Alexander Harry
Justin Keeling
John Lambrecht
Andrew Smith

Design Process: K-NN

Our design consists of four different classes:

1. Main()
    a. The Main() class consists of three variables of which are instances of our three other classes. The only method in the Main class is main() which calls the methods in the other classes that are necessary for our program to operate..
2. Data()
    a. The Data() classes primary purpose is to process and prepare data for the algorithm to properly run. It consists of three DataFrame variables that contain a data dictionary, our training data, and our testing data. This class consists of methods to load, split, process, and prepare the data for machine learning algorithms. It also consists of a method to process the data to perform a k-fold operation.
3. KMeans()
    a. The KMeans class takes in a user defined k-value in which it specifies the medoids and centroids. Given the distances projected by calculating the Euclidian distances from the K-NN class, we are then able to calculate the error with the loss function method, for regression data.
4. KNN()
    a. The KNN class consists of multiple nearest neighbor algorithms such as edited, condensed, and traditional K-NN. Considering these algorithms, we use Euclidian distance which is universally accessed by each algorithm.

Our initial thought as a group, is to create multiple classes in which machine learning algorithms, data, and running the program can be separated. Separating the classes with this reasoning allows functions to be spread-out and included in the class that best represents them.

One reason to our design is the fact that our main class uses function calls to other classes much simpler than having function calls intertwining between other classes. Since our main method controls the inputs and retrieves outputs from other class functions, it is easy to find possible errors, as well as incorporating other machine learning algorithms such as medoids/centroids.

Another choice we have considered, is the difference between the regression type data and classification. A question that needs more contemplation, is: How to organize between train data, test data, regression type data, and classification type data? Our current design is to use two dictionaries. One dictionary will encompass the test data, and the other; training data. Although this allows keys (names of the dataset) and values (the data sets) to be accessed from for loops, it currently seems like a simple approach. But should four dictionaries be used? Two for both regression and classification data? This will be to the group to decide, but the design to which data is stored and accessed may change.

As we continue to learn more about the algorithms, complexity, and code to which our program must have, the UML may differ from the current design. We plan to discuss these changes as we begin to see discrepancies within our current UML class diagram.

Alexander Harry
Justin Keeling
John Lambrecht
Andrew Smith