Assignment I – Clayton Preston

# Prerequisite Information

The assignment was completed using python 3.7. The Pandas library was heavily used while some others like numpy and random were used as well. You can very easily run this script using the Anaconda installation of python.

# Part I

*Import and pre-process the dataset with customers Download the groceries.csv dataset from moodle. This dataset contains demographic characteristics of supermarket 10000 customers along with a list of groceries they bought. For any numerical missing values, you should replace them with the average value of the attribute in the dataset (keeping the integer part of the average).*

Part one of the assignment is fairly straightforward. We download the file provided in Moodle and create a pandas dataframe with it.

I looked through each column to identify potential issues; blanks, NaNs, irregular values, etc. I found that both Age and Income had – not null, but– values of type string which contained a space. As stated in the instructions, for each column, I replaced the space with a null value, then proceeded to fill in the null values of the columns with the average value of that column.

Text

Description automatically generated

*Preview of imported pandas dataframe*

Lastly, in anticipation to analyze sets later on in the assignment, I changed the Groceries columns into a more analysis friendly format. I took the one string value i.e. ‘a,b,c,d,e’ and turned it into a list within the pandas cell i.e. [‘a’,’b’,’c’,’d’,’e’]. This will allow us to more easily union and intersection these values later on.

Table

Description automatically generated with low confidence

*Preview of Groceries string vs. list*

# Part II

*To assess the similarity between the customers you could form the dissimilarity matrix for all given attributes. As described in lecture “Measuring Data Similarity”, for every given attribute you first distinguish its type (categorical, ordinal, numerical or set) and then compute the dissimilarity of its values accordingly. For set similarity use the Jaccard similarity between sets. Then, you can calculate the average of the computed dissimilarities in order to derive the dissimilarity over all attributes. Depending on the machine used to implement this assignment you should decide whether it is feasible to compute the dissimilarity matrices, or, have the computations performed on-the fly for a pair of customers.*

I tried to create a matrix for each attribute comparing all customers, but after waiting for 30 minutes for one attribute to complete, I decided that it was computationally too expensive. Nonetheless, I will explain the process which I would have proceeded with if I had a more powerful computer.

You can see my attempted matrix script below. What I’m doing is the following: first I create a blank matrix the size of all my customers (10,000 x 10,000), then for each row in that matrix – i.e. for each customer- I look at all other customers and would run the respective calculation for the attribute type. I will explain these calculations in further detail below.

Text

Description automatically generated

*Attempted Matrix for Sex Attribute*

In lieu of spending the next week of my life sitting at my computer waiting for my homework to load, I decided to create functions which compare any two customers to one another. This is significantly faster and with these tools in place, as you will see later one, you are able to iterate through all customers using them; if it were absolutely necessary to run this for all customers, you could do so with one loop instead of running each individual function.

Dissimilarity Functions

**Categorical**

Comparing categorical attributes is very simple. You compare the two strings; if they are equal, they receive a 0 score – meaning they are very far from being dissimilar, as they are equal- and if they are not equal, they receive a 1 score.

Graphical user interface, text, application

Description automatically generated

*Comparing Marital Status Between Customers 90 and 2*

**Numerical**

To compare the dissimilarity between two numerical values, we take the difference between the absolute values of the attributes and divide that by the difference between the maximum and minimum values of all possible attributes.

Graphical user interface, text, application

Description automatically generated

*Comparing Income Between Customers 1 and 1000*

**Ordinal**

Ordinal values are slightly more complicated, but are ultimately the same as numerical. The only difference is before we can perform the mathematical function, we must first decode the values. This is a uniquely human job as only we know which words in our language go in which order. So we order the strings in ascending order, assign them values starting with 1 and increasing, and decode each string in the list of attributes with these values. This process turns the ordinal values into numerical.

Graphical user interface, text, application

Description automatically generated

*Comparing Rating Between Customers 111 and 11*

**Sets**

As stated above, we have created ‘a list of lists’ for the grocery items of each customer. To calculate the dissimilarity, we take the number of items they have in common and divide that by the total number of ***unique*** items in the combined list of both customers’ lists. This gives us similarity, but as all other functions output dissimilarity, we must convert this to a dissimilarity score. To do this, we check if the value is 1 - in this case we change it to 0 – and if not, we take 1 minus the value.

Graphical user interface, text, application

Description automatically generated

*Comparing Groceries Between Customers 10 and 23*

Finally, to combine all of these dissimilarity scores, I have created a function which takes the values of all of these scores and computes the average. This tells us, given all attributes, how similar are two customers: **1 meaning they are the most dissimilar possible and 0 meaning they are perfectly similar.**

# Part III

*Using the implementation of the previous step, you will calculate the 10-NN (most similar) customers for the customers with ids listed below: 73, 563, 1603, 2200, 3703, 4263, 5300, 6129, 7800, 8555 For this task your script must take as input the customer-id and return the list of her 10 nearest neighbors (most similar), along with the corresponding similarity score.*

Using the function we created to produce an average dissimilarity score between two customers, we will create on top of that a function to create a dataframe of the top 10 nearest neighbors (most similar) customers to them.

To do this, we take an input of a single customer and compare them to all other customers. With this list, we sort it to get the top 10 most similar (i.e. the 10 ***lowest*** scores - as our function shows dissimilarity) and return a dataframe with this information.

Table

Description automatically generated with low confidence

*Returning the 10 NN for Customer 64*

To run this on a list of customers, we simply loop through the list of customers using the function we created.

Text

Description automatically generated

*Looping through the list of customers using the Top10 function*

***The results of the 10 NN for the customers requested can be found in the file named ‘Top10.csv’***

# Part IV

*For this assignment you will implement a classification algorithm which, for a given customer, will predict his rating (poor, fair, good, very good, excellent) for the supermarket. In order to implement the classification for a given customer you need to:*

1. *Calculate the similarities between the given customer and all other customers and compute his 10-nn (most similar) customers. IMPORTANT: In the similarity calculations for this step you need to exclude the customer rating attribute.*
2. *Based only on the 10 most similar customers computed in the previous step, predict the customer rating using:* 
   1. *The average rating of the 10 most similar customers (rounded to the nearest integer).*
   2. *The weighted average rating of the 10 most similar customers (rounded to the nearest integer).*
   3. *Weighted average rating = ∑ similarity(i) ∗ rank(rating(i)) 10 i=1 ∑ similarity(i) 10 i=1*
   4. *Where:*
      1. *rating(i) = the rating of the i-th nearest neighbor (i=1 for the most similar customer)*
      2. *similarity(i) = the similarity of the i-th nearest neighbor with the given customer*
3. *For the evaluation of your classification algorithm you will use the 50 first records of the groceries dataset and predict the rating for them. Then, for all n=50 records calculate the Mean Prediction Error for both prediction methods. Mean Prediction Error = ∑ |𝑟𝑎𝑛𝑘(Predicted rating(i))−rank(True rating(i))| n i=1 n*

Here we must leave out Customer Rating since we are trying to predict it and must tweak our previous functions a bit to accommodate this. Luckily, it’s as easy as taking the Customer\_Rating attribute out of our ComboScores function. Now we have a new function that takes the average value of all scores except Customer Rating.

In both our Average and Weighted Rating Prediction functions, we will take as input one customer and give as output the predicted Customer Rating. To do this, we run the modified Top10 function (without Customer\_Rating) to get the 10 most similar customers to the one of interest. For those 10 customers, we get their Customer\_Rating and decode it so that the values are numerical. With this information, we take the average or weighted average – using the similarity to our customer as the weight – of all of their Ratings to predict the value of our customer.

Graphical user interface, text, application, email

Description automatically generated

*Returning the Average Predicted Customer\_Rating for Customer 64*

Graphical user interface, text, application, email

Description automatically generated

*Returning the Average Predicted Customer\_Rating for Customer 64*

In the example above, we see that customer #64 has an actual rating of Fair (2) and we have predicted Good (3) and Very Good (4) with the Average and Weighted Average prediction methods respectively.

To evaluate this method, we will take a sample of the first 50 customers from our original data set. For these 50 customers, we will run our both our Average and Weighted Average Prediction functions and create two new columns in the dataframe called *Avg Prediction* and *W Avg Prediction* respectively containing our predicted customer rating values. With our predicted value and the actual value both in the same dataset, we can calculate the Mean Percentage Error of our prediction to evaluate how accurate our methods were.

Unfortunately, after letting this process run for an hour and a half, not even the first of the loops completed, so we could not complete the evaluation, but the code can be seen below. However, we can calculate MPE for both the Average and Weighted Average predictions for our single customer example above: We see that Customer 64’s rating is fair (2) and we predicted 3 with the Average method and 4 with the Weighted Average method yielding an MPE of 50% and 100%. We are looking for scores that are less than 50% - preferably as low as possible – so we would say that these are not very good scores.

Graphical user interface, text, application

Description automatically generated

*Process for MPE Evaluation*