Task A. The **sentiment scores** worksheet in the data file “Assignment 3 Sentiment Scores.csv” (on Canvas) provides sentiment scores (+5 to -5) of forum users on 10 car models. Each row represents a post (not shown) that can mention multiple models. Only positive and negative sentiments are noted.

From these sentiment scores, create a directed product comparison network (and use NodeXL or, even better, write your own code in Python using networkx or R). Use the principles laid out in the article “Product comparison networks” to answer this question.

Using the data provided we created the product network for the specific cars. Figure 1 below shows our product network created in python using Networkx. We created a digraph to show the sentiment for each product in both directions. The thicker black pieces of the line represent the arrows for the sentiment. For example, you can see that people who mentioned both the BMW 7 Series and the Audi A6 always preferred the 7 Series as opposed to the A6 because there is no arrow from the 7 Series to the A6.

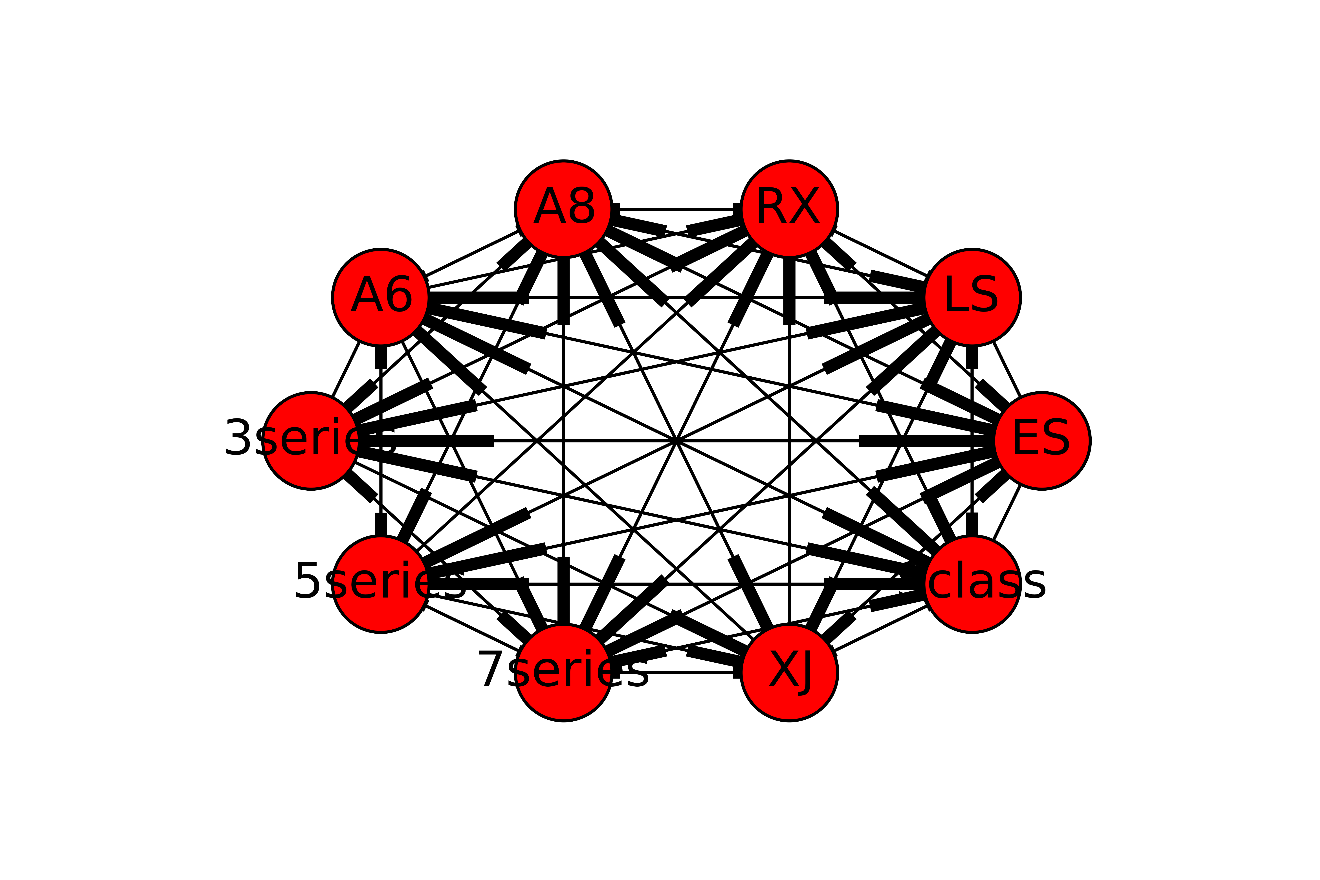


Figure 1: Directed Product Network

Task B. Calculate both unweighted and weighted PageRank scores for each car. Note that NodeXL can’t calculate weighted PageRank scores. What are the correlations between these metrics and sales figures shown below? What additional information do weighted PageRanks capture? Use a python script to calculate weighted PageRanks. Unweighted PageRanks can be calculated in NodeXL, or you can write a python script for that task as well.

|  |  |
| --- | --- |
| Model | Approximate # sold in the U.S.A. (2012+2013) |
| Audi A6 | 20k |
| Audi A8 | 12k |
| BMW 3-series | 220k |
| BMW 5-series | 60k |
| BMW 7-series | 14k |
| Jaguar XJ | 6.6k |
| Lexus ES | 135k |
| Lexus LS | 30k |
| Lexus RX | 120k |
| Mercedes S-class | 25k |

Task B took us some time to complete because we initially used the networkx PageRank function, only to find that our correlations were coming out terrible for the last part of this homework (part C), specifically for the weighted page rank. To offset this we took it upon ourselves to write our own page rank calculation using linear algebra to find the unit eigenvectors as was mentioned in class. Figure 2 shows our method for calculating the eigenvectors. We would perform iterations of the matrix multiplication until we reached convergence, setting a maximum of 30 iterations (many times we would reach convergence in the 20s with no tolerance adjustment). After doing this we found a paper about weighted page rank written by Dr. Wenpu Xing and Dr. Ali Ghorbani [http://people.cis.ksu.edu/~halmohri/files/weightedPageRank.pdf]. In this paper a formula was provided that took into consideration not only the inputs for each node referencing a page, but also the outputs of those nodes. Figure 4, Figure 5, & Figure 6 show the equations that we included in our code to calculate the weighted PageRank. The reason we did this seemingly extra work was to #1 improve our familiarity with PageRank and #2 create a more robust PageRank algorithm.

Figure 3 shows both our weighted and unweighted PageRank Spearman correlations to # of cars for the data from part A. For unweighted we were getting nearly no correlation (0.006). However, our work put into building the algorithm for a weighted PageRank allowed us to achieve a correlation of 0.58. Clearly weighted PageRank is able to tell us more about # of cars sold. This makes sense because weighted page rank does not just count the existence of a product comparison; rather, it includes the importance of each node that is referring to that product (as mentioned earlier with input and output being factored in). Figure 7 contains a summary of both our Weighted and Unweighted PageRanks as well as the sales figures. You can see that in general our Weighted PageRank increases as Sales does.

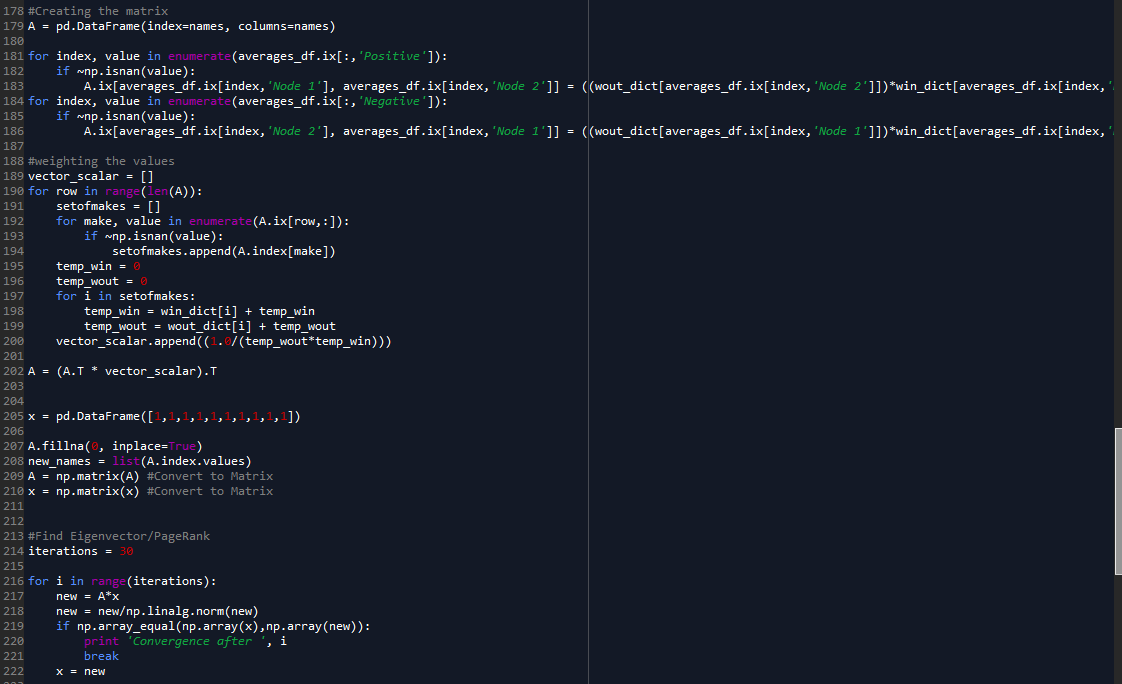


Figure 2: Calculating Eigenvalues

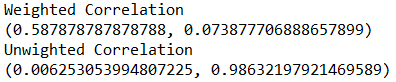


Figure 3: Correlations for Data from Part A

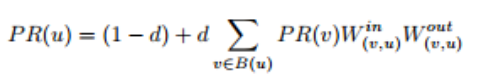


Figure 4: Equation for Weighted Page Rank, Source <http://people.cis.ksu.edu/~halmohri/files/weightedPageRank.pdf>

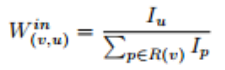


Figure 5: Equation for In Weight, Source http://people.cis.ksu.edu/~halmohri/files/weightedPageRank.pdf

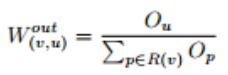


Figure 6: Equation for Out Weight, Source <http://people.cis.ksu.edu/~halmohri/files/weightedPageRank.pdf>

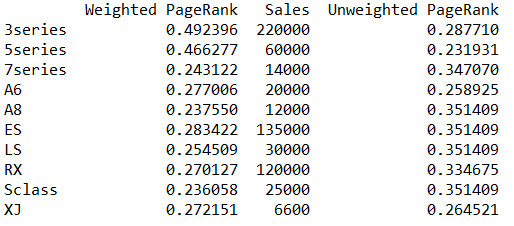


Figure 7: Summary of Ranks (Part A Sentiment)

**Task C.** The above sentiment scores above were obtained by manually reading each post. The file “Assignment 3 Edmunds Posts.xlsx” provide a bunch of actual messages (combine the worksheets). Your task is to automate the sentiment extraction from each post. As in tasks A and B, focus on the same 10 models (note that other models may also be mentioned, but that they should be ignored).

Write one or more python or R script(s) to generate sentiment scores for the 10 models just as in the **sentiment scores** worksheet. This will be an unsupervised approach. One possibility (but not theonly one) is to take the dictionary of SentiStrength (along with the default sentiment scores) and use it as inputs in your script(s). Your script should consider lemmatization (e.g., liking and liked must be treated as the same).

Generate sentiment scores with your script(s), find weighted PageRank of each of the 10 cars and correlate with the sales figures above. How does the correlation of this automated approach compare with that of manual scoring in task B?

We divided our unsupervised sentiment analysis task into three parts. First, we generated tokens. We chose to lemmatize to maximize the accuracy of our sentiment analysis by maximizing the number of words for which it knew sentiment scores. We also removed stop words (except ‘not’) to ensure that the tokens we retained were meaningful. We chose to keep the word ‘not’ because it is an important modulator of sentiment—“not good” is different from “good” and from “bad”.

Our next task was to find relevant tokens. We noticed that certain car models were often referred to in multiple ways: For example, “LexusES”, “ES”, and “ES330” all referred to the “ES” model. We standardized across all the posts, again to make sure we captured as many of the relevant sentiments as possible. After standardizing, we searched across all tokens for our models and pulled two tokens before and after, for a total of five words (two tokens before, the model token, two tokens after) per mention of each model. We found two to keep enough detail without crossing too much into another model’s territory, as it would be very difficult to disentangle sentiments across two models.

Our last task was to determine sentiment. We used a sentiment analyzer on each set of five relevant words to determine sentiment across that set. This sentiment was then used for that model for that review. In the case where a model was mentioned multiple times in the same review, we appended the sentiments; we assumed that people who discussed a particular model many times in the same review felt very strongly about it.

After processing the sentiments, we created a matrix similar to the hand-coded one we were given for Task B. The weighted and unweighted PageRanks for our generated sentiment scores are shown in Figure 10. Spearman correlation for unweighted PageRank interestingly was high, at 0.62; however, weighted PageRank was inversely related to sales (-0.79) (See Figure 9). Weighted PageRank has a high correlation, but not what we were expecting because it is negative. Doing some deeper thinking and evaluating the formula for weighted PageRank, my only thought is that some nodes were artificially pushed upward because they were referred to by other very important nodes. For example, our highest weighted PageRank was XJ at 0.917, you can see from Figure 8 that there is only one node pointing to it, 7 Series. While XJ does not appear to be a very important node on its own, 7 Series is actually quite an important node and thus increases the PageRank of XJ because of it. For unweighted PageRank we get a pretty good correlation with this method, most likely due to the lack of references between nodes. In our previous example we had a lot of connections between nodes, so many that unweighted PageRank might become useless since it factors in only the existence of the relationship.

In any case, our weighted PageRank does in fact have a high correlation (and significant p-value), just opposite from what we would expect. So there is some predictive power in this automated sentiment extraction and PageRank calculation. In fact it seems to have a higher correlation than the manual review.

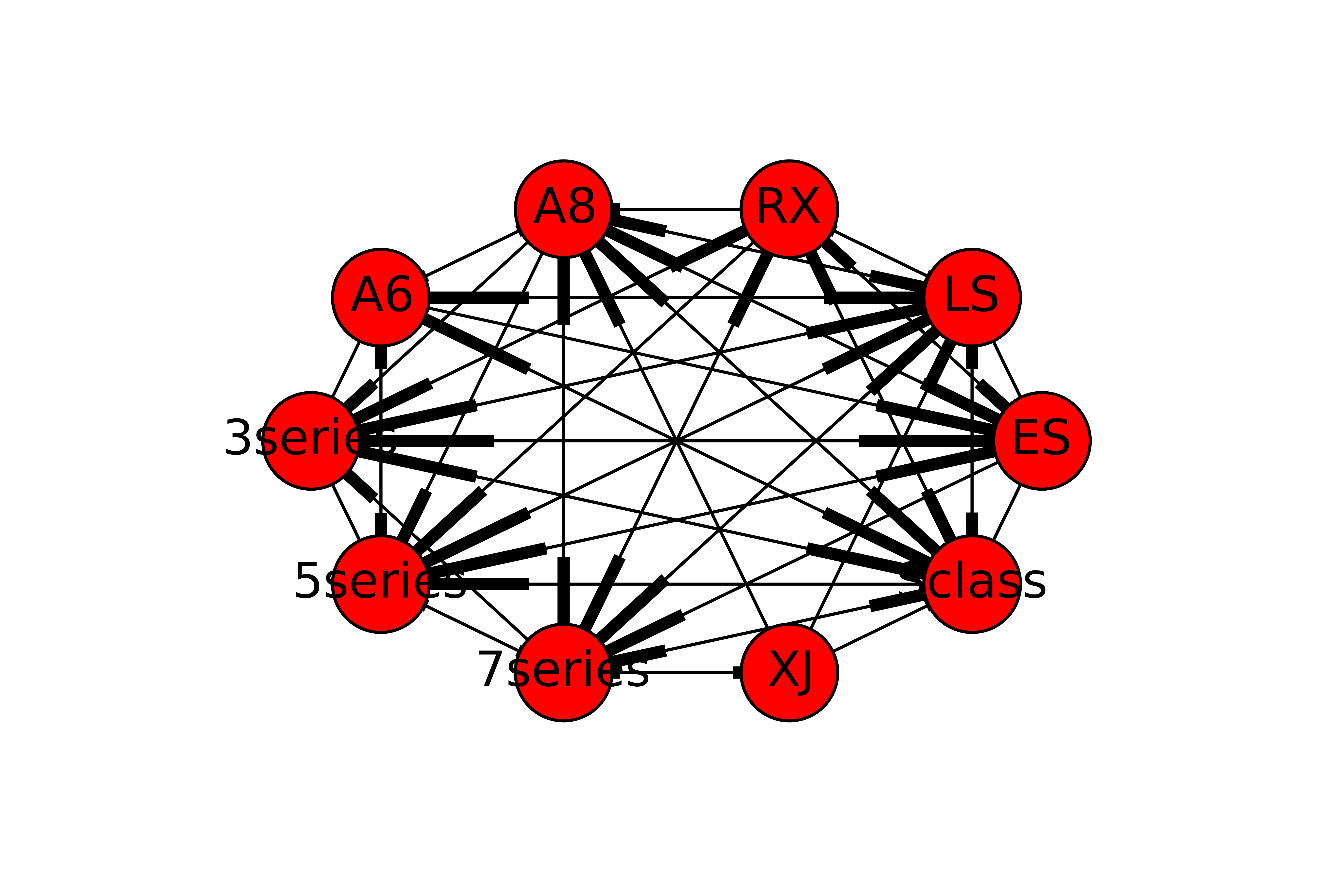


Figure 8: Directed Product Network (Part C)

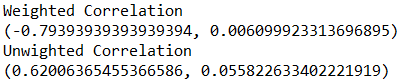


Figure 9: Correlations for Data from Part C

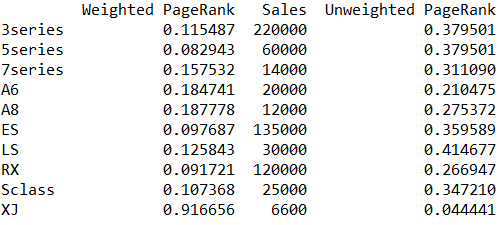


Figure 10: Summary of Ranks (Part C Sentiments)