FirstWork Recommendation Algorithm - 16 May 2023

1. Algorithm Overview:

- Purpose: Develop an algorithm that takes a student's lesson progress based on how many questions and what kind of questions they have completed successfully, and generate an optimal and suitable lesson plan for the child.
- Programming Language: Python
- Dependencies/Imports: Python Pandas, Scikit-learn, Numpy, Matplotlib, Scipy, math, os

2. Why Unsupervised Learning?:

- Agglomerative Hierarchical Clustering: allows us to analyze various types of data via clustering techniques
    - Individual data points merge into well-defined groups via analysis
- Explains what combinations of learning are best for an *individual* student (unique to each student)
    - Erases the problem regarding initial assumptions for a child's learning
- Disadvantage to watch out for in the future: may need a lot of user data to make more accurate recommendations, so they could be a bit unrepresentative of a student initially

3. Algorithm Description:

- High-Level Explanation: This algorithm takes student data (in the form of completed questions) and turns it into recommendations to be outputted to a parent creating lesson plans for their child. It does so by looking at a csv, cleaning it up, sorting the data into clusters, calculating the means for each cluster, and comparing the clusters in terms of correctness to identify the one most suitable for the child's learning
- Input: String. This string represents the directory the datafile to be analyzed is in. This datafile should be a csv containing student-completed questions with attributes:
    - Date, Time of Day (rounded 24 hour scale), Associated Lesson Entertainment Time (minutes), Number of Attempts, Shuffled, Questions per Lesson, Lesson per Session
- Output: Tuple of Strings. This tuple contains five recommendations for parents to consider when making lesson plans for their child: Time of Day, Reward Time, Shuffled, Questions per Lesson, and Lessons per Session

4. Algorithm Workflow:

- Step 1: The program reads in a csv file containing the data points of a student
- Step 2: The data is then loaded into a data frame and is scaled and clustered
- Step 3: To determine the amount of clusters, we used the silhouette method which is defined in **determineClusters**
- Step 4: After the data has been clustered, a new column stores the cluster label to each corresponding row depending on what cluster they were grouped into
- Step 5: The clusters are then separated into their own data frame and stored in a dictionary where the key is the cluster label

- Step 6: The dictionary of data frames is then given to **dict_of_cluster_means** where the functions calculate the means of each column for each data frame and stores a tuple of the means for each column into the dictionary where the key is the cluster label for each corresponding data frame
- Step 7: This data is then given to **compare_clusters**, which looks for the cluster with the highest correctness average and that contains the most data points in the original cluster
- The highest correctness tuple is then returned and given to **descaleData** and is descaled before being outputted

5. Implementation Details:

- Code Structure:
    - **processData:**
        - This function takes in a csv file and loads it into a data frame. The data is then scaled and a new column is created… based on 1/**number of attempts**
    - **determineClusters:**
        - Runs the silhouette method which determines the number of clusters based off of the students data points… the more clusters the more variation between data points. The suggested number of clusters is returned.
    - **hierarchicalClustering:**
        - This function clusters the points and creates a new column that contains the corresponding cluster labels for each row. The data frame is then returned with the number of clusters.
    - **Compare_clusters:**
        - This function iterates through each cluster in the dictionary and compares the mean correctness values. If the values are equal, it further compares the sizes of the clusters. If a cluster is significantly larger it is selected. Otherwise, if the mean correctness value is higher than it is selected as the best cluster. Returns a tuple with cluster label and the recommendation
- **Implementation Tools:** Visual Studio Code, Github
- **Variables and Data Structures:** used dictionaries to store the data, where the key is the cluster labels and the values is the data frame of the corresponding cluster.
- Used various helper functions and modules are used for data preprocessing, clustering and analysis

6. Other notes/suggestions:

- Currently there is a method to make histograms of the data, along with some extra code to work with “visualizations” in the folder named as such. This is only for visualizations of the data and is not necessary to get the required output

7. Contact Information:

- Andrea Ramirez: andrearamirez@sandiego.edu

- Christian Gideon: christiangideon@sandiego.edu
- Dianne Catapang: dcatapang@sandiego.edu
- Dillan Lopez: dillanlopez@sandiego.edu