Neural Network Matching Algorithm Project

The goal of this project is to create an algorithm to match an individual to their potential partner. The process of this project included researching existing algorithms, determining which traits are important and coming up with my algorithm.

Initially, I started this project by doing research on various dating websites and taking notes of their algorithms. However, there was not enough information online. Noticeably there were many posts of peoples' experience with online dating and how their matching algorithm tends to mostly match them with those with similar characteristics including hobbies and demographics. In other cases, people would explain that they found their significant other from the online dating website, but they did not match.

Then I decided to research which traits seem to be relatively important when it comes to matching. The theoretical model that I wanted to focus on was the rational choice model. The rational choice model focuses on a more natural and evolutionary approach to mating. It mainly focuses on finding mates based on physical attractiveness, values and interests, socioeconomic status, personality, and honesty. There are inherent biases when it comes to matching, and an individual is more likely to prefer someone who has similar characteristics or someone who has higher quality characteristics.

Originally, I had the idea of web scrap for the data online, but I was able to find a free dataset online. The dataset that I ended up using was from the OkCupid profile dataset from R's library. This dataset is publicly available and has been used for publication. Additionally, there were two main concerning issues with this dataset. The dataset did not have any dependent variable, and there was limited description of each profile.

Based on my research, the key focus of this algorithm was to make it flexible. The algorithm that I thought that Artificial Neural Network (ANN) would be the best machine learning algorithm for this project over regression, k-nearest neighbor, k-means, and support vector machine. ANN allows each variable to have a different weight and each weight can be adjusted. Therefore, the ANN can improve the accuracy of the model based on the weights applied to the model.

What I learned

Before doing this project, I had already known a decent amount of python. Most of my experience with python was with clean data and analyzing the data with python. I also knew how to create functions and loops. These combinations of experiences had helped me with completing the project. However, there was still a lot to learn when working on this project. I learned how to use text data for analysis and how to use the artificial neural network machine learning model. When working on the project it required me to use what I had learned and compiled it into the program.

I needed to analyze the summary description for each person in the data. One way of approaching this problem is to text mine. I learn that to text mine, I would need to strip my summary data of any common words, stop words, punctuation, and numbers. I also need to lowercase all the words and lemmatize the data. By doing all of this, it reduces the cost of processing, but it also reduces the number of words I would have to examine. I also learn about sentimental analysis which examines the opinion of the text. Even if I perfectly selected the keywords, I can do not know the context of the description. They are saying all these words, but it could be a negative context rather than a positive context. Additionally, I learned to use term frequency-inverse document frequency vectorizer (TF-IDF vectorizer) which can determine the importance of the words for each observation relative to the words in the whole data set. The first part TF represents the term frequency is and IDF represents inverse document frequency. TF-IDF multiples both parts, and it reflects the important a word relative to the collection of documents.

I also learned the general process and calculation with artificial neural networks. The basic concept of a neural network is the dot product of randomly generated weights with the set

of inputs. The output would tend to be placed into a sigmoid function which will then return a probability between 0 and 1. The sigmoid function is defined as where z is where we plug the value from the dot product of the input and weights. Notability, in my project I did a simple artificial neural network such that there only one set of weights. I could have added more than one set of randomly generate weights. After getting the results from the ANN, it would be able to adjust for the weights, by taking the dot product of the outputs with. The output from adjustment will be added to the existing weights (the adjustment values could also be negative).

After working on this project, I was able to hone my Python skills, and I was able to learn how to text mine, and the use of an artificial neural network. I also learn that when cleaning up the data I tend to over-complicate the process, but the majority of the data wrangling can effectively manipulate with what was being taught in econometrics.

The Coding Process

Section 1: Preferences & Survey Questions

In the first part of this section, participants will fill out their questions to prefer matches. The preference questions will be used to select a subset of a potential match. It is reasonable to assume that if an individual were to be matched with someone outside their preferences then they would reject them. For instance, if person A (who wants kids) with person B (who does not want kids) then person A will reject person B. Therefore, the preference questions will be used to eliminate matches in which participants are biased against. The following questions will be asked for the preference questions:

- What is the minimal and maximum age do you prefer you match to be?
- Would you be willing to date someone with kids?
- Would you date someone who does not want kids?
- What ethnicity do you prefer to be matched with?
- Does religion matter to you?

In the second part of this section, participants will fill out their demographic information and answer some survey questions. I will then ask the participant to write a quick 1-2 sentence summary about themselves. The demographic and summary information will be used to match the participants with the individual in the dataset.

Demographic questions will include the following:

- What is your gender?
- What is your sexual orientation?
- What is your ethnicity?
- What is your religion?
- What is your height?

- What is your bo dy type?
- What is your education level?
- What is your income level?
- What kind of job do you have?
- Do you have a specific diet?
- Do you have any offspring? Or want any?
- Do you want kids?
- Do you have any kids?
- What is your zodiac sign?
- Do you smoke?
- Do you drink alcohol?
- Do you do drugs?

Section 2: Cleaning Summary Data, Sentimental Analysis, and Selecting Important Words

For section 2.1, I cleaned the text data and in the process of cleaning the data, I also selected important words that I believe will be valuable for the matching algorithm. Firstly, I replace any text data that contains “nan” or if it is equal to a float with a blank string. I also remove common words, stop words, punctuations, and numbers from the text. Common words are words that are overly used in the text data. Stop words are filter out because they are generally the most common words in a language. Numbers were also removed because they have no relevance to keywords in their summary. Then I lowercase any capitalized words because the term frequency-inverse document frequency vectorizer (TFIDF Vectorizer) module does not think the word “Sports” and “sports” are the difference. Also, lemmatize the text data which removes inflectional forms and derivationally related forms of a word. For example, the mapping of the text “the boy’s cars” result in “the boy car”. By this point, the text data should only consist of words that are relevant and remove any redundancy with related words.

For section 2.2, I use the function sentiment from the module textblob. The function automatically processes the text data and determines if the text is positive, negative, or neutral opinions from the text. Using the value provided from the function I create the variable

sentiment. Note, the function only provides numerical values ranging from -1 to 1. A -1 would indicate the text is perfectly negative, a 0 would indicate the text is natural, and a 1 would indicate the text is perfectly positive.

In sectional 2.3, the process of extracting important keywords from the text was messy. It required me to manually select words and put them in a dictionary. First, I ran TFIDF Vectorizer and selected the top 200 words. Then I create a broad dictionary that consists of broad words as the keys, and any top 200 words that were similar. By this point, I had about 50 general words I thought were useful. Then I rerun the process until the majority of my word's popup in the TFIDF Vectorizer. If it doesn't pop up then I will remove the words that are unnecessary/useless then rerun the TFIDF Vectorizer. Additionally, if I find a word that was similar to my keywords, then I would add it to the dictionary. I stop this process until 43 of my words had appeared in the TFIDF Vectorizer.

Section 3: Funnel Class and Reduce Function

The class funnel imports the following information from the individual's preference and demographic questions. The reduce function is just one giant conditional statement. It will only select potential matches who meet certain criteria. It imports the following information from memory:

- Their Orientation
- Their Sex
- Their minimum and maximum age preference
- Their ethnic preference
- Their stance on religion
- Their religion
- Their stance with dating someone with kids
- Their stance with dating someone who does not want kids

Using this information, I construct the reduce function which will reduce the existing dataset to a subset. For instance, if the individual has the following information:

- Their Orientation: Straight
- Their Sex: Female
- Their minimum and maximum age preference: 27 - 35
- Their ethnic preference: White, Asian, Hispanic / Latin, and Pacific Islander
- Their stance on religion: Does not matter
- Their religion: Other religion
- Their stance with dating someone with kids: No
- Their stance with dating someone who does not want kids: No

Then the reduce function will select a subset of matches who are:

- Male
- Straight
- Between the ages of 27-35
- White, Asian, Hispanic / Latin, and/or Pacific Islander
- Religion does matter or does not matter
- Any religion
- Does not have kids
- Want kids

Section 4: Personalized & Pooled Function

The personalized function will be used to reconstruct the reduce dataset from above. The reconstructed dataset will be in terms of the participant (i.e. it will compare the person's information against his/her potential match). The pooled function will also be used to reconstruct the reduce dataset after using the reduce function. The reconstructed dataset will be in terms of the person's potential matches (i.e. it will compare the potential matches' information against the participant's information.)

Both functions will take the existing data and create dummy variables for data on their diet, drinking habit, drug habit, smoking habit, jobs, ethnicity, zodiac signs, and religion. These

variables will not be used to compare because the artificial neural network would be able to adjust to these preferences.

Section 5: Weights, Sigmoid, and Initial ANN

For section 5.1, I randomly generate a set of weights for each variable for the dataset. I also create the function sigmoid which is the activation function for the neural network. I also created the function sigmoid which is an activation function. The activation function determines the probability of an event between 0 and 1.

For section 5.2, I generate the class ANN and I initialized the following variables: inputs, weights, and outputs. The inputs are the variables from the datasets, the weights are the randomly generate an array of weights, and the output is just an array of 0s. Afterward, I created a feedforward function which is the dot product of the inputs and weights. The dot product is then put into the sigmoid function which provides a probability of the match. Additionally, I made another class ANN2 which similar to ANN, but it takes another variable y and has another function backprop. The variable y represents the person's response to their match (1 if they match the person, 0 if they do not.). The backprop function adjusts the weights based on the response of the individual.

Section 6: match_output, scores_df, initial_match_info, initial_match_df, loop_match

Both match_output and scores_df are relatively the same function but returns a different output. Both the two functions create a data frame using the outputs from ANN. The functions take the index from the personal and pooled data, the ANN output values from the personal data, and ANN value from the pooled data. They also create a new variable 'Multi Score' which multiply the personal and pool outputs. This is also the variable that will be used to determine the

ranking of the person's match. Match_output returns a single match by randomly selecting their top ten potential matches while the score_df returns the entire data frame.

Both initial_match_info and initial_match_df function takes on the personal, pool, and weights arguments. They both also run ANN on with the two data sets, and then it extracts the outputs from the ANN. Initial_match_info will use the outputs from ANN to run match_output while initial_match_df will scores_df.

The loop_match takes the person's response to their match as the argument. For this function to work, both initial_match_info and initial_match_df needs to run first. This will provide the person's initial match, and from this point forward the function will delete their match from the dataset. Then it will run ANN2 on their match and uses their response to adjust the weights. Then it'll run ANN on updated data (without their previous match) and the function will then select a new match.

Section 7: Running The Matching Code

For section 7, I start from the beginning with using the funnel class and reduce function. This returns a subset of a dataset, which I will also make a copy. I will also drop all the dummy variables that were generated from the reduce variables, and variables that I do not need for printing the person's potential match data. Then I will use the functions personalized and pooled to generate a new dataset for the matcher's information and the matched information. I will create the variable match_info which uses the initial_match_info to generate a person's first match. Then I will create another variable match_score which provides a data frame of all the matches score. I will also generate a pandas data frame that collects the data on the person's

match, and I will generate a list of the person's response to their match. Once the person stops matching their response will be added to the data frame.

When running the code, the code will print information about the person's match only if the information does not equal to 0. Then the person can decide if they want to match or not. Afterward, they can decide if they would want to continue to match. Each time they decided they match or did not match with the person the weights will change. Therefore, the longer the person was to match the more accurate the result will become.