Medi-App: A Python Application for Self-Diagnosis of Illnesses

Ali Rizvi
Alwin Baby
Kelly Holtzman
Mansi Patel
Faculty of Engineering and Applied Science, University of Regina

ENSE 496AC: Artificial Intelligence

Dr. Kin-Choong Yow

December 4, 2020

Medi-App: A Python Application for Self-Diagnosis of Illnesses

Project Summary

Medi-App provides the symptom-to-illness investigation of online clinical databases without the time consumption of searching online. Symptoms and illness prognosis data are from online machine learning database Kaggle (Anonymous, 2020) under the Database Contents License v1.0, and are fit for our purpose of academic use. The data are used to train a Bayesian Network (Bayes's Net) and an Artificial Neural Network (ANN) with Multi-Layer Perceptron (MLP). The resulting probabilistic models provide Medi-App with the capability to compare user symptoms with past observed symptoms, and report the probability that the user is observing symptoms that indicate the presence of a particular illness or disease.

Problem Description

The outbreak of the disease COVID-19 has drawn the common person to reflect on symptoms of illness more frequently. Symptoms similar to the coronavirus are being investigated more on search engines as a result (Google Trends, n.d.). The medical world is vast, and the average person may find themselves down a rabbit hole of symptoms and illnesses that may not even relate to what they were investigating in the first place. To prevent the potential situation where the user ends up convincing themselves they have an unlikely illness, our team created an application for self-diagnosis of illnesses that would provide legitimate suggestions for the mystery illness without the user needing to research alone.

Project Objectives

We identified objectives for the project based on the course project description handed out at the beginning of the semester: we first define a set of personal project objectives that we determined would drive forwards our understanding and experience with artificial intelligence; then we defined objectives we would appreciate our application reach from the programmer's perspective:

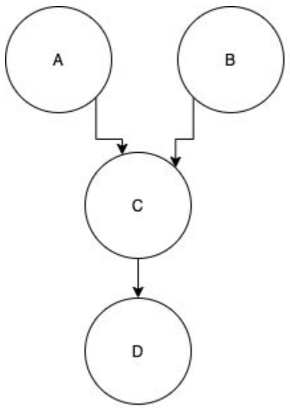
1. **Personal objective.** Our team wanted to use our knowledge from our Artificial Intelligence post-secondary course to build an application that used the algorithms we studied and also demonstrates our skills as software engineering students to apply what we learned into a useful application for the public.
2. **Personal objective.** We wanted the application to make use of common python packages that we could find ourselves using in industrial applications, and that these same packages were also commonly used in the software artificial intelligence community.

3. **Application objective.** We wanted the application to be as truthful as possible given available data and academic model accuracy: the application should reliably predict illnesses from the given data and user input, and should not give random results for the same input if the model were not fit to new data between tests.
4. **Application objective.** Finally, we want the application to be desirable to use over manual investigation of illnesses for the average, non-software or medical person: the application should not take more time than manual investigation of the same illnesses using arbitrary means not including the application or its underlying concepts.

Approaches Investigated

Bayesian Network (Kelly Holtzman)

A Bayes's Net is a probabilistic directed acyclic graph, made from sets of marginal and conditional probability distributions, which relate the nodes in the graph by their conditional dependence (or independence in the absence of an edge between nodes). In a Bayes's Net, we describe the conditional probability of each node as a weighted combination of it's parent's probability value (UC Berkeley, 2014), using the formula $P(X|a_{initial}\ldots a_{final})$. For a set of observed nodes, the probability that the observed set occurred is given by the product of the probability value of each node with the equation $P(X_1, X_2, .., X_n) = \prod_{i=1}^{n} P(X_i|a_{initial}...a_{final})$ (UC Berkeley, 2014). We can present the above equation with a graphical example:



Probability Distributions of root nodes	
+A	0.30
-A	0.70
+B	0.15
-B	0.85

Probability Distributions of non-root, non-leaf nodes			
+A	+B	+C	0.20
+A	+B	-C	0.80
+A	-B	+C	0.45
+A	-B	-C	0.55
-A	+B	+C	0.40
-A	+B	-C	0.60
-A	-B	+C	0.90
-A	-B	-C	0.10

Probability Distributions of leaf node		
+C	+D	0.50
+C	-D	0.50
-C	+D	0.75
-C	-D	0.25

Figure 1: Bayesian Network Graph Example

$$P(+A, +B, -C, +D) = P(+A)P(+B)P(-C|+A, +B)P(+D|-C) = 0.30x0.15x0.80x0.75 = 0.045$$

$$P(+A, -B, -C, +D) = P(+A)P(-B)P(-C|+A, -B)P(+D|-C) = 0.30x0.85x0.55x0.75 = 0.1051875$$

Figure 2: Bayesian Network Equation Example

For example, if we observed +A, -C, +D, then we can determine the probability that +B or -B occurred given +A, -C, +D, by the above calculations. Since P(+A, -B, -C, +D) > P(+A, +B, -C, +D), we can infer that it was mostly likely -B occurred.

Bayes's Nets are typically used to infer the presence of an unobserved node given a set of nodes that were observed, as seen in the above example (Schreiber, 2018). A real-life investigation of Bayes's Nets on existing legitimate medical data was executed by Auras, H., Merouani, H. F., & Refai, A. (2016) using Bayes's Nets as a way to support diagnosis of breast-related diseases. Auras, et. al (2016) suggest that the Bayes's Net model be retrained during use such that the probability distribution the algorithm executes on is accurate and up-to-date as possible (a hybrid method using past data and present decisions); in fact, the competence of the model and it's legitimacy in use as a medical support tool rely on the tool representing relevant data and evolving with the addition of new data. Furthermore, the study (Auras, et. al, 2016) suggests that additional maintenance be used to determine exact inference of the disease from reported symptoms; a method that would better predict the probability distribution in comparison to the Bayes's Net formulas presented above such that poor inferences become training data to move away from future poor inferences of the same type.

It is then evident from Auras, et. al (2016) and references to other papers in "Maintenance of a Bayesian network: application using medical diagnosis" that Bayes's Nets alone are not enough to accurately predict diseases - the model should be combined with forms of past data and expert opinion and a tool to maintain the probability distribution. Our implementation of a self-diagnosis application ought to be able to achieve a more accurate Bayes's Net model, compared to the basic algorithm described by UC Berkeley (2014), for predicting illnesses from symptoms if we retrain the network with new, expert input, and maintain the network's probability distributions from the results of first approximate inference. Our Bayes's Net could potentially reduce its probability distribution to an acceptable academic rate for prediction. Further benefits of a Bayes's Net implementation include the simplicity of it's graphical presentation for non-developers and non-statisticians, particularly as our application may be used by persons with an educational background that may not include software development or statistics. Further details on actually implementing the Bayes's Net are discussed under the Implementation Details section.

Artificial Neural Network (Mansi Patel)

An ANN is a computational model consisting of processing units called neurons, which are connected with associated weights (coefficients) with one another

(Shanmuganathan, 2016); It is also called the connectionist model because of the connections between neurons. When imprecise or complex attribute relationships are difficult to quantify, an ANN may be used in various types of application areas. The ANN is described by the following four parameters:

1. Type of Neuron - i.e. Perceptron or Fuzzy Neuron
2. Connectionist Architecture - defined based on the organization of the connectivity of neurons with next layer neuron; i.e. fully connected, partially connected
3. Learning Algorithm, which trains the networks
4. Recall Algorithm

The Connectionist Architecture is also distinguished based on the number of inputs and outputs. Layers are used as Autoassociative, Heteroassociative, Feedforward architecture, or Feedback architecture.

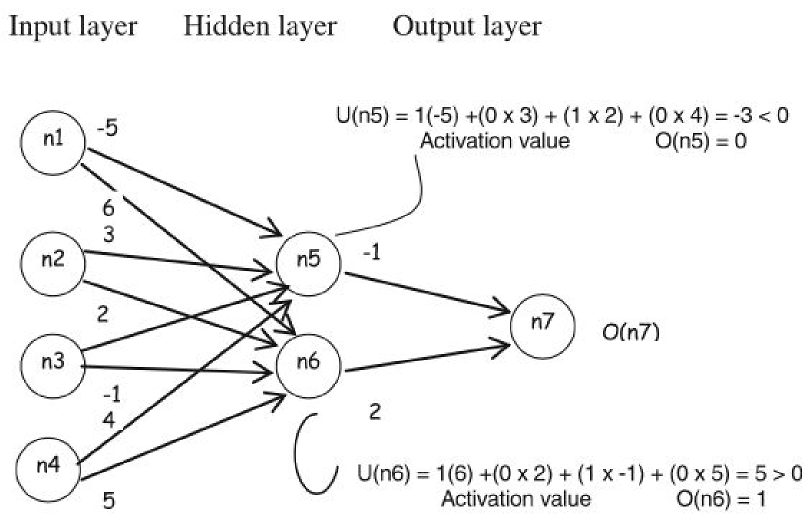

Input layer
Hidden layer
Output layer
n1
n2
n3
n4
n5
n6
n7
-5
6
3
2
-1
4
5
-1
2
U(n5) = 1(-5) +(0 x 3) + (1 x 2) + (0 x 4) = -3 < 0
Activation value O(n5) = 0
O(n7)
U(n6) = 1(6) +(0 x 2) + (1 x -1) + (0 x 5) = 5 > 0
Activation value O(n6) = 1

Figure 3: Simple Neural Network with four inputs (Shanmuganathan, 2016)

To calculate the output of given inputs we use the definition

$$\mathrm{U_j} = \sum \left(X_i . w_{ij}\right) \mathrm{for\, i} = (1, 2, \ldots, \mathrm{n}) \mathrm{for\, j} = (1, 2, \ldots, \mathrm{m})$$ (Shanmuganathan, 2016) :

Where, U_j = the net input signal to each output neuron j, X_0 = is the bais, X = input vector. To calculate the nonlinear transformation activation function y_j we use

$$y_j = f(a_{0j} + a_{1j}x) = \frac{1}{1 + e^{-(a_{0j} + a_{1j}x)}}$$.

According to the article “Artificial Neural Networks in Diagnosis of Liver Diseases” (2015), it is necessary to have prediction values (weights) of input neurons that should be accurately predicted. The neuron weights are needed in the training datasets. Further from the article “Artificial Neural Networks in Diagnosis of Liver Diseases” the main recommendations from using ANN to diagnose Liver Disease are:

1. Raw measurements are difficult to translate into interpretable and actionable data, and;
2. Data cannot be considered as information

Using ANN we may predict the disease based on the input symptom neurons. With weighted neuron symptoms, the output neuron can be activated. Our only concern would be to have proper weights for the symptoms to have any sort of accurate implementation. Further details on actually implementing the ANN are discussed next under the appropriate Implementation Details section.

Implementation Details

Bayesian Network (Kelly Holtzman and Mansi Patel)

Our initial thoughts when working with `pomegranate` were that we needed to develop probability distribution tables for the symptoms-to-illness dataset manually and build a Bayes’s Net direct from the tables. Obviously, making precise tables manually is subject to human error as it is very likely for us to mistakenly use the wrong dataset row because our chosen data were so large. This lead us to our first implementation that used the tables we created from the dataset with a known number of symptoms-to-illness (in our case, 5 symptoms to 6 illnesses); Mansi developed a component of the system that would take user input and that input had to be exactly the symptoms we had in the dataset. While the implementation performs as well as expected given known data, it is not a flexible design for future iterations of Medi-App given that we used data that would not be subject to change.

Once again visiting our design of the Bayes’s Net we found that it was possible to build the graph directly from samples using the `BayesianNetwork.from_samples()` function provided by `pomegranate`. The first attempt at using `from_samples()` we found that the resulting graph was not at all what the dataset intended to represent - we looked further into the function documentation and discovered that if we knew what format the data were representing that we could provide the function with a list of edges to exclude/include (knowing that each column in the dataset represented a node in the graph). Our second attempt at a Bayes’s Net implementation now allows for a dataset of any size to be used, knowing that all columns except the last in the dataset represent symptoms and the last represents illnesses. The second attempt continues to use the input component of the system.

Artificial Neural Network (Alwin Baby and Ali Rizvi)

ANN implementation uses `sk-learn` libraries for splitting, training and testing the data. In order to avoid overfitting, data was split into training and testing. 80% of the data was split into training and the rest 20% was split for testing. After splitting the data, features were scaled using `scaler.fit` module. Features were scaled so that the data can be uniformly evaluated. Feature scaling is performed only on the training data and not on the testing data. This is because in the real world, data is not scaled and the purpose of the neural network is to make predictions in the real world problems.

In the ANN implementation, most of the hard work is performed by the `MLPClassifier` module. `MLPClassifier` specifies how many iterations neural networks execute and how many hidden layers are present. Medi-App uses a total of three hidden layers of five nodes each and thousand iterations for the data. `MLPClassifier` creates data for `mlp.fit` function to train the algorithm on training data. The implementation then calls `confusion_matrix` and `classification_matrix` to find the scores. ANN implementation was not as hard but the plotting was tricky and unsuccessful: our initial goal was to show a graph with processed data and loss data. Since `matplotlib` is not compatible with Python 3, we could not generate the plots.

Results (Alwin Baby and Kelly Holtzman)

As per our program objectives outlined in the Project Objectives section, we evaluated the two above implementations for comparable features and used those results to determine a better suited algorithm for our application. It should be noted that our results are not necessarily absolute in making a decision, rather our intention is to single out one algorithm found to be easier to understand and implement as compared to the other, given our implementation and experience working with the algorithm Python packages. In the below table, we identified criteria for runtime (related to out project objective) and other criteria that were qualitative in nature but helped us to identify what Python packages should be used in the future (related to our personal project objectives):

Criterion for Comparison	Preferred Algorithm Given Criteria	Bayesian Network (Using `pomegranate` package)	ANN (Using `sk-learn` packages)
Full runtime (See Appendix A)	Bayes's Net, faster time	0.1044455678 seconds (from samples)	3.2411391276 seconds

Runtime to generate model (See Appendix A)	Bayes's Net, faster time	0.0436754752 seconds (from samples)	2.685259133 seconds
Understandability	Bayes's Net, because of it's simple, graphical nature	Bayes's Nets are comparatively simple to understand as the graphical format is straightforward and the idea that the distribution tables are tied to nodes made it easier to translate to a program	ANN was complex and confusing: choosing how many layers, nodes, and iterations would be needed was not straightforward. Not sure if our implementation is even the optimal choice for these features
Implementation Complexity	Bayes's Net, because examples online could be used almost directly for our purposes despite the difficulty looking at deeper package methods	Fairly simple to start with using the online examples, however the `pomegranate` package is not very well defined and the input/output provided by package objects and their functions was difficult to understand and use	Finding the right package modules for training was difficult because no online examples were using the latest python version. `sk-learn` had very good documentation once the right modules were identified
Dataset Complexity	ANN, because the structure of the ANN can be taken directly from dataset header data	It is difficult to prepare the dataset manually for the algorithm, we must know the format of data and program for it. It is possible for the package to figure out it's own pattern but it is usually not correct, hence we have to explicitly define the graph nodes and edges	Have to explicitly know the structure of the ANN in order to program it from a known dataset. Similar to Bayes's Net
Accuracy (of package used)	ANN, because it is easier in our case to obtain more data than it is to obtain known, accurate data	Depending on data used, if the dataset is accurate the Bayes's Net's predictive accuracy will be on par at least	More data means more accuracy: the number of data attributes is directly proportional to accuracy

Table 1: Comparison of Bayes's Net and ANN

We consider further how well the models solve our problem context: the Bayes's Net implementation can be manipulated to create more nodes per illness in the dataset - we can support the idea that illnesses are not mutually exclusive and a person may have a combination of illnesses given their symptoms; the ANN does not provide a straightforward way to relate and manipulate the data without having to refactor how the layers of the network are connected. Identified above, our team did not find a way to determine the optimal number of nodes, layers, or iterations our ANN should have. Further from the above table, Bayes's Net is preferred for most criteria over ANN.

Conclusions and Future Work (Kelly Holtzman and Mansi Patel)

From the results in the above section we determine that the Bayes's Net implementation was most effective for our purpose: Bayes's Net was the preferred choice for the criteria outlined, including runtime, understandability, and implementation complexity. We can conclude that future work on the Medi-App application should expand further on the Bayesian Network implementation portion. Please refer to Appendix C for additional information.

A major difficulty we had as a team on the project was dealing with such a large amount of data in our dataset, it caused us problems such as: creating the manual probability distributions took a long time and a lot of effort to ensure no mistakes were made, our machines could not manipulate and do work on the large dataset (133 columns by 4920 rows, not including the header row) (Anonymous, 2020) in a reasonable amount of time; on one such occasion where we tried used the dataset in its entirety in the Bayes's Net from samples method, the program did not complete in 10 minutes or less so it was not worth our time to use the full dataset (not stating that 10 minutes was a benchmark, but that it was obvious the time was exponential against dataset size).

Manual creation of the distribution tables is a better method as compared to samples if data can be obtained in such a form already or of Medi-App is expanded to contain a function for transforming samples into probability distributions: however, `pomegranate` already provided a method for learning the distributions from samples - we cannot say that the program can be made more efficient or faster if we were to implement learning from samples on our own. We can conclude then that our available hardware cannot perform learning from samples in a reasonable timeframe with increasing sample size, similar to how, as people, we take a long time to generate such tables by hand - far more time than software takes to do. From Das (2008): "...a possible solution would be to automate the process of [distribution] generation by learning the conditional probabilities from the database... for many practical problems one rarely finds a relevant database, even an inadequate one at that." It should be possible for our Bayes's Net implementation to work in a reasonable amount of time once compatible hardware is more commonplace, and we can anticipate our application would be more useful at such a time.

References

Anonymous (Nirma University) (2020). Disease Prediction Using Machine Learning (Version 1) [Data set]. Kaggle. https://www.kaggle.com/kaushil268/disease-prediction-using-machine-learning

Auras, H., Merouani, H. F., & Refai, A. (2016). Maintenance of a Bayesian network: application using medical diagnosis. *Evolving Systems*, 7(3), 187-196. https://doi.org/10.1007/s12530-016-9146-8

Das, Balaram. (2008). Generating Conditional Probabilities for Bayesian Networks: Easing the Knowledge Acquisition Problem. https://arxiv.org/abs/cs/0411034

Google Trends. (n.d.). *Coronavirus Search Trends.* Google. https://trends.google.com/trends/story/US_cu_4Rjdh3ABAABMHM_en

Neves, J., Cunha, A., Almeida, A., Carvalho, A., Neves., J., Abelha, A., Machado, J., & Vicente, H. (2015). Artificial Neural Networks in Diagnosis of Liver Diseases. *Information Technology in Bio- and Medical-Informatics (2015)*. https://doi.org/10.1007/978-3-319-22741-2_7

Schreiber, J. M. (2018). *pomegranate* (Version 0.13.2). Schreiber, J. M.. https://pomegranate.readthedocs.io/en/latest/

Shanmuganathan S. (2016) Artificial Neural Network Modelling: An Introduction. Artificial Neural Network Modelling. *Studies in Computational Intelligence*, 628. https://doi.org/10.1007/978-3-319-28495-8_1

University of California, Berkeley (UC Berkeley). (2014). *Bayes's Nets I, II, III, IV* [PowerPoint slides]. University of California, Berkeley. http://ai.berkeley.edu/lecture_slides.html

Appendix A: Algorithm Runtime Average Calculations

Iteration	Bayes's Net (seconds)	ANN (seconds)
1	0.123239388	2.763918929000001
2	0.07368002100000037	2.2947295830000005
3	0.12445978000000046	2.5182277599999994
4	0.12976832000000194	4.245963313
5	0.07108033000000002	4.382856053000001
Average	0.1044455678	3.2411391276

Table A1: Runtime Iterations for full program execution*

Iteration	Bayes's Net (seconds)	ANN (seconds)
1	0.04373709299999895	1.795540496000001
2	0.03516251100000112	1.9773536829999996
3	0.055997171000001345	2.0349295079999994
4	0.047476234999999534	3.799651648000001
5	0.03600436600000023	3.8188203299999994
Average	0.0436754752	2.685259133

Table A2: Runtime Iterations for program model completion*

*Results from a 2015 MacBook Air running macOS Catalina (version 10.15.7) with 4 GB 1600 MHz DDR3 RAM and 1.6 GHz Dual-Core Intel Core i5 processor.

Appendix B: Failed Implementation of K-Means Clustering

K-Means Clustering (Ali Rizvi)

K-means clustering is an unsupervised machine learning algorithm for partitioning a given data set into a set of *k* groups (i.e. *k* clusters), where *k* represents the number of groups pre-specified by the analyst. It classifies objects in multiple groups (i.e clusters), such that objects within the same cluster are as similar as possible (i.e. high intra-class similarity), whereas objects from different clusters are as dissimilar as possible (i.e. low inter-class similarity), (UC Business Analytics, 2017). The basic idea behind k-means clustering consists of defining clusters so that the total intra-cluster variation (known as total within-cluster variation) is minimized. There are several k-means algorithms available.

The standard algorithm is the Hartigan-Wong algorithm (1979), which defines the total within-cluster variation as the sum of squared distances Euclidean distances between items and the corresponding centroid: $W(C_k) = \sum_{x_i \in C_k} (x_i - \mu_k)^2$. Where Xi is a data point belonging to a cluster C_k and μk is the mean value of the points assigned to the cluster C_k (UC Business Analytics, 2017). K-means stores centroids that it uses to define clusters. A point is considered to be in a particular cluster if it is closer to that cluster's centroid than any other centroid. K-Means finds the best centroids by alternating between (1) assigning data points to clusters based on the current centroids (2) choosing centroids (points which are the center of a cluster) based on the current assignment of data points to clusters (Stanford, 2012). This can be shown in the figure below:

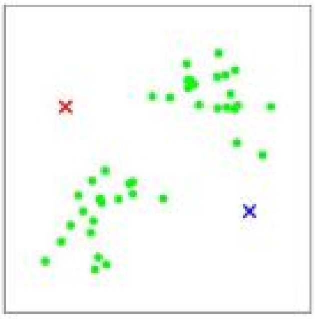


Figure B1: Random initial cluster (Stanford, 2012)

K-means clustering implementation is investigated in the detection of Alzheimer's disease by Vijayalakshmi, Pallawi, Shruti, and Genish (2020). Vijayalakshmi, et. al enhanced k-means algorithm is used where it is able to successfully extract the hippocampus and gives comparatively better results than existing approaches.The advantage of this algorithm is that it takes very less time in computational and is very simple as compared with other algorithms. Below is the workflow model of this algorithm:

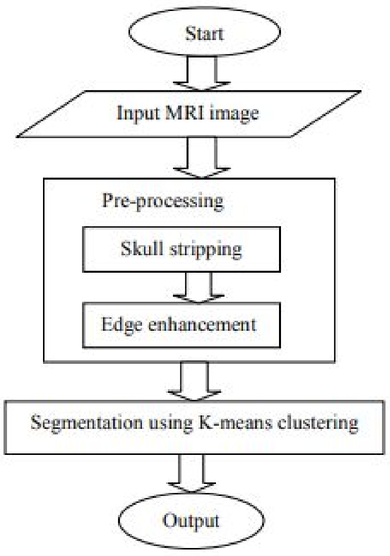


Figure B2: K-Means clustering algorithm used in the segmentation process of hippocampus.

The major finding of the paper was that the k-means algorithm was able to reduce the complexities that were faced during the segmentation of hippocampus through the combination of region growing and clustering technique.Using the clustering technique, it was able to classify the tissues and groups them having similar types of characteristics which helped in reducing the information so that the hippocampus can be easily analyzed in less time. Therefore, k-means should be used for a data-predicting model as the data structure it requires is very simple and each iteration uses the information of previous iteration which then improves the computational speed of the algorithm and gives us the faster and better results.

Implementation Failure Details (Alwin Baby)

K-Means clustering is a simple but powerful algorithm. There are a lot of real world applications such as insurance fraud detection, identifying fake news, marketing and sales, etc. (Whittaker, 2019), which uses K-Means clustering. Clustering is the process of dividing the entire data into groups based on the data pattern. Clustering works great with numerical data sets but not so much with non-numerical data. Since Medi-App uses numerical data and non-numerical data, it's best not to use clustering. Clustering works really well when the dataset has different classifications of non-numerical data such as age, gender, symptoms etc.

Medi-App uses data which are classified similarly to each other, and so it was difficult to group the data for prediction. Additionally, limited knowledge on clustering before this course made it difficult to understand how datasets may be grouped. Further difficulties understanding the associated `sci-kit` packages for clustering algorithms made it so the team went on to investigate and implement ANN, alternatively.

Appendix B References

K-means Cluster Analysis: UC Business Analytics R Programming Guide. (2020). Retrieved 28 October 2020, from https://uc-r.github.io/kmeans_clustering

CS221. (2020). Retrieved 29 October 2020, from https://stanford.edu/~cpiech/cs221/handouts/kmeans.html

Whittaker, C. (2019). 7 Innovative Uses of Clustering Algorithms in the Real World. Retrieved December 03, 2020, from https://datafloq.com/read/7-innovative-uses-of-clustering-algorithms/6224

Vijayalakshmi, S. S., Pallawi, Shruti & Genish, T. (2019) Alzheimer Disease Detection Using Edge Enhanced K Means Clustering Algorithm. *5th International Conference on Next Generation Computing Technologies (2019)*. http://dx.doi.org/10.2139/ssrn.3545092

Appendix C: Medi-App Codebase

Our team has a GitHub software repository which we used to track our progress and make our study publicly available. Please find the repository at the following:

Baby, A., Holtzman, K., Patel, M., and Rizvi, A. (2020). Medi-App. GitHub. https://github.com/holtzmak/Medi-App