Scaling Mentoring for Graduate School: An Algorithm to Streamline the Formation of Mentoring Circles for the GradTrack Scholars Program

Abstract

The GradTrack Scholars program prepares undergraduate students for graduate school while building a community of students excited to pursue advanced study. GradTrack uses mentoring circles – a proven model for supporting individuals pursuing graduate school [1], postdoctoral roles [2, 3], and faculty careers [4]. In GradTrack, each mentoring circle unit consists of two graduate student mentors and 6–8 undergraduate mentees, where each unit is part of a scalable mentoring system. In 2024, GradTrack included 26 mentors and 100 mentees. Currently, the manual process of matching mentors and mentees is time-consuming, requiring 4–8 hours of administrator-assigned matching per program. While tools exist for grouping students into teams for class projects [5], most widely used tools are commercial, fee-based, and/or are more complex than needed. Therefore, we identified a need for a simple, open-source solution specifically for mentoring structures.

The purpose of this research project was to develop a streamlined method for the formation of mentoring circles as GradTrack continues to grow, specifically by leveraging widely used statistical algorithms such as k-means clustering. Using data from three years of manually created mentoring circles, we developed a python algorithm which uses groups of mentor pairs as seeds for clustering mentees. To do this, k-means clustering and one-hot encoding were used to create balanced groups of mentors and mentees based on similarities of interests and a specified number of groups. The output of this python algorithm is a list of mentees grouped by similar interests and mentor group number.

This study evaluates the benefits and limitations of using a computer-based and automated method to assign mentees to mentoring circles, and shares the process of development as well as a protocol for use. The key implication is a more efficient process for creating mentoring groups with a reduction in human time and subjectivity, which will support the continued growth of the GradTrack Scholars program and other mentoring circle program structures. To the author's knowledge, this is the first study to develop a k-means clustering algorithm applied to mentoring purposes. Future study should evaluate the comparison between manual and algorithm based mentoring group formation in connection to assessment of mentoring group success.

Introduction and Literature Review

Background on the GradTrack program and mentoring circles

The GradTrack Scholars program focuses on preparing undergraduate students for graduate school, with a specific focus on increasing access to graduate education and broadening participation in engineering. The program was established in 2020 and works by developing small mentoring circles that are units in a scalable network. The mentoring circles are one part of the GradTrack program structure, which includes scheduled meetings in the fall semester with set curriculum and deliverables related to graduate school preparation [6]. Pre- and post- program surveys of participants show that a) GradTrack mentees feel more prepared for graduate school, b) GradTrack increases mentees' feelings of belonging with graduate students and to the engineering community at large, and c) GradTrack increases professional development for graduate student mentors [1]. The goal of GradTrack is to build community and prepare the next generation of graduate students from all backgrounds – the mentoring circle structure is an integral component for success.

However, the process of assigning and developing mentoring circles takes time and can be biased. First, applications are received from both mentors and mentees to the program. The application includes questions about the applicant's interest in being part of the GradTrack program. Once mentors and mentees are reviewed and accepted, an email is sent to each participant requesting that they provide additional demographic information that is not requested in the application, but will be helpful for mentoring circle creation. Applications are reviewed independently from the mentoring circles creation process.

Once all relevant information is gathered from participants, the current process of forming mentoring circle groups consists of pairing up mentors so that there is a balance of genders in each circle (one man and one woman, whenever possible). Graduate majors are also taken into consideration, especially if there are many undergraduate mentees requesting a specific graduate major (and not enough mentors in this area). When this happens, and we have more mentees than mentors in a given discipline, then graduate student mentors of similar majors might be split up between mentoring circle groups to expand reach. Once mentors are paired, then comes the lengthy process where an administrator goes through and manually adds the admitted mentees to these newly made mentor groups.

To make circles, student information is compiled from application data and the post-acceptance demographic survey. A number of factors are systematically used to group mentees with mentors, including undergraduate university, undergraduate major, gender, race/ethnicity, parents' education level (first generation college student status), and prior research experience. Additional factors were typically given priority for group formation, including using similarities based on graduation date, level of interest in graduate school, and graduate degree being considered. Circles are complete when the administrator decides that the best possible combination of 6–8 mentees and 2 mentors has been achieved. This process typically can take 4–8 hours in order to get even and well-balanced circles.

Characteristics to consider when making mentoring circle groups

When developing mentoring groups, consideration must be made regarding how students will be grouped together. It has been shown that different individual characteristics will affect the success of a mentor-mentee relationship, including matching by shared experiential, surface level and deep-level characteristics [7]. Specifically, best practice from the literature suggests that mentoring matching should be based on deep-level characteristics. These deep-level characteristics include aspects that are not necessarily visible to the eye and include a student's interests, values, personality, and more [8].

It is worth noting in the literature that surface level characteristics used in mentor matching, such as those based on gender, race, or ethnicity, do not exhibit an enhanced mentorship quality experience [7] - [10], and also simultaneously emphasizes the importance of culturally aware mentoring. However, it is also shown that peer mentoring between women in engineering increases the belonging, confidence and retention of these same women in engineering over a multi-year study [11]. Taken together, this demonstrates that while mentor matching best practices depend on deep-level characteristics, representation is still an important factor in higher education and mentoring relationships [12] - [14].

For GradTrack, the decision-making process for selecting specific factors for grouping mentees and mentors includes those characteristics falling within the experiential, surface level, and deep-level areas. This mentor matching paradigm is built upon the foundation of deep-level characteristics and the goals of the GradTrack program itself, which values students who want to learn more about pursuing a graduate education and have an interest in doing so.

What is known about using algorithms for making teams?

Due to time constraints and subjectivity, program administrators identified the need for an alternative form of matching mentees with mentors. Grouping students together using a computer-aided support is not new; however much of the previous use includes teams and course-based applications. For example, team activities in courses have been shown to benefit from pre-determining groups [15], however it is noted that this does not occur as often, as instructors may not have the time to work through all of the logistics to put students into groups [5], [16], [17]. Significant advances have been made to develop a computer-generated team maker for courses through the CATME project [18], which includes team maker [5] as well as ways for team members to complete peer evaluation [19] - [22].

While these resources are excellent, they require an institutional login in order to access and require much more information than is needed for development of mentoring circles. There is a demonstrated need for algorithmic clustering methods [23], [24], but previous methods focus on clustering only students into student-based groups or teams. With student only groups, all group members come from the same population. With mentoring groups, the groups consist of both mentors and mentees, which are two different and distinct populations. To the authors knowledge, there have been no studies that demonstrate the development of an algorithm or protocol for clustering students specifically for mentoring using k-means clustering.

Types of algorithms known to be used in clustering

The first step in developing the algorithm was selecting the clustering method to use. The two most likely candidates for clustering units in a large dimensional vector space include agglomerative hierarchical clustering and k-means clustering.

Agglomerative hierarchical clustering is a technique that groups data points (students) into clusters (mentoring groups) by making a hierarchy of clusters based on their similarities. It continues to group data points by building tree-like structures, with similar data points grouped together. This is a natural fit for forming groups of similar students, since groups are formed by sequentially grouping students closest to each other in the defined vector space, and no initial group centroid is required. However, for mentoring circle creation we actually do want to start with a centroid (mentors) and hierarchical clustering cannot consider mentor pairs as part of the clustering algorithm. Therefore, it is not possible to use only hierarchical clustering for matching mentors and mentees.

K-means clustering is a machine learning technique that groups data points (students) into a pre-defined number (k) of clusters (mentoring circles) based on how close the data point is to the nearest centroid, or center of the cluster. The centroid is defined through different weights on specific mentor characteristics or categories. This allows for more adaptation, as a specific number of group centroids are required and defined, and then data points are moved into clusters with the closest group centroid.

Previous work has joined these two approaches, using hierarchical clustering to create initial centroids that would then be used with k-means [16]. However, for creating mentor groups the mentor pair centroids are a natural choice for initial cluster centroids. The traditional weaknesses of k-means clustering, centroid number and centroid initialization, proved to be a strength for creating a pre-specified number of groups centered on mentor pairs. Both hierarchical clustering and k-means clustering were implemented initially in this study, but ultimately, k-means clustering was selected due to the compatibility of forming mentor groups around mentor pairs.

Here, we seek to answer the following research questions:

RQ1: Can we make an open access algorithm and protocol that can quickly and objectively cluster mentees into mentoring group circles?

RQ2: How much faster is the algorithm method compared to the manual method?

RQ3: What are the benefits and limitations of using a computer-based and automated method to assign mentees to mentoring circles?

Methods

Developing the Algorithm

The algorithm was designed to replicate the manual group formation process as much as possible, while drastically reducing the time required. To do this, the same factors were used and weights were assigned to match administrator grouping preference. This code was developed based on previous cohort years of mentee and mentor data (2021-2023) with weights refined to create groups similar to what would have been created manually. In the final product, the factors and weights can be changed for each use.

The input data is in the form of three csv files, one containing the groups as defined by manually paired mentors, another containing a subset of student data to be used in group formation, and a larger student file containing all student data. The student data was intentionally split into two files so that the group formation process would only run on a subset without any identifying or sensitive information, potentially allowing the group formation code to be prepared by an employee or assistant without access to personal data.

The Python script takes these three csv files as input and returns one as output. The input files are a manually paired list of numbered mentors with data cleaned and coded for the algorithm, a numbered list of students with a subset of data cleaned and coded for the algorithm, and uncoded student data with a column of student numbers matching the previous file. The Python code itself is written using cell structure for ease of debugging in Spyder or Jupyter and is thoroughly commented on for use in the .py file. A workflow diagram visualizing the flow of data from the computer hard drive to computer memory via the algorithm is shown in Figure 1.

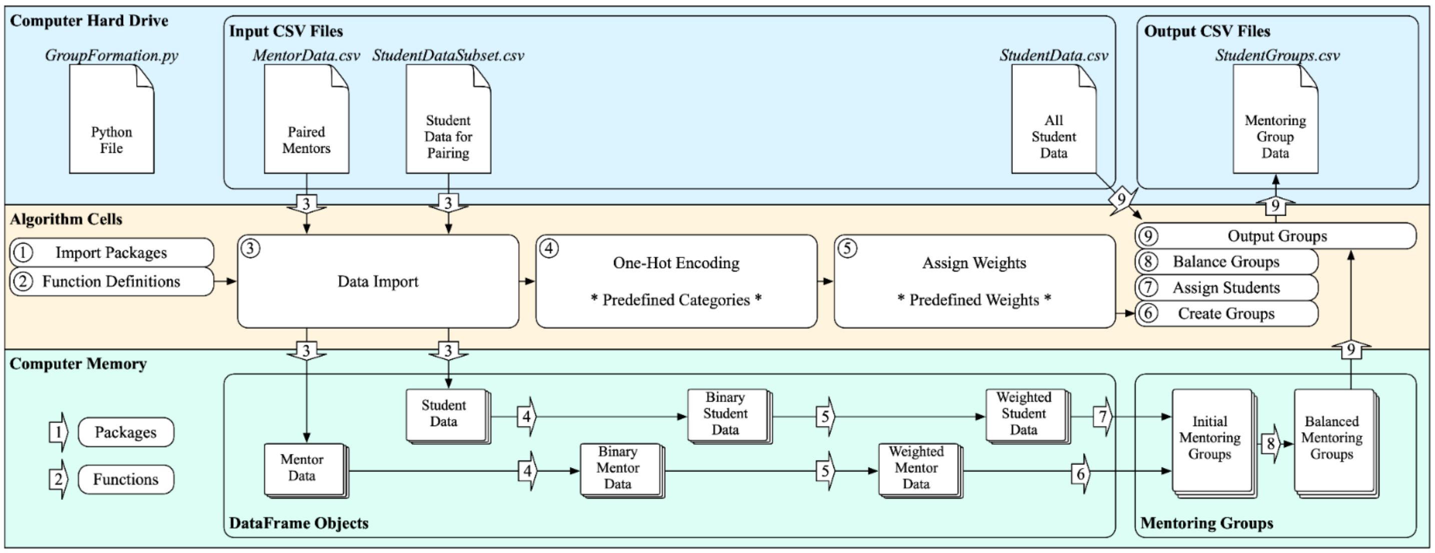


Figure 1. A workflow diagram visualizing the flow of data from the computer hard drive to computer memory via the algorithm. The top bar indicates files stored on the computer hard drive, the bottom bar indicates data stored in computer memory, and the middle bar indicates sequential cells of the algorithm. The sequential cells are labelled with numbers in circles and actions performed by those cells are labelled with numbers in arrows.

The first two cells import packages required for code completion and contain definitions for the functions used internally. The third cell imports the csv files of coded mentor pairs and coded student data as pandas DataFrame objects with the data as strings. This cell requires the file paths of the first two input csv files, the sample code assumes these are in the same directory as the .py file. This cell also contains debugging code which can be uncommented to check that the csv files were read correctly.

The fourth cell uses one-hot encoding to match the coded data in the DataFrame objects to the predefined categories. This returns DataFrame objects with the data as binary integers. This cell requires a list of the categories being evaluated, which must match the first row of each input csv file. After this step the size of the DataFrame will increase significantly. Each column is a category in the input file, and after one-hot encoding each column is an encoding. A single category will be mapped to multiple encodings. For example, in the sample data, the Gender category was encoded with the options of Man, Woman, and More. Thus, after one-hot encoding, the Gender category column will be expanded into three encoding columns: Gender_Man, Gender_Woman, and Gender_More. The University category column in the sample data set contains 62 unique entries and is thus expanded to 62 encoding columns.

The fifth cell assigns weights to each category to define the relative importance of each category. This returns DataFrame objects with the data as numbers. The weights in this cell can be entered differently for mentors and mentees and can be customized for different encodings within a single category. A data category can be weighted equally across all encodings by using the assign_weights function defined in the code or different encodings within a single category can be weighted manually. See the .py file for examples of weighting all universities equally, across all 62 columns, and for examples of treating the three gender encodings differently. Gender encodings may be treated differently if the user is interested in having a random mix of genders in each mentoring circle, as opposed to clumping the same gender encoding into the same group. Higher weights increase the chances of students with the same encoding being placed in the same group and with a mentor with the same encoding.

The sixth cell creates mentoring groups from the mentor pairs by calculating the average centroid of each mentor pairing. The seventh cell assigns students to mentor groups. The user will set the number of mentor pairs in the code. The code will assign students to mentor groups by first calculating the centroid of all encodings for each student and then finding the nearest mentor group centroid. This cell also contains commented out debugging code, which when activated will print out the initial mentor groups and the size of each group. These initial groups are often of vastly different size and some groups may contain no mentees at all.

The eighth cell balances the groups. For each group over the target size, the code identifies the student furthest from the group centroid and assigns them to the closest group that is not above the target size. This cell also contains code which prints out the balanced mentor groups and the size of each group. In testing, the groups are typically as balanced as numerically possible.

The final ninth cell sorts the uncoded student data (input csv 3) into the balanced mentor groups and creates the output file. This output file contains header rows for each group and all data from the uncoded student data with the rows rearranged into the mentor groups. For an example see the sample output file.

Sample Data

For this publication, sample data was generated to demonstrate the algorithm and is available alongside the code on Github. Sample mentee data was generated from previous mentee data by de-identifying mentees and randomly sorting each column independently to create a random but representative set of mentees. The mentees were then manually paired to create the seeds for mentoring groups. Sample student data was generated using a combination of generative AI (ChatGPT) and previous student data. Random student names were generated in ChatGPT by asking for a list of 100 random names from all ethnicities in the US with likely ethnicity and gender for each name (ChatGPT only returned 98 entries). Gender was classified as Man or Woman with 2% of the list being randomly classified with a gender of More to account for transgender and non-binary students. Sample student ethnicity was classified as one of: African American/Black, East Asian/Asian American (e.g., Korean, Chinese), South Asian/Asian American (e.g., Indian, Pakistani), Southeast Asian/Asian American (e.g., Vietnamese, Thai), Hispanic/Latino/a/e, Middle Eastern/North African, Native American/American Indian/Alaska Native, Native Hawaiian Pacific Islander, White/European American, or Not Listed. Sample student university affiliation was randomly assigned from a list of the top 50 engineering schools in the US (as determined by ChatGPT) plus the following target schools added to the list: University of Texas, El Paso (UTEP); University of Puerto Rico, Mayaguez; Morgan State University; Tuskegee University; North Carolina A&T University; Howard University; Chicago State University; University of California, Irvine; University of Houston; University of Texas at Arlington; Florida A&M University; and Florida International University. All other sample student data was randomized from a representative data set. All names were randomly generated and no actual previous mentee or mentor names were used in these sample data sets.

Results and Discussion

RQ1: Can we make an open access algorithm and protocol that can quickly and objectively cluster mentees into mentoring group circles?

Since we have demonstrated the methods of developing an algorithm to make mentoring circle groups, we next established a detailed protocol for its implementation. These results are a step-by-step guide that is meant to be used as a protocol for implementation, whether the user has previous coding experience or not.

This algorithm code, templates, sample data and a link to this paper is also shared on Github (github.com/tompkinn/mentoring-circle-group-formation). This is an excellent step for automation of developing mentoring circles, but it is always advised to check outputs to ensure that students are grouped accordingly.

1. **Download the code that has been deposited on Github.** All required Python packages (NumPy, Pandas, CSV, etc…) are available via PIP (Package Installer for Python), and the code is written in cell structure, which allows for debugging in Spyder or Jupyter.

2. **Compile and clean up all student information**. In our study, student information that is used in the algorithm is compiled from the application itself and from a survey when students accept participation in the program. Therefore, it is important to identify your grouping categories and then compile all relevant student information before starting. This *SampleStudentData.csv* file will be used in step 4. Student information also will need to be cleaned up, and the following rules should be applied:
   a. *Replace special characters in names and universities* (accents, etc…) with English alphanumeric characters
   b. *Recode graduation month and year* (which can be variable in an application) to something more standard. For this study, we coded month and year to one of three options: “Senior” “Younger” or “Dec Senior”. Note that it is not advised to only put the year (or only numbers) as Python may not interpret this as a string.
   c. *Recode Gender* to “Man”, “Woman” or “More”. The “More” category is developed to combine any students who identify as transgender, non-binary, or non-conforming into one encoding.
   d. *Ensure that all universities are spelled out identically*, and update any universities spelled out in shorthand.
   e. *Ensure encodings are using the same values for both Mentors and Mentee spreadsheet*. An example of this is if your Mentee csv file lists “Man”, “Woman” or “More” under the gender category, then your Mentor csv file cannot use “Male” or “Female” and therefore these encodings should match exactly.
   f. If your form has any open response forms or requests to “Please Specify” then make sure these are updated. A helpful tip is that ideally, you would have a drop-down box for responses and reduce as much text entry as possible (to ensure consistency when using the code).

3. **Manually sort and pair mentors.** These mentor pairs will be used as the seeds to define the centroid of the mentee small groups clusters. For mentors, we do not necessarily want

mentors to be paired or clustered based on similarities, since we want to ensure that there is a balance of graduate majors, genders and backgrounds in each circle. For example, sometimes we will have a lack of mentors in a given graduate major, and we must strategically distribute mentors with similar majors across groups. This is in opposition to what this algorithm does, which is clustered by similarities. Therefore, mentors are manually paired for the *SampleMentorData.csv* file to ensure that there is a balance of these categories in mentoring circles. Best practices include:

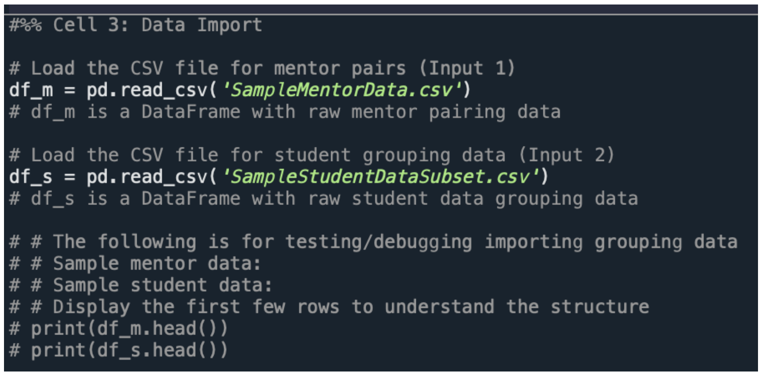
   a. *Pairing mentors based on categories such as gender, citizenship status and major*. For an optimized mentoring experience, it is important to make sure that there is representation spread out across circles. Best practice dictates that there be one man and one woman in each circle, when possible, and that if you have mentors with different citizenship statuses, that there is at least one U.S. citizen in each mentoring pair. This can be helpful for discussions about federal grants and fellowships for U.S. Citizens and Permanent Residents only.
   b. *Code mentors with their mentor groupings.* Code and save them in the *SampleMentorData.csv* file with the naming of m1a and m1b for the first two mentors in mentoring circle one, m2a and m2b for the second mentoring circle, etc…

4. **Save student information in the appropriate file format.** After student data is cleaned up, this information should be saved as usable csv files. Give mentees and mentors a student number to de-identify student names and protect privacy. Student data should be included for the following categories, as the code will weight these categories for matching. The code requires three input csv files and outputs mentoring circle groups as a csv file. The following headings are listed below for each input file.
   a. *SampleStudentData.csv | Input 3)* Student source data for the output file (Input 2 is a subset of Input 3) All of the columns from Input 2 and any additional columns in source data. This file has student’s first and last names and randomly assigned student numbers.
   b. *SampleStudentDataSubset.csv | Input 2)* Student data subset for grouping with the following category columns: ['Student Number', 'Graduation', 'University', 'Major', 'Gender', 'Race', 'Firstgen', 'Interest', 'Degree', 'Research']. This file does not have student’s first and last names, and uses only the assigned student numbers.
   c. *SampleMentorData.csv | Input 1)* Mentor pairs with the following columns: ['Mentor Number', 'University', 'Major', 'Gender', 'Race', 'Firstgen']. More information about mentor information can be found in Step 3.
   d. *Save csv files and GroupFormation.py algorithm in the same folder.* All file names should be identical to the files names above to run properly. This is case sensitive.

5. **Open the GroupFormation.py file in either Jupyter or Spyder.** For our particular use, Spyder was opened using the Anaconda Navigator. After you open the GroupFormation.py file in Spyder, you will need to make sure that Python is running in the local directory (or specific folder). The easiest way to check this is to run the entire script in Spyder, by clicking the large “Run File” or play icon button at the top. If you do not receive a line error, then you have successfully run the code and created your groups in under 5 seconds with the preset category weights.

6. **If your input file names are not identical to those in step 4, then enter these into the GroupFormation.py file.** While the .py file and csv files are not strictly required to be in the same directory or folder to run the code, the code is written with the assumption that they are. If your files are in different folders, then the full file path needs to be manually entered into the code.

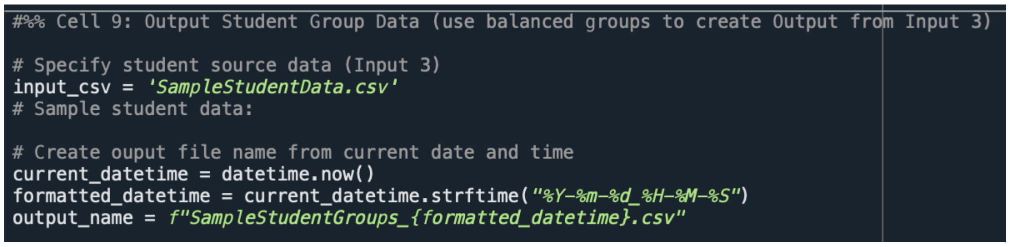
   The first two inputs, mentee data and student data subset, are in cell 3 (Fig. 2) and the full student data is in cell 9 (Fig. 3). The code is defaulted to the file names shown in Step 4 above. The desired name for the output file should also be entered into cell 9, the default name is "SampleStudentGroups_" appended with time the script was run as "year-month-day_hour-minute-second" ending with .csv as required (Fig. 2).



```
#%% Cell 3: Data Import

# Load the CSV file for mentor pairs (Input 1)
df_m = pd.read_csv('SampleMentorData.csv')
# df_m is a DataFrame with raw mentor pairing data

# Load the CSV file for student grouping data (Input 2)
df_s = pd.read_csv('SampleStudentDataSubset.csv')
# df_s is a DataFrame with raw student data grouping data

# # The following is for testing/debugging importing grouping data
# # Sample mentor data:
# # Sample student data:
# # Display the first few rows to understand the structure
# print(df_m.head())
# print(df_s.head())
```

Figure 2. The .py code showing where to enter file names for Input 1 and Input 2 into Cell 3.

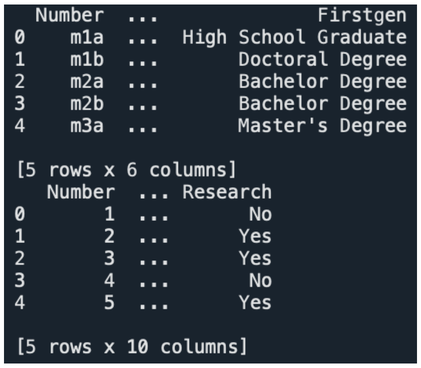
7. **Check for python package dependencies.** Run the first cell of the code, or copy the first cell to run in a standalone python script, to check for the installation of required packages. Any missing packages can be installed via pip.



```
#%% Cell 9: Output Student Group Data (use balanced groups to create Output from Input 3)

# Specify student source data (Input 3)
input_csv = 'SampleStudentData.csv'
# Sample student data:

# Create ouput file name from current date and time
current_datetime = datetime.now()
formatted_datetime = current_datetime.strftime("%Y-%m-%d_%H-%M-%S")
output_name = f"SampleStudentGroups_{formatted_datetime}.csv"
```

Figure 3. The .py code showing where to enter file names for Input 3 and Output into Cell 9.

8. **Check data imports.** Run the script up through the third cell with the debugging code uncommented. The terminal should display a sampling of the import data. An example of the import debugging code after successfully importing the sample data is shown here (Fig. 4).

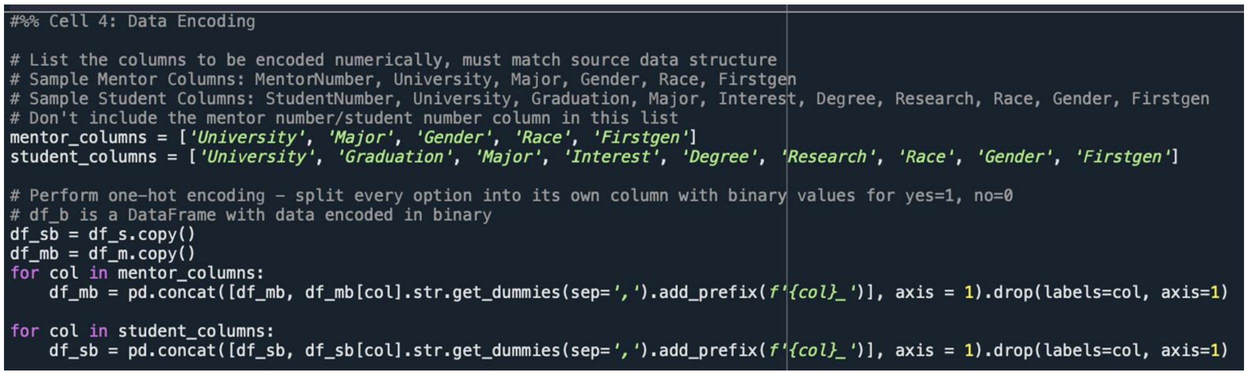


```
  Number  ...              Firstgen
0    m1a  ...  High School Graduate
1    m1b  ...       Doctoral Degree
2    m2a  ...       Bachelor Degree
3    m2b  ...       Bachelor Degree
4    m3a  ...       Master's Degree

[5 rows x 6 columns]
   Number  ... Research
0       1  ...       No
1       2  ...      Yes
2       3  ...      Yes
3       4  ...       No
4       5  ...      Yes

[5 rows x 10 columns]
```

Figure 4. The terminal output for Cell 3 of the .py code successfully importing the sample data.

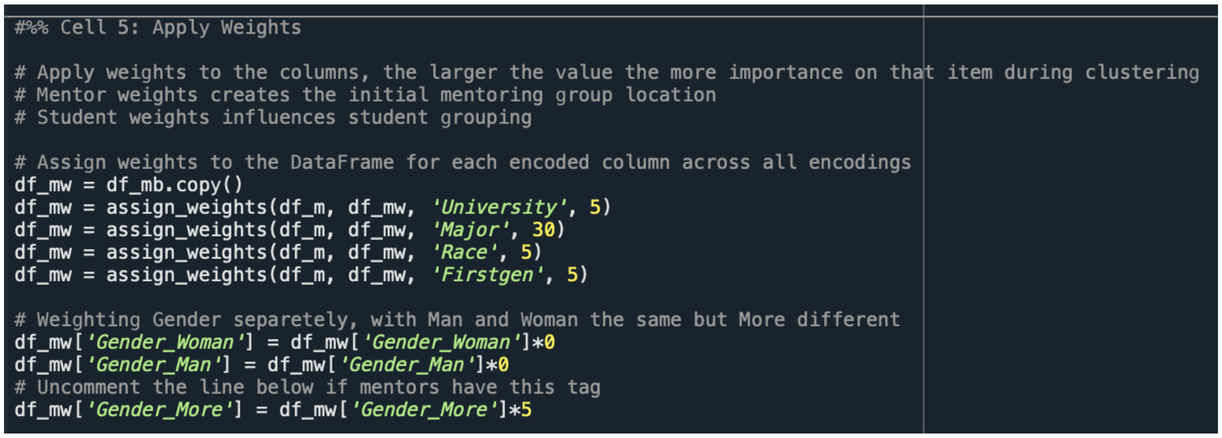
9. **Encode data and apply weights.** Run the fourth and fifth cells, no output is expected.
   a. In the fourth cell the column names for the mentor data (input 1) and student data subset (input 2) need to be manually entered as lists of strings. The sample code contains lists of the sample data column names which can be easily modified (Fig. 5).

b. In the fifth cell the desired weighting for each category is applied. These weights must be entered manually (Fig. 6).



```
#%% Cell 4: Data Encoding

# List the columns to be encoded numerically, must match source data structure
# Sample Mentor Columns: MentorNumber, University, Major, Gender, Race, Firstgen
# Sample Student Columns: StudentNumber, University, Graduation, Major, Interest, Degree, Research, Race, Gender, Firstgen
# Don't include the mentor number/student number column in this list
mentor_columns = ['University', 'Major', 'Gender', 'Race', 'Firstgen']
student_columns = ['University', 'Graduation', 'Major', 'Interest', 'Degree', 'Research', 'Race', 'Gender', 'Firstgen']

# Perform one-hot encoding - split every option into its own column with binary values for yes=1, no=0
# df_b is a DataFrame with data encoded in binary
df_sb = df_s.copy()
df_mb = df_m.copy()
for col in mentor_columns:
    df_mb = pd.concat([df_mb, df_mb[col].str.get_dummies(sep=',').add_prefix(f'{col}_')], axis = 1).drop(labels=col, axis=1)

for col in student_columns:
    df_sb = pd.concat([df_sb, df_sb[col].str.get_dummies(sep=',').add_prefix(f'{col}_')], axis = 1).drop(labels=col, axis=1)
```

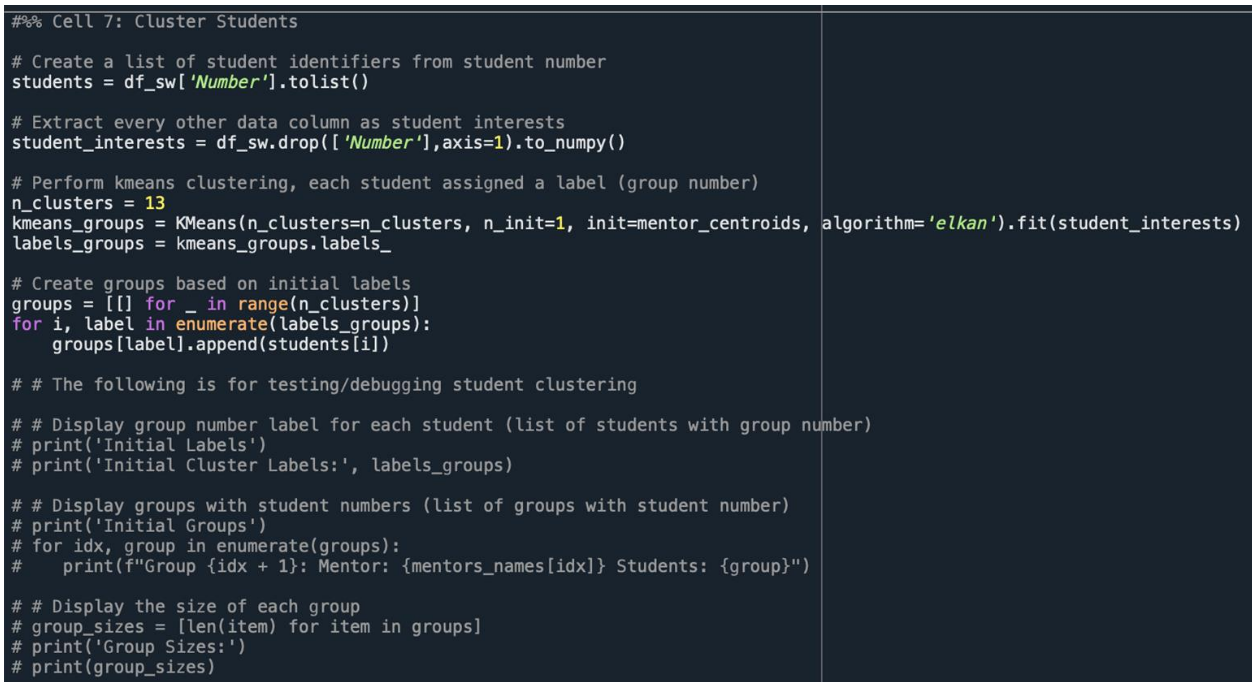
Figure 5. The .py code showing where to enter category names into Cell 4.



```
#%% Cell 5: Apply Weights

# Apply weights to the columns, the larger the value the more importance on that item during clustering
# Mentor weights creates the initial mentoring group location
# Student weights influences student grouping

# Assign weights to the DataFrame for each encoded column across all encodings
df_mw = df_mb.copy()
df_mw = assign_weights(df_m, df_mw, 'University', 5)
df_mw = assign_weights(df_m, df_mw, 'Major', 30)
df_mw = assign_weights(df_m, df_mw, 'Race', 5)
df_mw = assign_weights(df_m, df_mw, 'Firstgen', 5)

# Weighting Gender seperately, with Man and Woman the same but More different
df_mw['Gender_Woman'] = df_mw['Gender_Woman']*0
df_mw['Gender_Man'] = df_mw['Gender_Man']*0
# Uncomment the line below if mentors have this tag
df_mw['Gender_More'] = df_mw['Gender_More']*5
```

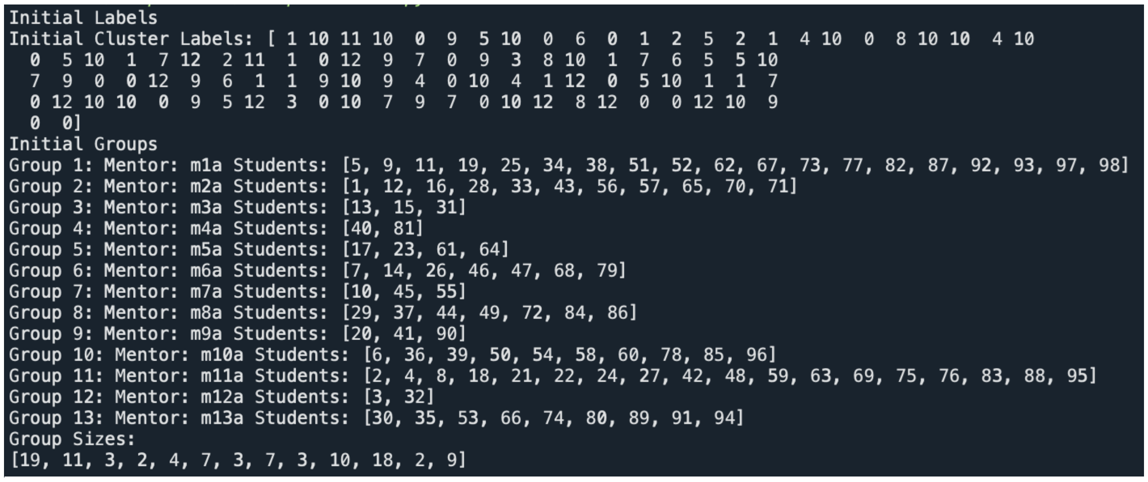
Figure 6. The .py code showing where to enter category weights into Cell 5.

10. **Update the number of mentoring groups that you require in the code and then cluster students.** Run the sixth and seventh cells with the debugging code in cell seven uncommented. The number of mentor pairs must be manually entered in cell seven as the number of clusters to be formed. The target size for each group is automatically calculated from the number of mentees and the number of groups. The terminal should display the initial cluster labels, initial mentor groups, and initial group sizes. The "Initial Cluster Labels" is a list of the group number initially assigned to each student (starting at 0, not 1). The "Initial Groups" is a list of each of each group, the first mentor of each pair for that group, and the students (identified by number) in that group. The "Group Sizes" is a list of the number of mentees within each group. An example of the debugging code after successfully clustering the sample data is shown here (Fig. 7). As can be seen the initial groups are very imbalanced (Fig. 8).



```
#%% Cell 7: Cluster Students

# Create a list of student identifiers from student number
students = df_sw['Number'].tolist()

# Extract every other data column as student interests
student_interests = df_sw.drop(['Number'],axis=1).to_numpy()

# Perform kmeans clustering, each student assigned a label (group number)
n_clusters = 13
kmeans_groups = KMeans(n_clusters=n_clusters, n_init=1, init=mentor_centroids, algorithm='elkan').fit(student_interests)
labels_groups = kmeans_groups.labels_

# Create groups based on initial labels
groups = [[] for _ in range(n_clusters)]
for i, label in enumerate(labels_groups):
    groups[label].append(students[i])

# # The following is for testing/debugging student clustering

# # Display group number label for each student (list of students with group number)
# print('Initial Labels')
# print('Initial Cluster Labels:', labels_groups)

# # Display groups with student numbers (list of groups with student number)
# print('Initial Groups')
# for idx, group in enumerate(groups):
#     print(f"Group {idx + 1}: Mentor: {mentors_names[idx]} Students: {group}")

# # Display the size of each group
# group_sizes = [len(item) for item in groups]
# print('Group Sizes:')
# print(group_sizes)
```

Figure 7. The .py code showing where to enter the number of mentor pairs into Cell 7.



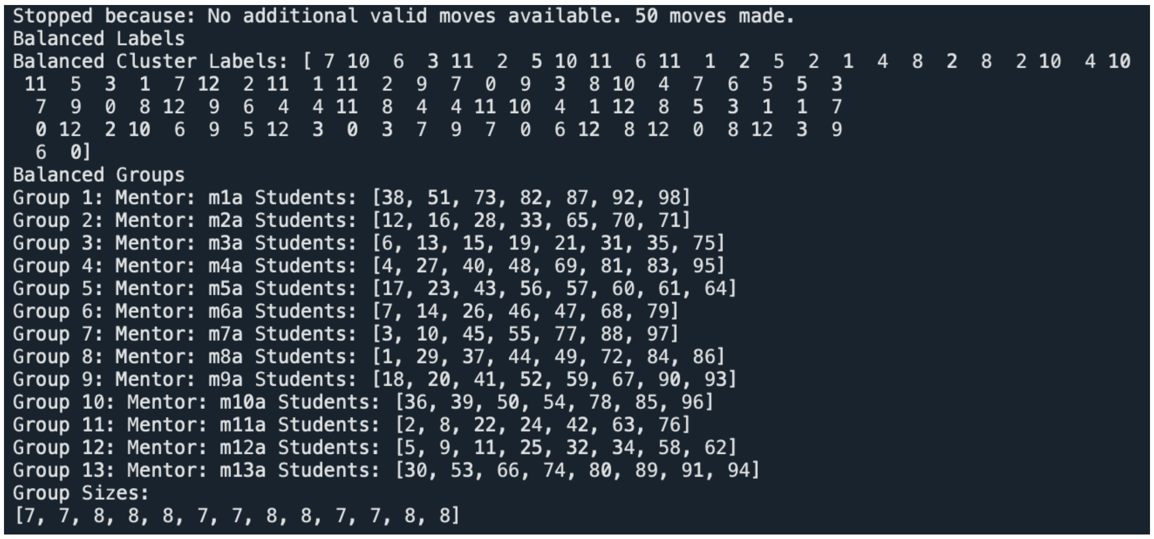
```
Initial Labels
Initial Cluster Labels: [ 1 10 11 10  0  9  5 10  0  6  0  1  2  5  2  1  4 10  0  8 10 10  4 10
  0  5 10  1  7 12  2 11  1  0 12  9  7  0  9  3  8 10  1  7  6  5  5 10
  7  9  0  0 12  9  6  1  1  9 10  9  4  0 10  4  1 12  0  5 10  1  1  7
  0 12 10 10  0  9  5 12  3  0 10  7  9  7  0 10 12  8 12  0  0 12 10  9
  0  0]
Initial Groups
Group 1: Mentor: m1a Students: [5, 9, 11, 19, 25, 34, 38, 51, 52, 62, 67, 73, 77, 82, 87, 92, 93, 97, 98]
Group 2: Mentor: m2a Students: [1, 12, 16, 28, 33, 43, 56, 57, 65, 70, 71]
Group 3: Mentor: m3a Students: [13, 15, 31]
Group 4: Mentor: m4a Students: [40, 81]
Group 5: Mentor: m5a Students: [17, 23, 61, 64]
Group 6: Mentor: m6a Students: [7, 14, 26, 46, 47, 68, 79]
Group 7: Mentor: m7a Students: [10, 45, 55]
Group 8: Mentor: m8a Students: [29, 37, 44, 49, 72, 84, 86]
Group 9: Mentor: m9a Students: [20, 41, 90]
Group 10: Mentor: m10a Students: [6, 36, 39, 50, 54, 58, 60, 78, 85, 96]
Group 11: Mentor: m11a Students: [2, 4, 8, 18, 21, 22, 24, 27, 42, 48, 59, 63, 69, 75, 76, 83, 88, 95]
Group 12: Mentor: m12a Students: [3, 32]
Group 13: Mentor: m13a Students: [30, 35, 53, 66, 74, 80, 89, 91, 94]
Group Sizes:
[19, 11, 3, 2, 4, 7, 3, 7, 3, 10, 18, 2, 9]
```

Figure 8. The terminal output for Cell 7 of the .py code after successfully creating initial groups from the sample data. The “Initial Cluster Labels” is a list of the group number initially assigned to each student (starting at 0, not 1). The “Initial Groups” is a list of each of each group, the first mentor of each pair for that group, and the students (identified by number) in that group. The “Group Sizes” is a list of the number of mentees within each group. Initial groups are very imbalanced.

11. **Balance groups.** Run the eighth cell with the debugging code uncommented. The terminal should display why the balancing code stopped, how many moves were made, the balanced cluster labels, balanced mentor groups, and balanced group sizes. An example of the debugging code after successfully balancing the sample data is shown here (Fig. 9). The

output indicates that 50 moves were made to better balance the number of students in each group and the code stopped because no more valid moves were available. As can be seen, the groups are as balanced as numerically as possible with a group size of either 7 or 8 students.

12. **Create the output csv file.** Run the ninth and final cell. The terminal should display "Student groups output csv file generated successfully!" The output file will be located in the same directory as the .py file with whatever name was entered in cell 9.



```
Stopped because: No additional valid moves available. 50 moves made.
Balanced Labels
Balanced Cluster Labels: [ 7 10  6  3 11  2  5 10 11  6 11  1  2  5  2  1  4  8  2  8  2 10  4 10
 11  5  3  1  7 12  2 11  1 11  2  9  7  0  9  3  8 10  4  7  6  5  5  3
  7  9  0  8 12  9  6  4  4 11  8  4  4 11 10  4  1 12  8  5  3  1  1  7
  0 12  2 10  6  9  5 12  3  0  3  7  9  7  0  6 12  8 12  0  8 12  3  9
  6  0]
Balanced Groups
Group 1: Mentor: m1a Students: [38, 51, 73, 82, 87, 92, 98]
Group 2: Mentor: m2a Students: [12, 16, 28, 33, 65, 70, 71]
Group 3: Mentor: m3a Students: [6, 13, 15, 19, 21, 31, 35, 75]
Group 4: Mentor: m4a Students: [4, 27, 40, 48, 69, 81, 83, 95]
Group 5: Mentor: m5a Students: [17, 23, 43, 56, 57, 60, 61, 64]
Group 6: Mentor: m6a Students: [7, 14, 26, 46, 47, 68, 79]
Group 7: Mentor: m7a Students: [3, 10, 45, 55, 77, 88, 97]
Group 8: Mentor: m8a Students: [1, 29, 37, 44, 49, 72, 84, 86]
Group 9: Mentor: m9a Students: [18, 20, 41, 52, 59, 67, 90, 93]
Group 10: Mentor: m10a Students: [36, 39, 50, 54, 78, 85, 96]
Group 11: Mentor: m11a Students: [2, 8, 22, 24, 42, 63, 76]
Group 12: Mentor: m12a Students: [5, 9, 11, 25, 32, 34, 58, 62]
Group 13: Mentor: m13a Students: [30, 53, 66, 74, 80, 89, 91, 94]
Group Sizes:
[7, 7, 8, 8, 8, 7, 7, 8, 8, 7, 7, 8, 8]
```

Figure 9. The terminal output for Cell 8 of the .py code after successfully balancing groups formed from the sample data. The output indicates that 50 moves were made and the code stopped because no more valid moves were available. The "Balanced Cluster Labels" is a list of the group number assigned to each student (starting at 0, not 1). The "Balanced Groups" is a list of each of each group, the first mentor of each pair for that group, and the students (identified by number) in that group. The "Group Sizes" is a list of the number of mentees within each group. Groups are as balanced as numerically as possible.

RQ2: How much faster is the algorithm method compared to the manual method?

To determine how much time this algorithm saves for developing mentoring circles, we timed how long it took for a person not familiar with the code to prepare and run the protocol. We then compared this with the time reported for manual group formation. The protocol required approximately 40 minutes, and involved two phases: document preparation (20 minutes) and updating code and running the algorithm (20 minutes). This time was measured for an individual not familiar with python starting with a new dataset that was not previously encoded for this algorithm. Given the manual process took a reported 4-8 hours, this demonstrates that using the algorithm and new protocol is 6-12 times faster than the manual process of matching mentors and mentees.

RQ3: What are the benefits and limitations of using a computer-based and automated method to assign mentees to mentoring circles?

Benefits of algorithm or computer aided matching for mentoring circles: There are several significant advantages to using an algorithm to determine groups of student mentees with mentors (Table 1). First, algorithmic matching accelerated the group formation process, allowing for fast creation of groups and making the process easier and more streamlined compared to manual methods. Second, another benefit includes the reduction of human bias during the matching process. Using an algorithm can potentially lead to more fair grouping and increase the backgrounds of perspectives within each mentoring circle. Finally, the automated system demonstrates the ability to scale this process and continue to grow the mentoring program in the future, suggesting it is helpful in automating administrative tasks to handle increases in mentor-mentee relationships. Not only does this algorithm match students, but the export function automatically compiles and organizes all student information after circle creation, eliminating the need for manual movement of people on spreadsheets.

Some more nuanced benefits also include the benefit of enhanced efficient matching, as algorithms can analyze more data to identify patterns and correlations between mentors and within mentees, which can lead to more accurate and effective matching. With 98 students and 13 groups, there could be over 900 novemvigintillion (900×10^{90}) potential ways to combine mentees for mentoring circles. It thereby enhances the potential for amplifying intersectional relationships in mentoring circles, and not grouping students based on one specific characteristic. The algorithm can also be tweaked to ensure consistent groupings by number of mentees, allowing for an even distribution of mentees across mentoring circles. An algorithm can thus increase the ability to fine-tune the creation of circles by setting different category weights, allowing for administrators to develop different circles depending on different contexts.

Table 1. Benefits and Limitations to computer-aided matching for mentoring circles

Benefits	**Limitations**
● Fast creation of groups ● Easy ● Streamlined ● Reduce Bias ● Efficient Matching ● Automated ● Scalable ● Personalized weights ● Exporting functions	● Lack of human touch ● Can amplify existing bias ● Privacy concerns ● Lack of transparency ● Not helpful for pairing mentors ● Can only create homogeneous groups, no option to add heterogeneity

Overall, the implementation of an algorithm-aided matching for mentoring circles has resulted in improvements in this process, including efficiency, reduction in bias, matching accuracy and administrative ease. This process has now evolved from a multi-hour matching process to something that can be completed in less than 40 minutes. All these improvements enhance the effectiveness of the mentoring program.

Limitations of algorithm or computer aided matching for mentoring circles: While algorithm-aided matching for mentoring circles offer many benefits as previously described, there are also limitations that should be considered (Table 1). One drawback is the lack of human touch in the process. Algorithms might miss nuances in the mentee and mentor application statements and essays that a human administrator might capture. These subtle aspects of interests and personality may be overlooked by automated systems. This limitation is closely linked to the lack of transparency that exists in algorithmic decision-making, which can make it difficult for administrators to understand how groups are formed without troubleshooting.

Another concern is the potential for amplifying existing biases. If the algorithm is not carefully designed, then it could reinforce stereotypes or limit diversity [25]. This issue is particularly relevant when considering the balance between creating homogeneous groups (amplifying similarities) and heterogeneous groups (dispersing similarities). Groups with too many similarities might not learn and grow from one another, limiting the number of perspectives that are helpful for effective mentoring.

While computer-aided matching can be incredibly helpful, it is possible that some manual rearrangement will still be necessary to optimize group characteristics. This manual process at the end would then reduce the efficiency gained by the automated process. Further, the current algorithm is not helpful for pairing up mentors, as the goal is to have differences rather than similarities between mentors, especially in smaller programs with 24-30 mentors where having mentors from different backgrounds and majors is important. Existing clustering algorithms create clusters based on similarities and thus aren’t optimized for creating heterogeneity.

Finally, privacy concerns exist when using algorithms for matching, as they require collecting and analyzing personal information. Measures can be taken to protect privacy, such as excluding names and coding students prior to input, but privacy is still an important concern. Overall, while algorithm-aided matching offers significant advantages, limitations exist and human oversight is important. Therefore, the current hybrid approach – combining the efficiency of the algorithm with human oversight – is the preferred method of creating mentoring circle groups.

Conclusions

Here, we have shown that our algorithm can successfully cluster students into mentoring circles, and have described the key benefits and limitations of this system. The development of this algorithm is significant, as it develops one of the first resources to center the formation of these mentee groups on mentor characteristics through the use of k-means clustering. With 98 students and 13 groups, there could be over 900 novemvigintillion (900×10^{90}) potential ways to combine mentees in mentoring circles, and we have developed a fast and objective method of making these mentoring groups. The protocol that we developed takes approximately 40 minutes, and it reduces the administrative time required for mentoring circle group formation, making the process six to twelve times faster. The algorithm and sample data sets are available to download and use for free on Github (github.com/tompkinn/mentoring-circle-group-formation).

While we have used this algorithm for the development of mentoring circles for the GradTrack Scholars Program, this code could be used and adapted for many other mentoring circle applications – such as mentoring circles for peer-mentoring of first year graduate students, graduate students and postdocs interested in postdoctoral scholar positions and community [3], future faculty careers [4], as well as any other mentoring circle applications. The code's categorization system can be directly adjusted to align with desired grouping criteria, enhancing its flexibility and analytical precision.

Future work should compare the composition of the groups formed by this method with the groups formed manually. However, if this comparison were to be completed then one would also want to determine the effectiveness of each mode of grouping, which would likely require parallel study and programming. In this case, it would be important to implement peer evaluations, similar to the CATME platform for team evaluations [20] - [22], or perform pre- and post- program surveys to understand if specific group formations altered preparation for graduate school.

In developing our algorithm, we encountered some challenges, particularly regarding the clustering approach. We initially considered a two-part process for the algorithm: automating first the clustering graduate student mentors, followed by clustering undergraduate student mentees. However, this method proved ineffective, as clustering mentors failed to achieve the desired distribution and representation of mentors across circles. Recognizing these limitations, we ultimately opted for a hybrid approach: manually clustering mentors to ensure representation and the presence of differences between the mentors, followed by use of the k-means clustering algorithm to directly cluster mentees based on their similarities with each other and with the mentors. This revised method allowed us to better address the specific grouping criteria we sought to implement, overcoming the initial obstacles we faced.

Overall, development of this algorithm has supplied administrators with a more efficient process for creating mentoring groups with a reduction in human time and subjectivity, which will support the continued growth of the GradTrack Scholars program, and other mentoring circle programs.

References

[1] L. C. Arinze, J. M. Beagle, and J. E. McDermott, "Assessing the Effectiveness of the GradTrack Virtual Mentoring Program," Paper presented at the *2023 ASEE Annual Conference & Exposition, Baltimore, Maryland, USA, June 25-28, 2023*. 10.18260/1-2—42681. Available: https://peer.asee.org/42681

[2] M. A. Fridkis-Hareli, "A mentoring program for women scientists meets a pressing need," *Nat Biotechnol*, 29, 287–288, 2011

[3] C. Kuhn and Z. Castaño, "Boosting the career development of postdocs with a peer-to-peer mentor circles program," *Nat Biotechnol*, 12;34(7):781-3, Jul 2016.

[4] M. Broberg, B. Bose, R. Pineda-Mendez, D. Devine, R. Gehr, C. G. Jange, J. McDermott, M. Loui, and J. Eisma, "Lessons learned - preparing graduate students and postdoctoral researchers for tenure track careers through mentoring circles," Paper presented at the *2022 ASEE Annual Conference & Exposition, Minneapolis, MN, USA, June 26-29, 2022*. 10.18260/1-2—40662. Available: https://peer.asee.org/40662.

[5] R. A. Layton, M. L. Loughry, M. W. Ohland, and G.D. Ricco. (2010). Design and validation of a web-based system for assigning members to teams using instructor-specified criteria. *Advances in Engineering Education*, 2 (1), 1-28.

[6] J. McDermott, J. Beagle, "GradTrack Scholars: A comprehensive online mentoring program to build community and prepare the next generation of underrepresented minority graduate students (Work in Progress)," Paper presented *2022 ASEE Annual Conference & Exposition, Minneapolis, MN, USA, June 26-29, 2022*. 10.18260/1-2—40778. Available: https://peer.asee.org/40778.

[7] C. Deng, D. B. Gulseren, and N. Turner. (2022) "How to match mentors and protégés for successful mentorship programs: a review of the evidence and recommendations for practitioners", *Leadership & Organization Development Journal*, Vol. 43 No. 3, pp. 386-403

[8] D. A. Harrison, K. H. Price, and M. P. Bell. (1998) "Beyond relational demography: time and the effects of surface- and deep-level diversity on work group cohesion", *Academy of Management Journal*, Vol. 41 No. 1, pp. 96-107, doi: 10.5465/256901.

[9] P. R. Hernandez, M. Estrada, A Woodcock, and P.W. Schultz. (2017) "Mentor qualities that matter: The importance of perceived (not demographic) similarity." J Exp Educ. 85(3):450-468.

[10] T. T. Tuma and E. L. Dolan.(2024) "What makes a good match? Predictors of quality mentorship among doctoral students." *CBE—Life Sciences Education*, 23(2), ar20.

[11] T. C. Dennehy, and N. Dasgupta. (2017) "Female peer mentors early in college increase women's positive academic experiences and retention in engineering." *Proceedings of the National Academy of Sciences*, 114(23), 5964-5969.

[12] J. B. Main, L. Tan, M. F. Cox, E. O. McGee, and A. Katz. (2020) "The correlation between undergraduate student diversity and the representation of women of color faculty in engineering." *Journal of Engineering Education*, 109(4), 843-864.

[13] J. M. Valla, and W. Williams. (2012) "Increasing achievement and higher-education representation of under-represented groups in science, technology, engineering, and mathematics fields: A review of current K-12 intervention programs." *Journal of women and minorities in science and engineering*, 18(1).

[14] J. R. Posselt. *Equity in science: Representation, culture, and the dynamics of change in graduate education*. Stanford University Press, 2020.

[15] B. Oakley, R. M. Felder, R. Brent, and I. Elhajj. (2004). Turning student groups into effective teams. *Journal of Student Centered Learning*, 2 (1), 9–34.

[16] D. R. Bacon, K. A. Stewart, and W. S Silver. (1999). Lessons from the best and worst student team experiences: How a teacher can make the difference. *Journal of Management Education*, *23*(5), 467-488.

[17] R. Decker, (1995). Management team formation for large scale simulations. *Developments in Business Simulation and Experiential Learning: Proceedings of the Annual ABSEL conference* (Vol. 22).

[18] M. L. Loughry, M. W. Ohland, and D. J. Woehr, (2014). Assessing teamwork skills for assurance of learning using CATME Team Tools. Journal of Marketing Education, 36(1), 5-19.

[19] D. R. Bacon, K. A. Stewart, and E. S. Anderson. (2001). Methods of assigning players to teams: A review and novel approach. *Simulation & Gaming*, *32*(1), 6-17.

[20] A. C. Loignon, D. J. Woehr, J. S. Thomas, M. L. Loughry, M. W. Ohland, and D. Ferguson. (2017). Facilitating Peer Evaluation in Team Contexts: The Impact of Frame-of-Reference Rater Training. Academy of Management Learning & Education, 16(4), 562-578.

[21] M. W. Ohland, M.L. Loughry, D.J. Woehr, L.G. Bullard, R.M. Felder, C.J. Finelli, R.A. Layton, H.R. Pomeranz, and D.G. Schmucker. (2012). The comprehensive assessment of team member effectiveness: Development of a behaviorally anchored rating scale for self and peer evaluation. Academy of Management Learning & Education, 11 (4), 609-630.

[22] M. L. Loughry, M.W. Ohland, and D.D. Moore. (2007). Development of a theory-based assessment of team member effectiveness. Educational and Psychological Measurement, 67, 505-524.

[23] S. Akbar.,E. F. Gehringer, and Z. Hu. (2018) "Improving formation of student teams: a clustering approach." *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings* (pp. 147-148).

[24] M. I. Victor-Ikoh, and D. C. Ogunmodimu. (2021). "Students' Group Formation Using K-Means Clustering in Combination with a Heterogeneous Grouping Algorithm." *American Journal of Engineering Research*, *10*(11), 1-9.

[25] L. Rainie, J. Anderson, "Code-Dependent: Pros and Cons of the Algorithm Age. Pew Research Center, February 2017. Available at: http://www.pewinternet.org/2017/02/08/code-dependent-pros-and-cons-of-the-algorithm-age