
- On row 229, ‘Liana ‘ in the Model column becomes ‘Liana’.
- On row 232, ‘ Nissan’ in the Manufacturer column becomes ‘Nissan’.
- On row 240, ‘Toyota ‘ in the Manufacturer column becomes ‘Toyota’.

The only whitespace that remained in the csv file was the one in the header that reads ‘Engine CC’. Of course, this is intentional, so all whitespaces have been removed.

Typos

Viewing .csv files in Jupyter allows for regex searches within the file. Using this, I searched for all expected Manufacturers given in the ‘data_description’ file. This way the search would highlight all correct inputs (including capitalisation) and I could simply look for the gaps that need fixing.

Upon performing a regex search on the file and scrolling down, looking for any gaps, there were no typos found within the Manufacturer column.

The same was then repeated for the Model column, searching for each of the Models, along with their associated Manufacturer. Once again there were no typos detected.

Finally, we have the Fuel column that may contain errors. Since there are only three possible values within this column, I was able to locate the errors by using the .unique() function to view all unique values in the Fuel column as follows.

- finalFrame['Fuel'].unique() returns:

```
<StringArray>
['petrol', 'diesel', 'peatrol', 'automatic', 'diasel', 'autometic
']
Length: 6, dtype: string
```

- Now it is clear that there are three typos that appear in the column, I am able to handle them individually using the replace() function.
- With these errors addressed, the data frame is void of typos, and all string values can be reliably operated on.

Incorrect Values (Sanity Checks and Outliers)

For this section, I will address the errors column by column.

Starting with the **Price column**, I knew that price values must be > 0 and ≤ 650.

- This means I was able to cycle through the column and set each value that is not within this range to NaN. (numpy must be imported for this step, so that I can assign values to be NaN).
- Then, I could set the value of all the NaN values in the column to the mean of the column. Mean is effective to use here since all outliers had been removed.
- Finally, I set the amount of decimal places in the column to 3, to match the price in dollars.

Then, looking at the **Transmission column**, I knew that the transmission values must be > 0 and ≤ 10.

- Just like last time, I removed the values outside of the specified range, replacing them with NaN values.
- I then filled in the NaN values with the Mean of the column.
- I am led to believe that the transmission value is equal to the number of gears the transmission has. For example, a BMW 535i has an 8-speed transmission, which is listed correctly in the data

frame, other transmission values in the data frame contain decimals, though are correct if rounded to the nearest whole number. Knowing this, I rounded all values in this column to their nearest whole number, removing all decimals and giving me an accurate value for each row.

Moving on to the **Power column**, I knew that the values here must be > 0 and ≤ 500.

- Immediately I could see that some of these values were appearing as scientific notation in the merged data frame (ex. 6.857143e+01 for Ford Fiesta in row 2). This is because the values had such long decimal points. I adjusted the amount of decimal places used to 2 first this time, fixing many of the errors in the column immediately (Ford Fiesta in row 2 adjusted to have Power value of 68.57).
- Secondly, after a quick filter of the column to display all negative values, I saw that there were only two negative values which needed fixing. Comparing these negative values with other data in the column, I found that the positive alternatives to these values are the correct values, so I replaced these negative values with their correct positive alternatives.
- Of course, there were still many outliers within the column, so similarly to last time, I replaced all values outside of the specified range with NaN values, then replaced all NaN values with the column Mean.
- Since performing this Mean function sets the amount of decimal places back to 6, I simply ran the function I used at the start of this section again, to set the decimal count back to 2.
- There was one last outlier in this column. On row 41, Citroen C5 has a Power value of 0.12. Of course, this was incorrect, as a car would never have less that 1BHP. Luckily, this car appears multiple times elsewhere in the data frame, with a Power value of 104.23. I simply replaced the Power value on row 41 with the correct value of 104.23, and all incorrect values had been taken care of in the Power column.

Finally, looking at the **Engine CC column**, I knew that the values here must be > 0 and ≤ 6,500.

- Unlike previously, there were no values outside of the specified range in this column.
- So, the only thing that needed adjusting was the amount of decimal places. Since engine CC is based off displacement, which is measured in millimetres, I set the decimal place count here to 2 to match the measurement of millimetres.

Data Exploration

1. Composition of Total Number of Owners

Since we are analysing a form of static composition (values measured have no time relevance to us), we can visualise the data using a pie chart.

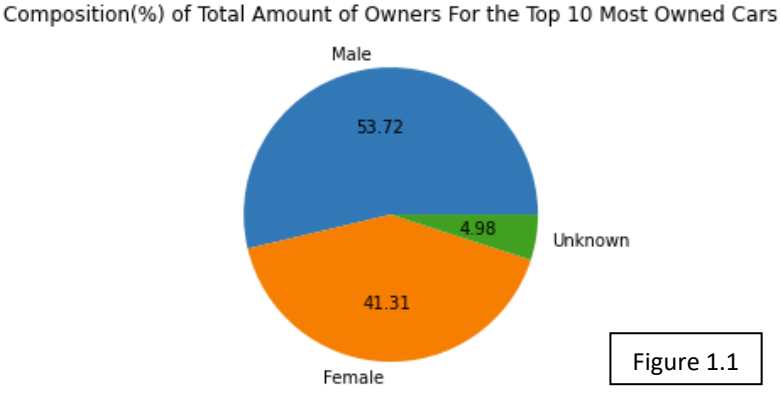

Figure 1.1

The data in this pie chart (Figure 1.1) tells us clearly that out of the top 10 most owned cars, male owners make up for than half of all owners, at 53.72% of total owners. Female owners make up 41.31% of all owners, and unknown owners make up only a small percentage at 4.98%.

2. *Error Visualization*

Spotting errors is most clear in the form of spotting outliers, so using a box plot to display the data in the Price and Power columns makes potential errors become much more apparent.

Figure 2.1

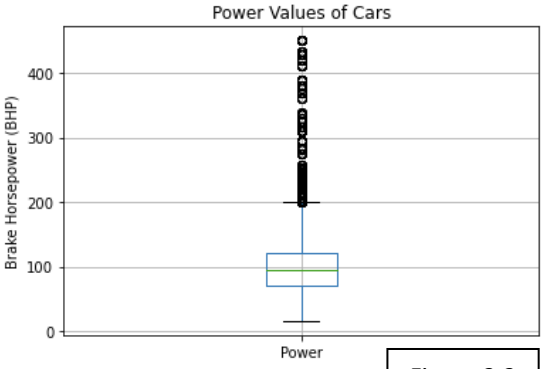

Figure 2.2

During the data preparation phase, the errors found within the Price and Power columns were taken care of. However, looking at the **Price** Box plot (Figure 2.1), it would seem there is still a dramatic outlier sitting above 600. Upon searching through the price column with a simple loc [finalFrame ['Price'] > 600], I gather that while this is a large outlier, it is actually not an error. The most expensive car in the data frame; the Porsche Carrera has a Price column value of 646.605, which appears correctly in the data frame 20 times. and since this price is under the specified value limit of 650, it is a valid entry.

For the **Power** box plot (Figure 2.2), we see a steady increase towards the specified value limit of 500, with no outliers or strange values. Much like the Price column, all errors were successfully addressed during the data preparation phase, meaning all entries here are valid.

It is also good to note that there are no negative values in either plot, as they were also addressed during the preparation phase.

3. *Relationship Between Male Owners and Other Features*

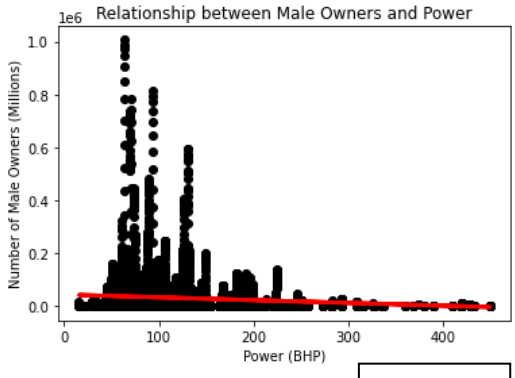

Figure 3.1

Looking at the linear regression plot above (Figure 3.1), we can see a positively skewed distribution, with a large portion of male owners opting for cars with around 70-80 brake horsepower. This high number of owners decreases as power increases, so we can say that the Male column in the data frame has a moderately negative correlation to the Power column.

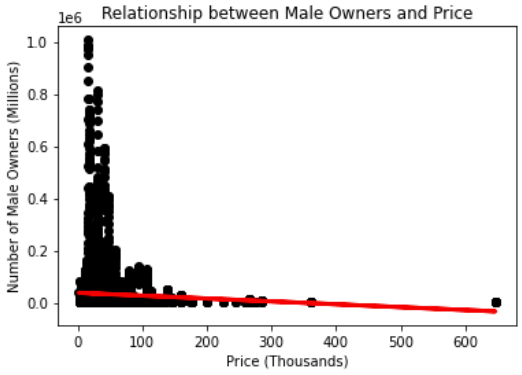

Figure 3.2

Considering the linear regression plot above (Figure 3.2), we can again see a positively skewed distribution. The vast majority of male owners here opt for cars valued at around the $20,000 - $40,000 price point, with quickly dwindling numbers of male owners opting for higher price points. There are still some male owners opting for incredibly high-priced cars, so we can say that the Male and Price columns in the data frame have a strongly negative correlation with outliers.

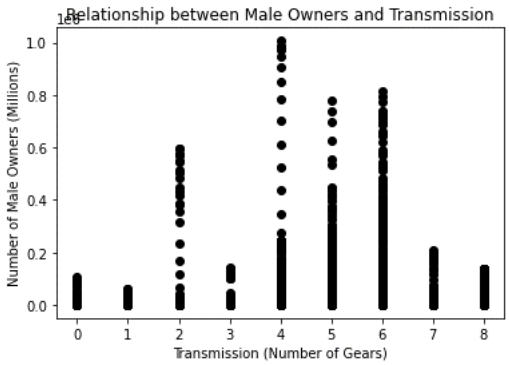

Figure 3.3

Considering the scatterplot above (Figure 3.3), we can see that this time there is a negatively skewed distribution, with the mean of male owners opting for cars with 4- 6 gears. There is a few transmission values that get a lot less attention from male owners than others, however this may be due to less cars offering these options as a whole. With these factors in mind, we can say that there is likely no correlation between the Transmission and Male columns in the data frame.

References

Python/pandas

- Stack Overflow
- Statology
- pandas documentation
- Pandas Tutorial - w3resource
- Grepper

Graphing/Visualization

- Pandas Tutorial - w3resource
- How to Choose Charts to Show Data | WebDataRocks

Subject Matter

- Horsepower - Wikipedia
- Various Google searches (ex. bmw 535i transmission)

General Research

- Canvas Assignment 1 discussion board
- Course Materials/Practicals
- Lecture/Lectorial Recordings