Task 1c

Task1a

First get the cross product for the data values in google and amazon. Write a function that tokenises string, converting it to lower case, removing stop words and punctuations as well as stemming in order to clean the data as a prepossessing for data matching. By observe the two csv files, I find out that name could be the main feature fir comparison while the description and price could also help a bit. Since the manufacturer information is largely missing hence I do not take this feature into comparison.
Use several algorithms from textdistance library to compare the matched names from the truth csv, I find out that the jaro_winkler and overlap algorithm will give a relatively high similarity for the matched strings. Also, when I test the product name for the same two pairs of matched product from the truth csv, sometimes jaro_winkler could get a higher similarity for one pair than the other pair, while the overlap algo will give me a reverse answer. Hence I apply both jaro_winkler and overlap algo to get the similarity for the product name to balance the inconsistency and use the overlap algo for product description as a tonken-based algorithm is more time efficient for long string. I also simply calculate percentage difference for the price distance. I then use linear regression model to determine the approximate importance of each feature. I create a dataframe containing every pair of product with each of their feature's similarity recorded. The final similarity score is calculated by adding the feature's similarity together with each of their coefficient varies according to the importance of each feature that I calculated before. By trial and error, I find out that 0.67 is a optimal value to be set as my threshold. I classify a product pair as a match if their similarity score passes my threshold and finally it will give a relatively high precision and recall of 0.813 and 0.769 respectively.

Overall, the performance of your product comparison is acceptable but not the best solution since there are still several 30 FNs and 23 FPs matches exist with 0.813 precision and 0.769 recall. Some method to improve my comparison is to try with other algorithms that evaluates similarities while we could also try some combinations of these algorithms to see which of them will give me a higher precision and recall.

Task1b

Look at the features of the products, matched products usually do not have the exact same name hence hard to match them in blocks, manufacturer information are largely missing where the description for matched products are also really different. Hence, price is the most suitable feature to implement blocking since it be easily allocate data into blocks by separating them according to their price range. After convert all price in GBP to AUD, I find out that the price of these products is obviously left shewed with its data largely distributed between 0-500 and gets fewer and fewer once the price gets larger hence I create a range list with the range cut more frequently for small price and less frequent for large price. Use the price range to create 32 block keys and then use the cut method to replace the price of products of google and amazon by the corresponding price range indicated by different block keys. Make a dictionary that has keys as block keys and values as another dictionary for storing lists of ids of amazon and google. Append the id from amazon and google into the dictionary for each its block key respectively, assign the id to nan blocks if the price is nan. Convert the dictionary of products in amazon and google with their block keys into datafames and save as csv files.

My blocking method It gets pair completeness 0.638 and reduction ratio of 0.860. It is time efficient in a sense that I made the blocks with different range length so there will not be some blocks that have way more data compare to the others, which reduces the numbers of comparison within the blocks. However, there are many FP and FN exist. I find out that there are 200 products in the amazon file that have a price of zero. Therefore, when we create blocks, many of these products will not go to the blocks that has its matched pairs hence many FN occurs. One way to improve my result is that I can make each block smaller while using another pass over the dataset with the product name as a block key, especially for these products that do not have a valid price so that they have more chance to be in the same blocks as their matched pairs. Although this might take a bit more time since I have another pass over the dataset, but it is still linear while it could reduce the number of FN and FP.