Task 1C

In task 1a, the naive data linkage without blocking is performed. First of all, import the csv files and match each product in both amazon and google once for later comparison. That is, produce a data frame data\_small with M x N rows where M is the length of amazon and N is the length of google. Before linking the data sets, the data need to be cleaned and the nltk library is imported. The null value in the name and description is replaced by an empty string. Loop through each row of the data frame, for the product names and descriptions, first use the function to get\_vectors normalise the comparable texts by removing stop words, punctuation and lemmatize it. Then transform it into vector form from the tokenised texts. After we got the vector, put it into another function get\_cosine\_sim to calculate the cosine similarity between the texts. Do the above step for both amazon and Google’s name, description and manufacturer to get their similarities for each row of the data frame. In addition, to calculate the price similarity, I choose to simply divide the smaller price by the larger price. After, we need to come up with a final score of the similarity between each product as each attribute contribute different weight in determining the final score. According to the truth match table, A label column is added to each pair of products to see whether it was a match. Then I fitted the label and the attributes into the decision tree classifier to measure the importance of each attribute in decide whether the products are matched. I chose to discard the manufacturer similarity as it is not so significant and has a negligible effect in determining the match, it might because of the largely missing manufacturer value in the google file. Then I had decided the coefficient of the attribute’s similarity and compute the final similarity score. For the threshold score, an accuracy score vs threshold graph was plotted to visualise for which score can produce both a relatively high recall and precision. Then, for the rows in the data frame, if the final similarity score reached the threshold, I will record the amazon id and google link to a new data frame, which is the final output.

The output contains 107 TP, 28 FP and 23 FN with precision 79.26% and recall 82.31% which is kind of balance. This output is acceptable but not so great. We should aim to make lower FP and FN. Some possible ways to improve this result could be compare and choosing better similarity function such as hamming distance, Levenshtein and so on to see which one is more appropriate in this case. Also, we can find the combination of the comparison algorithms that returns a stronger outcome.

In task 1b, I chose price as the key for blocking, the reason is that manufacturer has too much missing value so obviously it is not a really good key, description also has a portion of missing value with diverse texts even the product is actually the same, name is a reasonable key but the names it’s also not so identical, it is kind of hard to find the exact match. In summary, price would be a relatively easy and appropriate key for blocking. To start with, convert the gdp to aud in the price to ensure data consistency. As the price in both amazon and google are not evenly distributed, with data largely clustered at 0-100. I attempted to make the blocks with more equal frequency, so I make the block interval 15 in 0-100, interval 100 in 100-1000, interval 10000 in 10000-100000, and a single block with data between 100000 to 1000000 as the maximum value in amazon is 101515.55, where google is 443164. Each interval is my block key. After that, I used cut function to assign the price values to particular intervals and replace the actual price. Because the block implementation time must be linear, so I choose to create dictionary to assign the product id to each block, the dictionary key would be each price interval. If the product is in the price interval, the make the price interval as the block key, and record the product id as the dictionary value. After assigning each product id into their corresponding block key, turn the dictionary to a data frame then return the output with block key and product id.

By this blocking method, it gives the result which pair completeness is 0.625 and reduction ratio is 0.892. The pair completeness is significantly lower than the reduction ratio, it is because we got a bit high portion of product in false negative, that means there are many product that should be a match was not assigned to the same block. For improvement, as there are many prices in google appear as 0 which is obviously a problem, it can cause some errors when assigning the products to blocks. Maybe we can assign those products to other blocks using the name as block key to solve the issue. In addition, we can also reduce the number of blocks to achieve a higher accuracy, but we should be careful of that this method would lower the time efficiency when it comes to actual matching process.