Machine Learning for Understanding Aging

Design Document

sdmay20-41
Dr. Julie Dickerson - Client/Advisor
Aria Sheets - Report Manager
Ian Simon - Chief Engineer
Jacob Laing - Chief Engineer
Nathan Carter - Test Engineer
Samantha Williams - Meeting Scribe
Scott Rose - Meeting Facilitator
sdmay20-41@iastate.edu
http://sdmay20-41.sd.ece.iastate.edu/

26 April 2020

Executive Summary

Engineering Standards & Design Practices

This project will consider the standards and practices, including:

1. The product must ensure the privacy of the people whose data we are using to use in the creation of our machine-learning program.
    a. All Personal Health Information (PHI) must be anonymized.
    b. Any PHI that is transmitted must be encrypted.
2. The product must be accessible, and the information output must be easily understandable.
3. The product must be fast to learn, and up to today's standards of machine learning.
4. The product must output results that are accurate so that others can use the data we obtain through our program with reliability.
5. The product must be created considering the knowledge that we have learned from taking the *CITI Program's Social/Behavioral Research Course*[7].

Summary of Requirements

Functional Requirements:

- The program accurately assesses patterns in data related to the MIDUS datasets.
- Users can continue to input data to increase the accuracy of the program.
- The program outputs aspects of the input data that affect the experience of aging.
- Project completed by May 2020.
- The project will follow the guidelines defined by the community derived data specifications
- The program must have extension points that will allow for users to add their own data processing methods

Non-Functional Requirements:

- Written in Python.
- Clear, well-documented code.
- The privacy of subjects included in test data is considered.
- An appropriate size of training data is used to properly train the program.
- The results of the running program are outputted in a user-friendly format.

Applicable Courses from Iowa State University Curriculum

Iowa State has prepared us for this project with many classes, including:

- COM S 227: Object-oriented Programming
- COM S 228: Introduction to Data Structures
- COM S 311: Introduction to the Design and Analysis of Algorithms
- COM S 474: Introduction to Machine Learning
- MATH 207: Matrices and Linear Algebra
- S E 309: Software Development Practices
- S E 329: Software Project Management
- S E 339: Software Architecture and Design

New Skills/Knowledge Acquired Not Taught In Courses

This project has taught and will teach us new knowledge that we haven't learned before, including:

- Neural network design.
- Cost functions used in neural networks.
- Analyzing large data sets.
- Managing the privacy of data used programs for analyzing datasets.
- Regression analysis.

Table of Contents

1. Introduction 5

1.1 Acknowledgment 5

1.2 Problem and Project Statement 5

1.3 Operational Environment 5

1.4 Requirements 5

1.5 Intended Users and Uses 6

1.6 Assumptions and Limitations 6

1.7 Expected End Product and Deliverables 6

2. Specifications and Analysis 7

2.1 Proposed Design 7

2.2 Design Analysis 7

2.3 Development Process 7

2.4 Design Plan 7

3. Statement of Work 8

3.1 Previous Work And Literature 8

3.2 Technology Considerations 8

3.3 Task Decomposition 9

3.4 Possible Risks And Risk Management 10

3.5 Project Proposed Milestones and Evaluation Criteria 10

3.6 Project Tracking Procedures 10

3.7 Expected Results and Validation 10

4. Project Timeline, Estimated Resources, and Challenges 11

4.1 Project Timeline 11

4.2 Feasibility Assessment 12

4.3 Personnel Effort Requirements 12

4.4 Other Resource Requirements 13

4.5 Financial Requirements 13

5. Testing and Implementation 13

5.1 Interface Specifications 13

5.2 Hardware and software 13

5.3 Functional Testing 14

5.4 Non-Functional Testing 14

5.5 Process 14

5.6 Results 16

6. Closing Material 16

6.1 Conclusion 16

6.2 References 17

6.3 Appendices 17

List of Figures/Tables/Symbols/Definitions

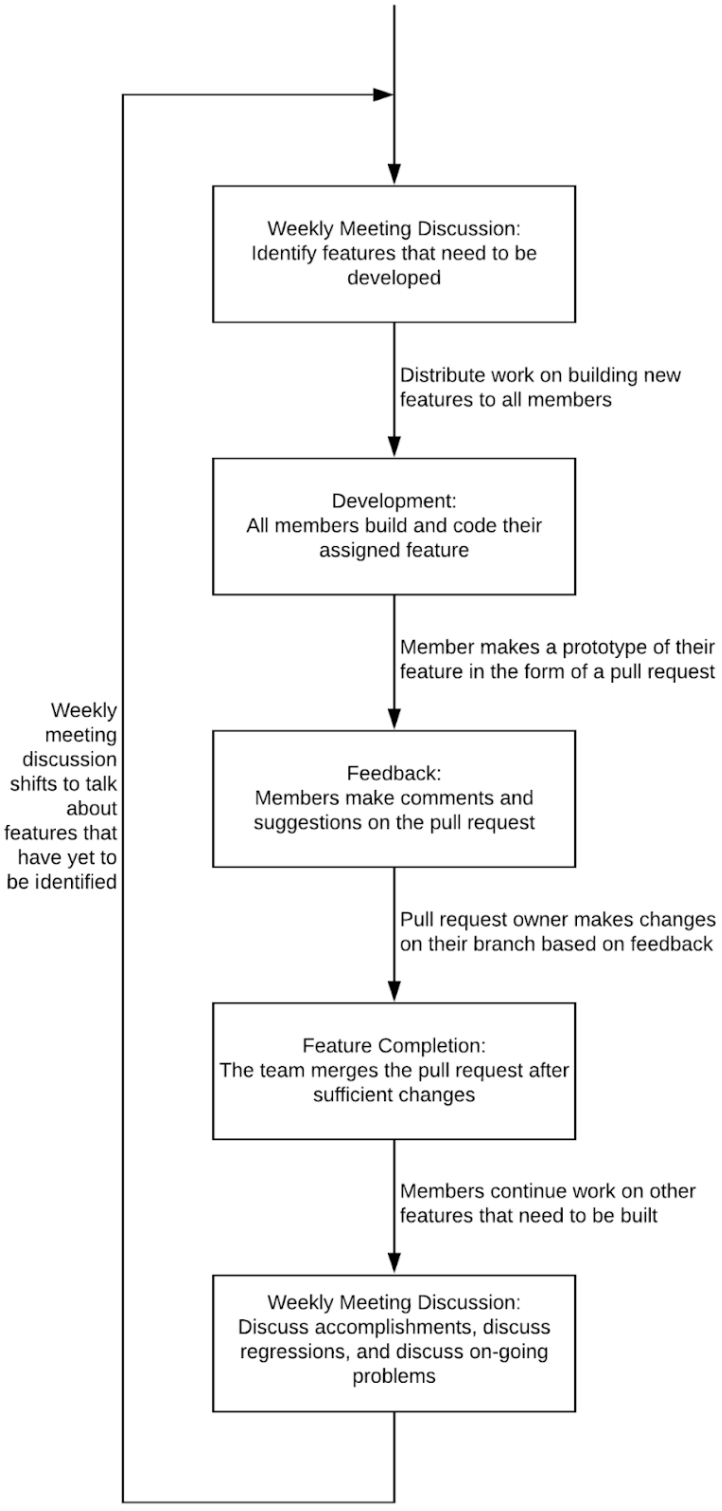

Figure 0.1: Weekly Project Plan

1. Introduction

1.1 Acknowledgment

Thank you to Dr. Julie Dickerson for providing guidance and structure for this endeavor. Thank you to Inter-University Consortium for Political and Social Research (ICPSR) for providing the large data sets that were used in the training and creation of the program. Thank you to Dr. Jennifer Margrett for providing information regarding the topic of aging and the important factors to focus on throughout our research. Thank you to Iowa State University for allowing us to use their hardware to run and test our program.

1.2 Problem and Project Statement

General problem statement: Human aging is a topic that has been studied throughout history. Scientists, doctors, sociologists, and the general public all want to know what characteristics indicate a decline in health and what actions can be taken to slow down this decline. Knowing this information can allow people to increase their life expectancy and overall quality of life.

General solution approach: Using health data collected by ICPSR during the *Midlife in the United States* (MIDUS) *Biomarker Project*[8], by the use of sensors and questionnaires, and research regarding machine-learning techniques, our product will find and return the various patterns found in the data. After training the program using large quantities of training data, it will be able to accept data and return information regarding actions in which the user can take that have proven effective for other patients with similar characteristics.

1.3 Operational Environment

Because our end-product requires mining a large set of given data to make the machine learn and output results, we recommend that our software is run on a high-end machine, preferably one with a high-end GPU. We will be developing our software expecting our users to be using a high-end machine. If they do not, the time for results to appear from our product will take longer. We will be using an Iowa State University Virtual Machine during the development of our product and will strive to host a service on an Iowa State Virtual Machine for others to access our project.

1.4 Requirements

Functional Requirements:

- The program accurately assesses patterns in data related to aging.
- Users can continue to input data to increase the accuracy of the program.
- The program accurately assesses patterns in data related to the MIDUS datasets in a visual/graphical form.
- Project completed by May 2020.
- The project will follow the guidelines defined by the community derived data specifications
- The program must have extension points that will allow for users to add their own data processing methods.

Non-Functional Requirements:

- Written in Python.
- Clear, well-documented code.
- The privacy of subjects included in test data is considered.
- An appropriate size of training data is used to properly train the program.
- The results of the running program are outputted in a user-friendly format.

1.5 Intended Users and Uses

The users of our end product will mostly consist of sociology and psychology scientists. They will be using our product to analyze data in hopes of finding patterns related to aging.

1.6 Assumptions and Limitations

Assumptions:

- The program will be used by scientists and researchers.
- The program will be run on high-end hardware.
- The program will be developed with the *TensorFlow*[2] machine learning framework
- We will use existing libraries such as *Plotly*[5] to develop our machine learning module.
- The data provided to us is already anonymized.

Limitations:

- The software will only be used by scientists, sociologists and researchers.
- The budget is limited to hardware owned by the team members and hardware owned by Iowa State University.
- The data input into the program must fit a specific format.
- The program will only have English as the language.
- The program will rely heavily on GPU usage.
- The program needs to be completed by the beginning of May 2020.

1.7 Expected End Product and Deliverables

Our first major deliverable would get the data parser and data formatter set up. These are necessary parts to be completed before we can expect our program to learn data. Even though this is our first deliverable, it is expected that we can work on the machine learning module by using a dummy or mock data until the data parser and formatter is set up. The expected delivery date for this is late February 2020.

Our Second major deliverable is to create the machine learning module. This module reads in formatted data from the data parser and formatter module. It then produces a result in the form of data from the learning process. This data is then sent to our visualizer module where the loss will be displayed on a graph through the plotly library. The expected delivery date for this is late March 2020 to early April 2020.

Our last major deliverable is to create the result data parser and visualizer modules. These are the final pieces of our program, and will allow scientists/researchers to visualize the data in a friendly format, as well as show the pure data results of learning. The expected delivery date for this is late April 2020.

In the end, we will have created a program that can receive and analyze large datasets concerning aging and its effects. The program would use machine learning to possibly view this data in a different light than conventional methods. Our clients will be provided with our findings, as well as a framework for using the trained program on new sets of data. The hope will be that our program can provide new insights into the data collected in the MIDUS data sets.

2. Specifications and Analysis

2.1 Proposed Design

We will be using TensorFlow in Python to create an unsupervised machine learning algorithm. The program will take in data from the MIDUS (Midlife in the United States) study conducted at the University of Michigan which already anonymized data to protect the privacy of individuals. The program will find patterns in the MIDUS dataset based on the data input into the program. The program will run on a GPU cluster provided by Iowa State University. The data will be produced in a readable format so that it can be presented to scientists interested in our findings.

During the first semester, we spent most of our time researching the building blocks required to understand how machine learning functions. This includes research on concepts like neural networks and back-propagation. We have watched simulations and videos discussing machine learning and have inspected and compiled machine learning code that has been shown to work.We have also completed trial runs of regression analyses with other data sets in the sklearn database. We did this in order to understand which type of regression will best fit the data we hope to use for our project. We decided that the lasso regression was the most effective.

The second semester was spent creating our three main deliverables. We first spent time developing our data parser and formatter module. We had to research the MIDUS dataset to determine what data we wanted from the dataset.The data then needed to be cleaned, normalized, and sent to our machine learning module. Our machine learning module then separated the data into training and testing subsets before running. The machine learning module then outputs loss data points for our data visualizer to display.

2.2 Design Analysis

We have done preliminary research into what methods we will be using to design our system. We have found that the TensorFlow library will best suit what we need. To continue with the project, we need to figure out a system to read in the data for cleaning. We will also need to do experiments with TensorFlow to familiarize ourselves with the library and also do more research into unsupervised learning because most of our research so far has been on supervised learning.

2.3 Development Process

For this project, we have decided to follow the *Agile Development Process*[1]. The Agile Development Process is a very intuitive process and is well-known amongst everyone within our group making it great for our team to use. This process allows us to easily assign new tasks to members, monitor the tasks that are being worked on, and quickly fix any issues that may arise from any tasks. Along with those positives Agile allows us to quickly and efficiently make changes to our project as we may see fit.

2.4 Design Plan

There are four parts to our project: The GUI, the data extraction module, the machine learning module, and the data visualization module. The parts of the GUI module are colored in yellow in figure 2.4.1. The point of this module is to help users set up the parameters for running the program. In this module, users can select what data points they want to use and how they these data points to be processed. In the GUI users will also have to set what files to use and any parameters they want to set for the machine learning module.

In the figure 2.4.1, the data extraction modules are colored in purple. This is where the data is read from the files, preprocessed, normalized , and combined into one big array. Since the preprocessing of data for a machine learning experiment is usually very specific to what the user is trying to accomplish, the module will include an extension point for users to code their own

preprocessing methods if the preprocessing methods that we supply do not cover the needs of the user.

The modules in blue in figure 2.4.1 are the machine learning modules. They will be passed data from the data extraction modules, and perform a lasso regression on the data. The results will then be passed to the data visualization module, which is colored in pink in figure 2.4.1.

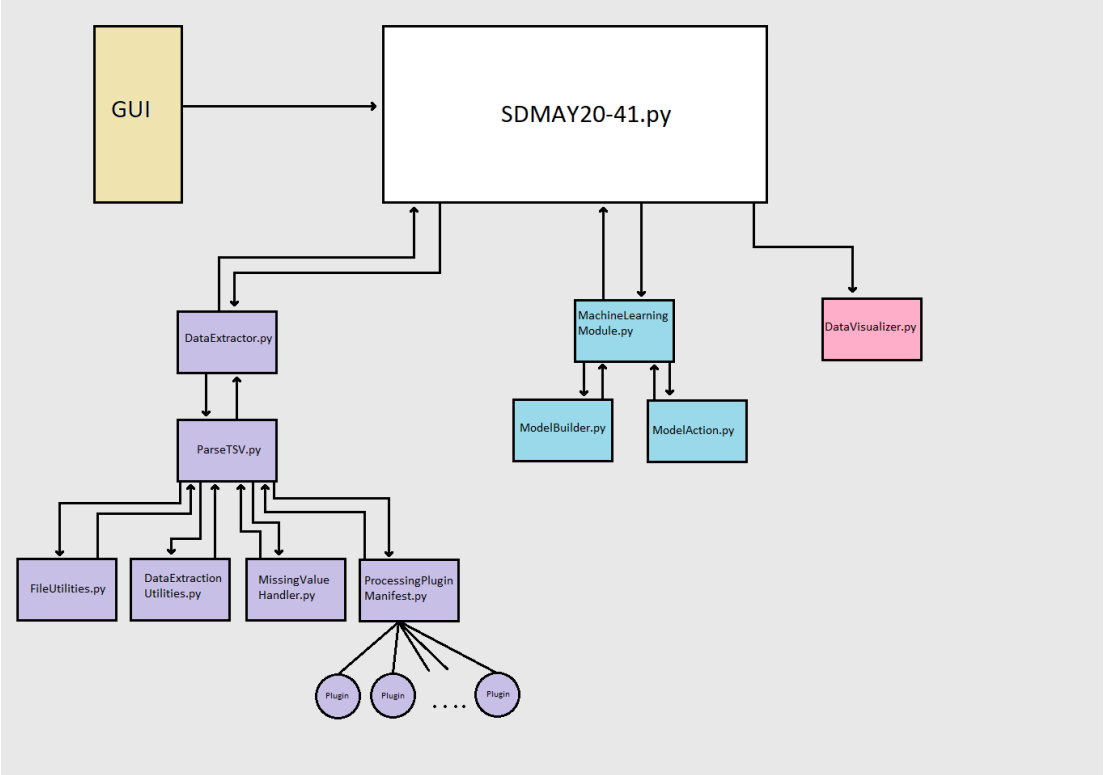

Figure 2.4.1: Project Design and Modules

3. Statement of Work

3.1 Previous Work And Literature

We will not be using previously created software, and we expect to use our own, with the exception of libraries and dependencies. There has been research on machine learning to study aging, but not using the data we will be using. We expect our project to bring new information to the world of Gerontology.

3.2 Technology Considerations

We plan on using TensorFlow which has many advantages and some disadvantages, especially when compared to its competitors:

- Advantages of TensorFlow include its graph visualization, its library management, its tools for debugging, and scalability.
- Disadvantages of TensorFlow include its lower computation speed and limited GPU support.

Other technologies we could've used included *PyTorch*[4] and *scikit-learn*[3]. While Pytorch may have some better debugging tools it is a fairly young framework and generally isn't suggested

for what we are trying to accomplish. scikit-learn, while well documented, has a tendency to be slower as it doesn't use hardware acceleration.

We plan on using Plotly as a way to visualize the results our project creates:

- Advantages of Plotly include its interactive chart, diverse set of graphs, and ease of use.
- Disadvantages of Plotly include its low level of standardization.

Alternatives to Plotly include *Matplotlib*[6], but we determined that Plotly fit our project better and was easier to use. Plotly is not as standardized as Matplotlib, but it is generally simpler than Matplotlib.

3.3 Task Decomposition

We expect to decompose each module into a series of tasks to complete, each with their own dependencies on other tasks. These tasks will be split up throughout the weeks in order to keep progressing towards the final project. Decomposing the tasks will also allow teammates to be able to focus on small goals/tasks to finish rather than feeling overwhelmed by a large daunting task. The large decomposition includes the deliverables for the project, which we will then break down into smaller tasks similar to the following:

1. Inception
    a. Gathering requirements
    b. Discussion with our client
    c. Research machine learning concepts
2. Design
    a. Document project components
    b. Document component communication
    c. Create UML diagrams
3. Data Parser
    a. Parse column parameter file getting column information from the user
    b. Parse MIDUS TSV data files collect columns specified by the user
    c. Create a plugin system that will allow users to create their own preprocessing methods
    d. Create a data processing plugin specific for the system test.
4. Data Formatter
    a. Allow the user to handle missing data
    b. Normalize the data
    c. Combine the data into one array if it was read from multiple files
    d. Send the data to the machine learning module.
5. Machine Learning Module
    a. Split the data into training and testing subsets.
    b. Split the training subset into smaller subsets for individual epochs.
    c. Train the machine learning algorithm on each of the testing subsets once.
    d. Evaluate the machine learning algorithm on the testing subset.
    e. return the loss from the evaluation of the machine learning algorithm.
    f. Client approval of the machine learning module
6. Result Data Parser & Visualizer
    a. Create parser to parse result data from machine learning module
    b. Create plotly graph to display the loss from the machine learning algorithm.

3.4 Possible Risks And Risk Management

Our beginner's knowledge of machine learning and linear algebra affects some of our members. This is a risk that may impact the progress and timeline of our project. This may also contribute to accuracy issues in the results if we are unable to make up the time impacted. We're mitigating this risk by researching a lot on machine learning in our first semester and doing practice exercises related to the technical requirements.

Another issue that may slow down our plan is not being able to easily read in data to plug into our machine learning algorithm. To mitigate this we have been contacting other researchers to try and find a tool that can be used to read in DDI files and if that doesn't work we will be working on how we can interpret the massive CSV files to create a reader that does the work for us.

In any machine learning project, data is always a big issue. If there is not enough for a machine learning model to train on, the results of the model could be skewed. One risk related to this is that during our training and testing phase of our model, we could find out that we do not have enough data for accurate predictions.

3.5 Project Proposed Milestones and Evaluation Criteria

Each deliverable we have is considered a major milestone for our project. Our most important milestone will be finishing the machine learning module, as that is the main focus of our project. This milestone will be the most work-intensive module to create. In addition, each module will be thoroughly unit tested as it is being developed to ensure that it is bug-free.

Our first important milestone will be recreating a study already done on the MIDUS data set. If we are able to do that, then we will have a solid base for making additional observations about the data. This milestone will prove that both our data are filtered correctly and that our regression model works.

The project will be evaluated based on the accuracy and ability of the machine learning algorithm to interpret the data. Our client will work with us to determine whether it is meeting the standards we have set out to achieve. The results will also be evaluated on the visual presentation we display with the data. In other words, is it easy for someone not involved in the project to read and understand the results.

3.6 Project Tracking Procedures

We will use both GitLab and Trello to keep track of our progress. We will use GitLab for the low-level details, code reviews, and code changes, and use trello for the high-level details and card completion. We will have weekly meetings where we discuss the progress that each member has made and what parts we are struggling with.

3.7 Expected Results and Validation

The desired outcome of this project is to analyze data on aging to see if there are any patterns that show up that may indicate how one can lead a healthier lifestyle. With the analyzed data we would be able to then send it off to other researchers who know more about aging and can use our data to help others make healthier decisions.

We expect we will have a fully-functioning program that can input data and output results that help understand the effects of aging. We will do extensive testing and communication with faculty members knowledgeable on the matter to confirm what we create is accurate. We also hope to introduce new data at the end to see how successful the training data was in training our module.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

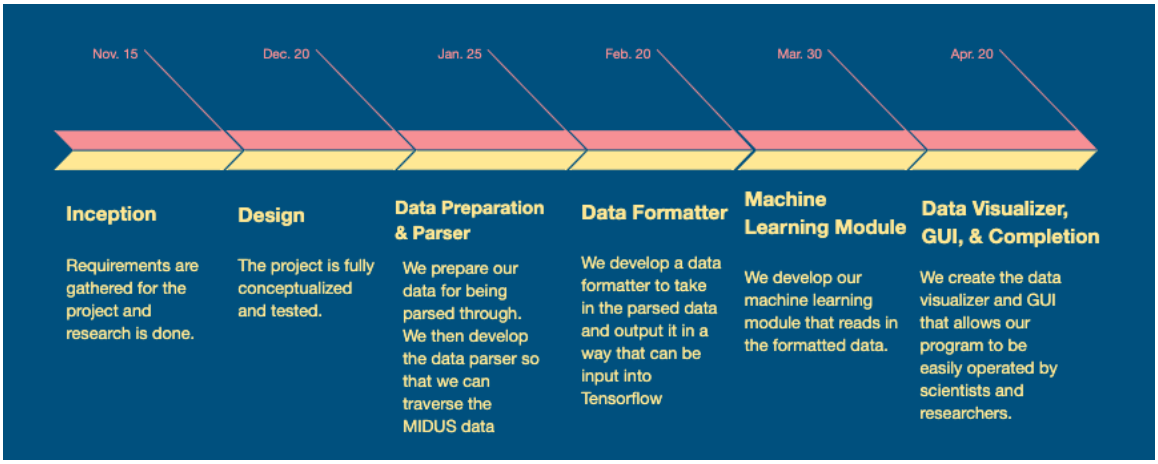

Figure 4.1.1: Project Timeline

Description of the milestones and tasks in the Timeline above:

Deadline - November 15, 2019: Inception

- Requirements are gathered for the project.
- Research is done on multiple machine learning concepts.
- The use of libraries and technologies are determined.

Deadline - December 20, 2019: Design

- The project is fully conceptualized and tested.
- Design of modules are created with appropriate data paths.

Deadline - January 25, 2020: Data Preparation & Parser

- We develop the data parser.
- This module is thoroughly tested.

Deadline - February 20, 2020: Data Formatter

- We develop the data formatter.
- This module is thoroughly tested.

Deadline - March 30, 2020: Machine Learning Module

- We create our machine learning module that reads in the formatted data.
- This module is thoroughly tested.

Deadline - April 30, 2020: Data Visualizer, GUI, & Completion

- We create the data visualizer that allows our results to be easily viewed by scientists and researchers.
- The GUI is created so that researchers don't need to use the command line.
- This module is thoroughly tested.

4.2 Feasibility Assessment

We predict that it is highly likely that our project will be finished by the end of the second semester. The biggest challenge we must overcome includes our beginner's knowledge on machine learning. However, the research and practice we are doing prior to the actual production of the module will help us overcome this challenge. Another challenge I may see us running into is being able to understand the data that we are using to train our machine learning algorithm. While the data may be well documented, there is a lot of it that we have to go through so being able to piece together the different variables will be essential to figuring out how the data was collected and what it means as a whole.

4.3 Personnel Effort Requirements

Task	**Hours per week**	**Explanation**
Inception	3-5, 6+	The inception process includes gathering requirements, discussing with the client, and researching machine learning topics. These are all items that require time and focus in order to create a strong base for the understanding of machine learning and how we can use it in our project. Therefore, the items here would be described as medium- to high-complexity items.
Design	1-2, 3-5	The design task includes documenting project components, documenting component communication, and creating UML diagrams. These items are more related to discussion and overview of our project and design process. Since these items don't require a large amount of technical research or time, they are considered low- to medium-complexity items.
Data Parser	3-5	The data parser involves parsing MIDUS TSV data files. These tasks are items that we have researched in the inception stage, so they should not be too time consuming. However, due to their technical nature, they are classified as medium-complexity items.
Data Formatter	3-5	The data formatter takes the parsed MIDUS TSV data files and formats it into a format that is easily usable by the machine learning module. This is a technical requirement, so this is considered a medium-complexity item.
Machine Learning Module	3-5, 6+	Creating the machine learning module consists of setting up machine learning training, testing the machine against new data, and client approval of the machine learning module. Depending on how much our preparation aids us in this phase, the task of creating the machine learning module could be considered medium- to high-complexity.

		The item that we suspect will take the most time will be training the machine and editing the module based on our results.
Data Visualizer & Completion	3-5	The data visualizer and completion phase includes creating a parser to parse result data from the machine learning module and creating a visualizer to show the plots of results. These tasks require a lot of data parsing and interpretation. We do not expect to run into many challenges while completing this task, so it is classified as a medium-complexity task.

Table 4.3.1: Outline of Project Tasks

The duration of the task depends on the complexity of it. For a low-complexity task, we expect 1-2 hours to be spent. For a medium-complexity task, we expect 3-5 hours spent. For a high-complexity task, we expect 6+ hours and plenty of technical communication with other team members to finish the task.

4.4 Other Resource Requirements

The resources required will be our own personal machines and the data that we will use for our machine learning program. The MIDUS data set will be required for our program.

4.5 Financial Requirements

We did not have any financial requirements due to all our work being on our own personal machines.

5. Testing and Implementation

5.1 Interface Specifications

Our current plan for the user interface is for the program to be run on a Command Line Interface (CLI) or our GUI. Users will pass parameters to the program through the CLI or GUI, and a graph will be created using the Plotly Python Open Source Graphing Library based on the output of the program using those parameters. Additional output for the user is printed to txt files. The program will be accessed via GitLab. The repository will be open for users to pull our code, and use it at their own discretion. There will be no significant Hardware and Software interfacing for our project.

5.2 Hardware and software

To run our program, users must have Python 3.7.4 installed. On top of that the following Python libraries must be installed: NumPy version 1.17.2, Matplotlib version 3.1.1, and TensorFlow version 2.0.0. Optionally, users will have the ability to run the program using the NVIDIA CUDA® Deep Neural Network library (cudNN). The program will be able to run without this library, but the use of this library will improve the performance of the program.

The minimum system requirements for running our program are an Intel Atom® processor or Intel® Core™ i3 processor, 4 GB of disk space, Windows* 7 or later, macOS, or Linux. These system requirements are based on the minimum system requirements for running Python 3.7.4 and the minimum system requirements of the Python libraries we will be using. If you are planning on

using the cuDNN library, you must have a cuDNN compatible GPU. To understand the compute capability of the GPU on your system, see NVIDIA's documentation on cuDNN.

5.3 Functional Testing

There are 3 types of functional testing that will be covered in our components: unit testing, system testing, and acceptance testing.

For unit testing, we plan on having 80% code coverage for all of our non-machine learning components of our project. For our machine learning components, we plan on doing black-box testing where we give our machine learning model certain inputs and see if the output matches what we expect it to be.

Our system tests will be done by recreating studies that have already been done on the MIDUS dataset. We will give the program appropriate parameters that should recreate those studies. We can then compare the output of those studies with the output of our program. If the output of our program matches the output of those studies, then we will know our system works.

The acceptance testing of our program will be done on a weekly basis. Each week we meet with our client and go over what we have done. During these meetings we will get feedback on whether or not our work complies with what the client is expecting.

5.4 Non-Functional Testing

There are 3 main areas that our non-functional tests will cover: performance, usability, compatibility. The performance requirements will be gathered from our meetings with our client. Performance will most likely be measured by the amount of time it takes for the program to run given a set of inputs. We will run the program on a virtual machine that has the minimum system requirements and see if our program is able to complete in the time specified by our client.

Usability Testing will be done by people who we deem to be the primary users of this software. People such as Dr. Julie Dickerson and Dr. Jennifer Margrett will be given the final project and asked to perform a task. We will use their feedback to improve our product.

Compatibility testing will be done using the variety of machines we use. This includes Windows, Linux, and macOS machines, all with different hardware. We will then test our program on these machines. Since our product will use popular and well developed Python libraries, compatibility issues are expected to be minimal.

5.5 Process

Our testing strategy will take part in 3 phases. Certain tests cannot be implemented until certain components are finished. We cannot test that without having multiple completed components. Phase 1 of testing will just run unit tests on the python components that parse and format data. Phase 2, will begin once the Python machine learning components are ready, and include the system test. Phase 3 will begin after the machine learning module is complete, and it will test that data visualization and GUI of the project. The flow of the overall testing strategy can be seen in Figures 5.5.1. The unit test flow can be seen in Figure 5.5.2.

We implemented continuous integration (CI) into our project using GitLab's built-in pipeline tool and a test runner. The test runner is one command that we can type into the command line to run all of the project's unit tests. The pipeline automatically runs a series of commands that compiles, builds, and runs our program's unit tests on the latest commit for all branches and all merge requests. The pipeline uses the test runner command in order to easily run all of our tests. The pipeline will detect any failure in compiling or the unit tests and show it on GitLab. Because of this, we were able to develop and automatically get notified each time we broke previously-working functionality, allowing us to develop without worries of unknowingly breaking other parts of the code.

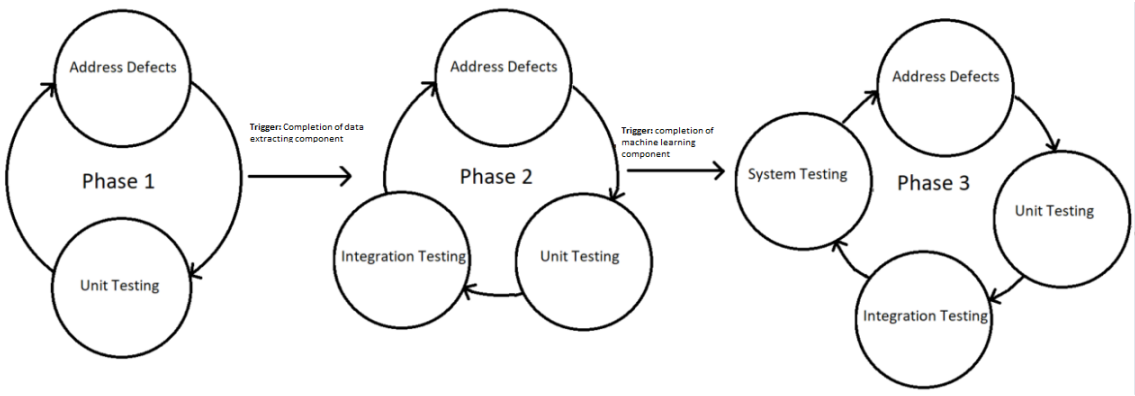

Figure 5.5.1: Testing Cycle Flow

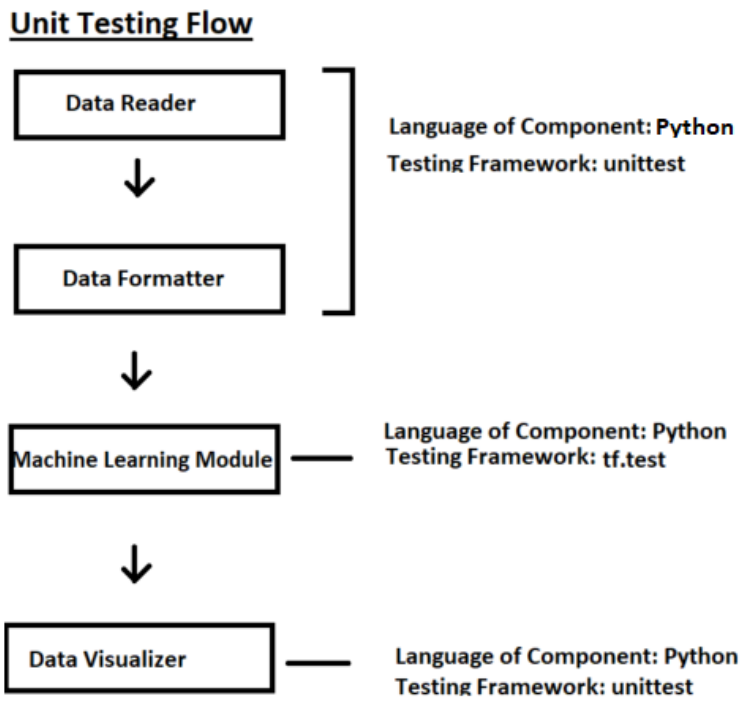

Figure 5.5.2: Unit Testing Flow

5.6 Results

We had many successes using the pipeline and the test runner of our program. The pipeline was able to catch many instances of breaking functionality. Because we only allow changes through merge requests, the pipeline would always catch these instances, and we wouldn't merge until those problems were resolved.

We had a few problems with the pipeline initially, but as we continued, the problems phased away. The main problem was that none of us had set up a pipeline before, and it required research to figure out how to set up a pipeline on Gitlab. Another problem that we ran into was when a dependency couldn't be installed on the machine that ran the pipeline. Because we don't own the machine and the code it ran remotely, it was hard to pinpoint why the dependency couldn't be installed. After research, we were able to find out it was a CPU issue, and we were able to find a configuration for the dependency that could be installed on the machine that ran the pipeline.

6. Closing Material

6.1 Conclusion

During the first semester we completed the inception phase of our project. The bulk of the design was finished then. We also researched potential studies to mimic for the testing of our machine learning module. We decided which columns should be included to best represent the study. Since we used agile, we continuously adapted our design to our project's needs. During the second semester we moved into our programming phase. We first prepared and finished our data parser and formatter to read our MIDUS dataset. Our 2nd component completed was our machine learning module, which took the cleaned data from our data parser and ran our machine learning algorithm on it returning the loss. We then developed our visualization module with plotly to display the loss in an easy to read way. We also included additional user output depicting the specifications of the program run so the user can easily replicate the run if they choose to. Lastly, with the extra time at the end of our semester we were able to finish a GUI which will give users an easier way to use our program.

Human aging has been a topic of research throughout history. It is human nature to want to live as long as possible, and people want to know what factors in their lives can contribute to the quality of their aging. Through machine learning, our product aims to answer some of these questions. By using the data collected in the MIDUS study, our machine learning program will output the patterns in the input files during training. We hope that our product can then be used to suggest actions the person of study could use to improve the quality of their aging. The product's completion date is May 2020.

Utilizing machine learning for this project automates the tedious task of manually analyzing large data sets. Once completed, users will be able to quickly receive results based on their input data. The time it takes to implement the machine learning methods will be well worth the time saved by being able to skip the manual process.

6.2 References

1. “Agile 101.” *Agile Alliance*, https://www.agilealliance.org/agile101/.

2. “End-to-End Open Source Machine Learning Platform.” *TensorFlow*, https://www.tensorflow.org/.

3. “Machine Learning in Python.” *scikit-learn*, https://scikit-learn.org/.

4. “Open Source Machine Learning Framework.” *PyTorch*, https://pytorch.org/.

5. “Plotly Python Open Source Graphing Library.” *Plotly*, https://plot.ly/python/.

6. “Python 2D Plotting Library.” *Matplotlib*, https://matplotlib.org/.

7. “Research Ethics and Compliance Training.” *CITI Program*, https://about.citiprogram.org/.

8. Ryff, Seeman, Weinstein. “Midlife in the United States (MIDUS 2): Biomarker Project, 2004-2009 (ICPSR 29282).” *ICPSR*, https://www.icpsr.umich.edu/icpsrweb/ICPSR/studies/29282.

9. “Cloning a Repository.” *Cloning a Repository - GitHub Help*, https://help.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository.

6.3 Appendices

Appendix I - Operation Manual

Introduction

This section will provide guidance for users of the sdmay20-41 machine learning module. The intention of this program is to receive data from the Mid Life in the United States study carried out by the MacArthur Midlife Research Network, and run a trained and tested machine learning algorithm on the data. At the end of the process, the machine learning module will have used the positive and negative affect experienced by the patients in the study to predict health indicators in the patient. In our example we will use the MIDUS 2 data set to predict a patient's Blood Serum Soluble IL6 Receptor and their Blood C-Reactive protein. The program has been written in python and uses TensorFlow to handle the machine learning component

Acquiring the Data

We are using the MIDUS data sets provided on the National Archive of Computerized data on Aging’s website. Searching the term “MIDUS” will give you all available data of the past MIDUS studies. Our program can accept three data sets common in the MIDUS studies, the biomarker project, cognitive project, and the daily diary project. To download these projects, click one of the links provided by the search. To download this dataset, click the Download button, and select Delimited in the dropdown that appears. Read the terms of use and accept, your download will begin immediately if you have made an account on the NACDA website.

Necessary Downloads

The project code can be downloaded on GitLab. The link to the project is here: https://git.ece.iastate.edu/sd/sdmay20-41/-/tree/master

The repository can be cloned onto a computer. To do this steps are provided by GitLab[9] in their help section titled: Cloning a Repository. Follow the steps for "Cloning a repository using the command line."

To run the program, several installations will be required on the computer being used. The following list enumerates these requirements:

- Python 3.7.4 or later
- TensorFlow 2.0.0
- Plotly 4.6.0
- PyQt5

If you are missing any of these downloads, most of them can be installed with the command "pip install" followed by the desired installation.

Preparing the Data

The files will be in a tab delimited format. The next thing the program will need are documents that will specify how the data will be read into the program. An example is shown below:

COLUMN_NAME	MIN_VALUE	MAX_VALUE	MISSING_VAL_PROCESSING_METHOD	INPUT_OR_OUTPUT
B2DC6	0	7	AVERAGE	INPUT
B2DC13	0	7	AVERAGE	INPUT
B2DC14	0	7	AVERAGE	INPUT
B2DC15	0	7	AVERAGE	INPUT
B2DC16	0	7	AVERAGE	INPUT
B2DC17	0	7	AVERAGE	INPUT
B2DC18	0	7	AVERAGE	INPUT
B4BIL6	0	23	THROWOUT	OUTPUT
B4BSIL6R	9781	102987	THROWOUT	OUTPUT

Table 6.3.1: Example Data

A similar document can be assembled in excel. The format of this document should match the one above. The first column specifies the name of the column being read. This should match the exact column name within the data set you are defining parameters for. The second column is the minimum value that might appear in this column. The minimum should not include the number that denotes missing or unanswered values. The maximum value is the same, it should be that maximum value that would appear in that column and it should not include any numbers that denote missing values. Missing values are denoted by different numbers depending on the column they are in. They indicate that a respondent to the survey did not answer that question or a response was unable to be obtained. How this data is handled can affect the results of the program. In the fourth column, we specify how we would like to address these values. "AVERAGE" in this cell indicates that if a missing value is found, it will be replaced with the average for the entire column. In this way, the mean of the data is preserved. If the cell is marked with "THROWOUT" the program will remove that entire row for the respondent when they failed to provide a response. This

is used when a column is necessary for analyzing the data and a missing response can not be accepted. After compiling these documents, save them and remember the file path as you will need that later.

Running the Program

When the project has successfully been downloaded and the data has been configured, the executable for running the program can be found in the project folder of the file. Clicking on the executable will run the program and display the GUI. Here you can select the files you downloaded from the MIDUS database.

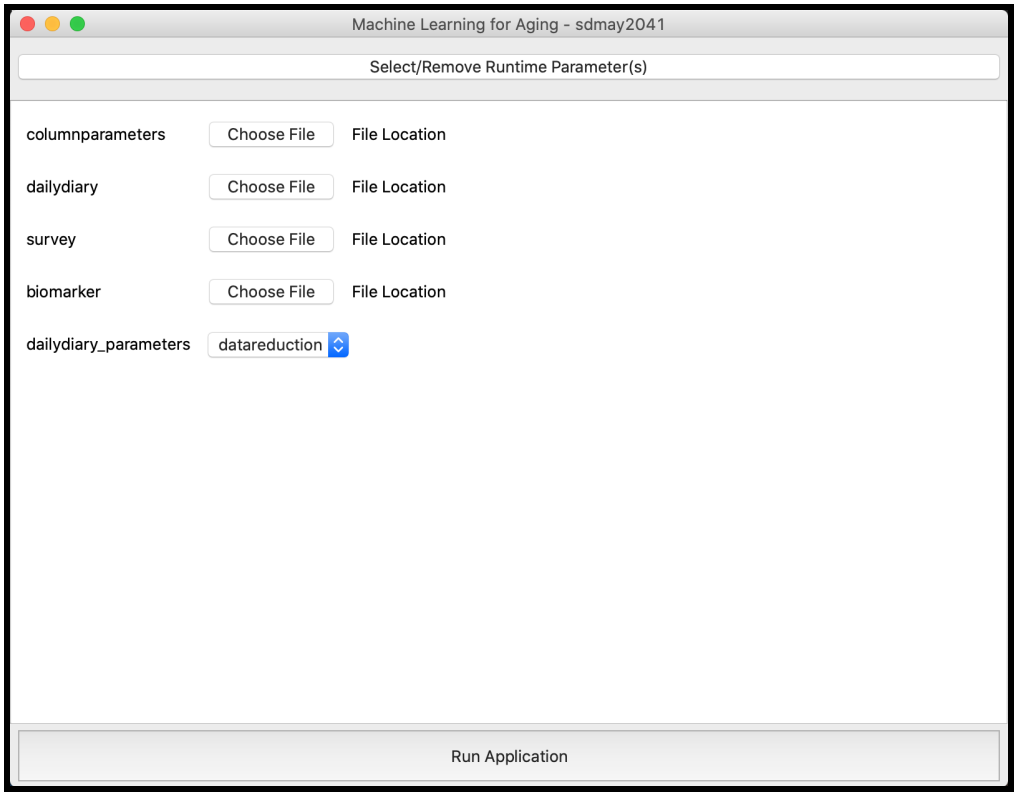

Figure 6.3.2: Program GUI

The daily diary dataset can have multiple lines for a single person, and so a processing method can be selected for this file. The flatten method is recommended for most runs of the program, as it will combine the rows into one line for the processing of the file. The data reduction parameter was created primarily for our example runs that analyzed the inflammation variables mentioned in the introduction. Users can define their own processing methods, and select them from this dropdown by editing the code yourself. On the main window, you can select the parameters that the program will run by clicking the button in the top right titled “Select/Remove Runtime Parameter(s)”. When you have finished the configuration, click Run Application on the bottom of the window and the program will begin, it may take some time to complete.

Interpreting the Results

The output of the program contains three files labeled “output.txt”, “reducedData.txt”, and “plotly_graph.html” which are written to the directory “Export”. The output.txt file contains the following information:

- Date and time the program was run
- Total time elapsed during the execution of the program in seconds
- Technique used to replace missing data
- Processing method used for the daily diary file
- TensorFlow Version
- Parameters entered by the user
- Names of the files used
- Column names included in the data
- Output logs which contain information about which users have all or no stressor days

The reducedData.txt file contains the data used to run the machine learning component after it has been reduced and processed by the program. The input data values will all be in the range [0, 1] because they have been normalized in preparation for the machine learning component. This file also contains the date and time of run at the top of the file.

The plotly_graph.html file contains a scatter plot graph displaying the mean squared error values over the course of the machine learning component’s runs. The mean squared error is the average squared difference between what we predicted the machine learning algorithm would output, and what it actually outputs in practice. There are 10 training sets that the machine learning algorithm is trained on, each set returning a mean squared error value. If the mean squared error values gradually tend towards zero then that could indicate that the algorithm is learning through experience.

Appendix II - Alternative Versions of Product

The original design of the product implemented a MySQL database to store the data parsed by the data extractor. However, our team decided that the database was unnecessary for the project. This is due to the fact that the machine learning module can directly input the data from the data extractor module. It also makes the product more extendable because users can enter MIDUS data sets of their choosing into the program and receive the results shortly.

Appendix III - System Test Description

Data Used

All data that has been used for this project has come from the official https://midus.colectica.org/ website. There are 3 files that are used by our python project.

MIDUS 2 Project 1 - 04652-0001-Data.tsv - Survey Data

MIDUS 2 Project 2 - 26841-0001-Data.tsv - Daily Diary

MIDUS 2 Project 4 - 29282-0001-Data.tsv - Biomarkers

These three files are Tab Separated Value files. From these files, 39 columns are used. 37 columns are used for input to the machine learning model and 2 are used for output. While 37 columns are used to create input data for our model, the number of input nodes is actually 11. This is because we use data reduction methods to combine 28 of those columns in two input values.

Input Values

There are two categories of input values. The first category is input values that do not require data reduction. In short, these values are just normalized and put into the model with no additional processing. There are 9 columns that fall into this category. Table 6.3.3 shows these columns. For these columns, if they were marked to be unanswered or unrecorded for that respondent, the value was replaced with the global average for that column.

Column Name	Column Description	File of Origin
B4H25	Do you engage in regular exercise, or activity, of any type for 20 minutes or more at least 3 times/week?	Biomarkers
B1PA39	Now smoke cigarettes regularly	Survey
B4S5	Rate sleep quality overall (past month)	Biomarkers
B1PA51	How often at least one drink (past mo)	Survey
B4QMA_D	MASQ: General Distress-Depressive Symptoms	Biomarkers
B4QTA_AX	Spielberger Trait Anxiety Inventory	Biomarkers
B1SNEURO	Neuroticism Personality Trait	Survey
B1SOPTIM	LOT: Optimism	Survey
B4QPS_PS	Perceived Stress Scale	Biomarkers

Table 6.3.3: Normalized Columns

The second type of input values is the input values that were involved in data reduction. These values were combined with other values to create a new column. 28 columns fall into this category. The 28 columns are described in Table 6.3.4 and Table 6.3.5. The only column that is not listed in these tables but was used in data reduction was B2DA_STR.

The goal of the data reduction done by these columns was to create two new values. These values would help determine the stress reactivity of a person. All of these values came from the Daily Diary file. This file contains data where respondents were asked a series of questions over the course of eight consecutive days. These questions included asking the participant whether or not the day was a “stressor day”, as well as a series of additional questions that described how the participant felt during these days. A column called B2DA_STR marked whether or not a respondent thought a day was stressful or not. If this column was marked with the number 8, meaning the respondent didn’t indicate whether or not the day was a stressor day, the corresponding row was

thrown out. This is done because we would not be able to determine whether or not the data should go into the stressor day average or the non stressor day average.

Column Name	Column Description
B2DC7	IN_GOOD_SPIRITS
B2DC8	CHEERFUL
B2DC9	EXTREMELY_HAPPY
B2DC10	CALM_AND_PEACEFUL
B2DC11	SATISFIED
B2DC12	FULL_OF_LIFE
B2DC21	CLOSE_TO_OTHERS
B2DC22	LIKE_YOU_BELONG
B2DC23	ENTHUSIASTIC
B2DC24	ATTENTIVE
B2DC25	PROUD
B2DC26	ACTIVE
B2DC27	CONFIDENT

Table 6.3.4: Positive Affectivity

Column Name	Column Description
B2DC1	RESTLESS_OR_FIDGETY
B2DC2	NERVOUS
B2DC3	WORTHLESS
B2DC4	SO_SAD_NOTHING_COULD_CHEER_YOU_UP
B2DC5	EVERYTHING_WAS_AN_EFFORT
B2DC6	HOPELESS
B2DC13	LONELY
B2DC14	AFRAID

B2DC15	JITTERY
B2DC16	IRRITABLE
B2DC17	ASHAMED
B2DC18	UPSET
B2DC19	ANGRY
B2DC20	FRUSTRATED

Table 6.3.5: Negative Affectivity

One of the values that was created was a person's Positive Affectivity (PA). This value was a measure of how the respondent's positive emotion state changed from stressor days to non-stressor days. There are 14 columns that were used to calculate a respondent's positive emotional feeling. These 14 columns are listed in Table 6.3.4. All of these columns are measured on a scale from 0 to 7. If any of these values were marked as missing or marked to indicate that the respondent didn't answer the question, the value was replaced with the global average for that column.

To calculate PA, a respondent's responses were divided into 2 categories, stressor day responses and non-stressor day responses. The average of the 14 columns were calculated for both stressor days and non-stressor days. PA was the absolute value of the difference between these two averages.

The second value that was created was a respondent's Negative Affectivity. This was created in the exact same way as PA, except the columns in Table 6.3.5 were used instead of the columns in Table 6.3.4.

In total, 11 columns were used as input for the machine learning model. The 9 columns that did not require data reduction, and the NA and PA columns.

Output Values

There are two values that are used as output values. The goal of the machine learning model is for the 11 input values listed above to be able to predict these two output values. Both of the values come from the Biomarkers file. These values are listed in Table 6.3.6.

Column Name	Description
B4BSIL6R	Blood Serum Soluble IL6 Receptor (pg/mL)
B4BCRP	Blood C-Reactive Protein (ug/mL)

Table 6.3.6: Output Values

Normalization

Before these values are put in the machine learning model, they are normalized to be between 0 and 1. A minimum value and a maximum value for each column is specified by the https://midus.colectica.org/ website. To normalize the value, the value is subtracted by that column's indicated minimum and then divided by the difference of the maximum and minimum. None of the values can be negative. The minimum for the NA and PA column is 0 and the maximum is 7. This is because the minimum and maximum of the columns that are used to calculate these columns are 0 and 7.

Randomization

Before these values are put in the machine learning model, they are put into a random order to eliminate any biases that could have been created when gathering the data.

Machine Learning Module

Once the data is normalized and randomized, it is passed to our machine learning component. Our model is a 3 layer model that uses back propagation. The first layer has 11 input nodes, our hidden layer has 44 nodes, and our last layer has 2 output nodes. The algorithm works by dividing the data into 10 parts. 1 part test and 9 parts training. The 9 parts of training data are then further subdivided into 10 parts. The algorithm is then trained on each of those 10 subsets once before tested on the original 1 testing part

Results

The loss of the program when down, which means that the machine learning model was able to use the input columns to predict inflammation. This is in line with what the study we were trying to recreate stated. As a result, the system test was a success. The results of the system test are shown in Figure 6.3.7.

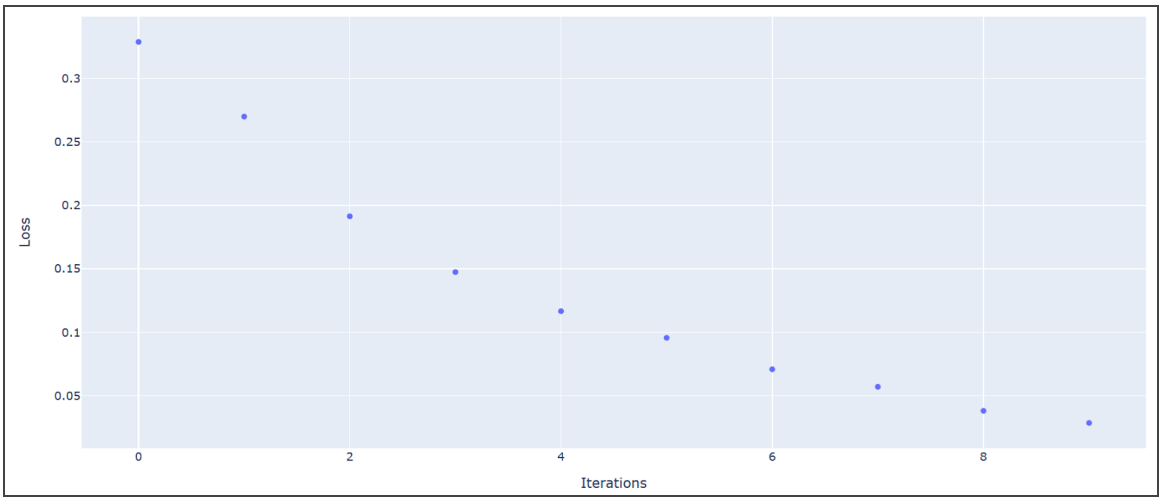

Figure 6.3.7: Machine Learning Loss