ARL-TR-9440 ● APR 2022

# Tools and Processes for Assessment of Blast Overpressure–Related Traumatic Brain Injury

by Dale Shires, Joseph Zirilli, Michael An, Robert Olsen, Michael Amato, Jacob Adams, and Katherine Breczinski

ARL-TR-9440 ● APR 2022

# Tools and Processes for Assessment of Blast Overpressure–Related Traumatic Brain Injury

by Dale Shires
*DEVCOM Army Research Laboratory*

Joseph Zirilli, Michael An, Robert Olsen, Michael Amato, and Jacob Adams
*Parsons, Inc.*

Katherine Breczinski
*Franklin and Marshall College*

| REPORT DOCUMENTATION PAGE | *Form Approved* *OMB No. 0704-0188* |
|---|---|

| | | |
|---|---|---|
| **1. REPORT DATE** *(DD-MM-YYYY)* April 2022 | **2. REPORT TYPE** Technical Report | **3. DATES COVERED (From - To)** April 2021–January 2022 |

| | |
|---|---|
| **4. TITLE AND SUBTITLE** Tools and Processes for Assessment of Blast Overpressure–Related Traumatic Brain Injury | **5a. CONTRACT NUMBER** |
| | **5b. GRANT NUMBER** |
| | **5c. PROGRAM ELEMENT NUMBER** |
| **6. AUTHOR(S)** Dale Shires, Joseph Zirilli, Michael An, Robert Olsen, Michael Amato, Jacob Adams, and Katherine Breczinski | **5d. PROJECT NUMBER** |
| | **5e. TASK NUMBER** |
| | **5f. WORK UNIT NUMBER** |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** DEVCOM Army Research Laboratory ATTN: FCDD-RLC-S Aberdeen Proving Ground, MD 21005-5066 | **8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-9440 |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** | **10. SPONSOR/MONITOR'S ACRONYM(S)** |
| | **11. SPONSOR/MONITOR'S REPORT NUMBER(S)** |

**12. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

**13. SUPPLEMENTARY NOTES**
primary author's email: <dale.r.shires.civ@army.mil>.

**14. ABSTRACT**
The blast overpressure studies (BOS) effort is attempting to understand and mitigate the effects of blast overpressure events for the military, and in particular address issues related to traumatic brain injury. The goal of this particular project was to better understand data analysis required to improve operational medicine for the military in terms of BOS. Also of interest was how this data should be cataloged and made available to researchers pursuing this topic. Assessed data generally consists of concentrations of blood biomarkers present, Defense Automated Neurobehavioral Assessment metrics, and self-reported symptoms by the participants. Several Python tools were used to reproduce the analyses present in the literature, including Spearman's rank correlation, analysis of variance, and interquartile range assessments. Additional methods of analysis common in machine learning were also explored to see if useful correlations could be found. This work shows the advantages of having a structured approach for data storage and access, as well as highlighting the ability for common open source tools such as Python and various statistical and machine learning packages to produce meaningful results for analysis.

**15. SUBJECT TERMS**
traumatic brain injury, symptomatology, TBI, outlier detection, overpressure, Biological and Biotechnology Sciences

| **16. SECURITY CLASSIFICATION OF:** | | | **17. LIMITATION OF ABSTRACT** | **18. NUMBER OF PAGES** | **19a. NAME OF RESPONSIBLE PERSON** Dale Shires |
|---|---|---|---|---|---|
| **a. REPORT** Not Secret | **b. ABSTRACT** Unclassified | **c. THIS PAGE** Unclassified | UU | 55 | **19b. TELEPHONE NUMBER (Include area code)** 410-278-5006 |

**Standard Form 298 (Rev. 8/98)**
Prescribed by ANSI Std. Z39.18

## Contents

## List of Figures

## List of Tables

## Acknowledgments

The authors wish to thank the Department of Defense (DoD) High Performance Computing Modernization Program and the DEVCOM Army Research Laboratory DoD Supercomputing Resource Center (DSRC) for a grant of computing time and resources to perform this research. We also thank the researchers at the Blast Induced Neurotrama (BINT) group at Walter Reed Army Institute of Research (WRAIR) for providing additional data and answering numerous questions about these data sets.

## 1. Introduction

Overpressure events generate pressure waves exceeding atmospheric pressure. Exposure to blast overpressure can have severe, cumulative, and long-term negative effects on people and their cognitive health. For example, Soldiers involved in breaching events, such as those that use explosives to punch holes into structures like walls or fences, are commonly exposed to overpressure. Soldiers proximate to weapon systems can also experience overpressure. Quite often exposure to overpressure events can lead to symptoms similar to those experienced with concussion or mild traumatic brain injury (mTBI).

First, due to the variability and uncertainty when it comes to reporting symptoms of mTBI or concussion, as well as diagnosing this injury, it is necessary to develop and explore analyses to make this task easier. The first goal of this project was to create a single platform where users can load, edit, and analyze data. Such a system that can be referenced as a source of truth will increase research reproducibility by reducing mistakes that can happen when using multiple platforms or when transferring data between groups of researchers. Further, having the data close to high-performance computing resources and being able to easily use them for large analysis workflows would facilitate new kinds of research previously infeasible. We developed a software stack that contains an ingest workflow, data store, data processing engine, and a user interface (UI), using the lakehouse architecture.[1] Once data is ingested, it can later be analyzed using a combination of modalities, including standard query language (SQL)-based exploration, Python-based exploration, and artificial intelligence(AI)/machine learning (ML) algorithms. Additionally, all iterations, changes, and newly derived data are recorded in the repository, so it is easy to see how the data may have been changed. Attribution and data provenance are enhanced in this model.

Secondly, during the course of this research, the topic of outlier detection and removal became rather important. There are numerous methods that have been studied and implemented over the years. Numerous papers published in the human health arena use an automated method that takes human judgment of outlier identification out of the process. This method is available in a commercial package, and the overall approach is discussed in a paper by the algorithm's author.[2] We implemented this approach in Python because our research indicated it had not yet been implemented in this computer language. Having this approach available also allows for reproducibility of research and application of this method on large-scale supercomputers.

Finally, the last main goal of this project was to show that Python and additional statistical and ML methodology can be used to reproduce and expand upon analysis common in this field. By building on established models and procedures developed and tested in the data sciences community, time to solution can be relatively fast with abundant customization along the way.

In order to better understand the types of processing and analysis done in this area of research and to determine what tools and procedures to apply and/or develop to do this analysis, we focused on two different data sets. The first comprises a total of 29 subjects and has blood biomarker data, human neurocognitive performance data, and blast pressure data.[3] This was the first data set that we analyzed and focused on using traditional techniques from the mTBI community. This analysis also highlighted the importance of outlier identification and removal, which is even more critical for small data sets. As mentioned, this necessitated a specialized method that is available commercially, but not yet available or implemented in a more generic software approach to the best of our knowledge. This approach is described in this report.

Following this, more data including 218 subjects from 8 cohorts was made available to us through a partnership with Walter Reed Army Institute of Research (WRAIR) that allowed us to focus more on emerging data science approaches like deep neural networks to look for patterns and causation. Our experiences with this data set compose the latter part of this narrative. The sum of these methods provides a way to use open source tools and techniques to allow for customized and in-depth data science processing for mTBI research.

## 2. Statistical and Preliminary Data Science Methods for mTBI

Our first task was to recreate and possibly expand upon inquiry avenues conducted in the mTBI community. Through initial contacts with WRAIR and subsequent literature reviews, we focused on a document by Boutté et al.[3] that provided spreadsheets with the data used in their study. The contents of these files helped to solidify ideas on how the data should be structured in customized data stores and led to the overall process for data ingest, storage, and access that is described in Section 3.1. Since this data was fairly straightforward and housed in limited spreadsheets, we chose to convert this data to comma separated value (CSV) files that served as the basis for the remainder of the processing. We treated this data as historical and made no attempt to fully recreate any analysis. For example, we did not possess full knowledge of any special circumstances regarding things like biomarker collection and scoring. Accordingly, our main goal here was only roughly to correspond to the analysis already done and not completely match results.

## 2.1 Analytical Tools

For analyses, we focused on tools that are supported on everyday computing systems and also massive computing assets that could be used to speed analysis and/or allow for processing of larger data sets. We also focused on tools that are open source and would require only a beginner's level of familiarity with the Python programming language since this language is a common entry point for data scientists who need to write programs but might not be seasoned coders.

Numerous open source methods, tools, and libraries were used,[4] including the following:

- Jupyter notebooks: Notebooks execute in web browsers and provide users a way to execute "cells" of code as needed rather than having to possibly re-run entire programs.
- Pandas: Provides easy-to-use data structures and data analysis tools. Pandas greatly simplifies the process of extracting, transforming, and loading (ETL) data.
- NumPy: Provides high-performance array data structures.
- SciPy: Contains numerous mathematical routines for linear algebra and curve fitting.
- Matplotlib: Visualization routines with simplified calls for customization.
- Seaborn: Builds upon Matplotlib to provide more complex visualization methods.

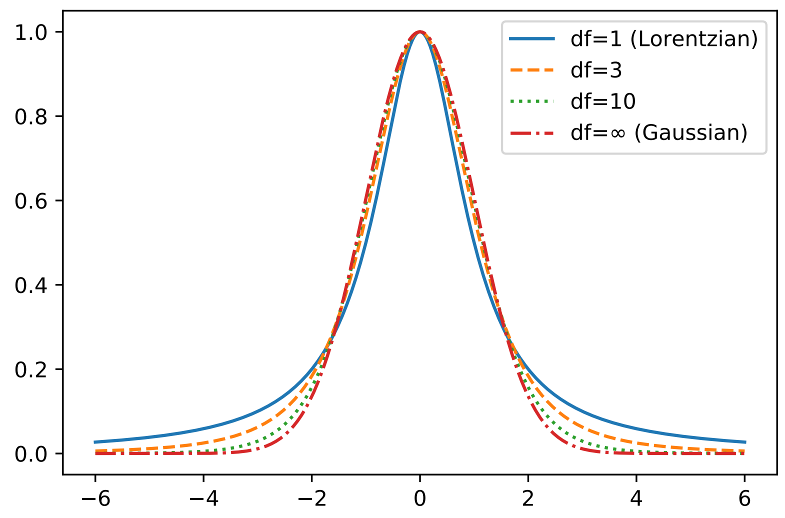


**Fig. 1 Normalized t probability for differing degrees of freedom**

- Scikit-Learn: Basic ML tools such as clustering, $k$-means, random forests, and so forth.

## 2.2 Identifying Outliers

A key aspect of any data analysis is determining which data points might be outliers. There are numerous methods to perform this task, including analyzing z-scores, Chi-square values, and so forth. We chose to implement a method used that somewhat automates the process, uses a nonlinear regression method, and has a low false positive rate.[2] This method is available in a commercial package, but did not seem to be implemented in Python based on our searches. It is also the method used to remove outliers in the Boutté et al.[3] paper we were studying. The process, known as ROUT for robust regression and outlier removal, is based on nonlinear regression and uses a concept where the residuals follow a Lorentzian rather than Gaussian distribution. The difference between these two curves can be seen in Fig. 1. Various degrees of freedom are shown with the Lorentzian (or Cauchy) curve where df = 1 and the more familiar Gaussian curve where df = $\infty$.

ROUT uses the Levenberg–Marquardt (LM) method to do curve fitting but makes several modifications. These include a different Lorentzian merit function, partial derivative vector, and computation of a robust standard deviation of the residuals. The method also requires a customized approach to determining if the iterative fit is improving. Since it was not possible to alter the built-in methods available in the Python libraries to perform these actions, we needed to implement a version with source code that would allow for modifications. We were able to get a start on this by modifying a C language algorithm to compute LM.[5] To test this implementation, we focused on a worked example from the ROUT paper[2] that uses a one-phase exponential decay based on signal values at various times shown in Table 1.

**Table 1 Example data points following a roughly exponential decay curve**

| Time | Signal |
|---|---|
| 0.0 | 1105.7 |
| 1.0 | 550.6 |
| 2.0 | 566.6 |
| 3.0 | 136.0 |
| 4.0 | 440.7 |
| 5.0 | 329.6 |
| 6.0 | 302.6 |
| 7.0 | 220.8 |
| 8.0 | 136.9 |
| 9.0 | 20.3 |
| 10.0 | 48.0 |
| 11.0 | –1.0 |
| 12.0 | 68.3 |

The fitting function equation for a one-phase exponential decay is given in Eq. 1. Here $a$ is the y value at time zero, $b$ is the plateau, and $c$ is the rate constant. The gradient vector for this function is given in Eq. 2.

$$y = (a - b)e^{-cx} + b \tag{1}$$

$$\nabla f(a,b,c) = \begin{bmatrix} \frac{\partial}{\partial a}((a-b)e^{-cx}+b) \\ \frac{\partial}{\partial b}((a-b)e^{-cx}+b) \\ \frac{\partial}{\partial c}((a-b)e^{-cx}+b) \end{bmatrix} = \begin{bmatrix} e^{-cx} \\ -e^{-cx}+1 \\ (-a+b)e^{-cx}x \end{bmatrix} \tag{2}$$

For the models in question, we never experienced a system that failed to converge. Nonetheless, we decided to use the scipy curve_fit function to compute the Gaussian coefficients and then supplied those as p0, the initial guess to the system. For a Gaussian least squares fit, the final coefficients were determined to be [1005.22, 73.66, 0.373], which served as p0. A plot of the individual data points along with this least squares fit can be seen in Fig. 2. For the ROUT method, the value for a parameter known as "Q" must be set. This parameter is related to the false discovery rate (FDR) and can be set to various percentages. Setting the value of Q to be 1% means that you can expect less than 1% of the significant findings to be false positives while the rest are real. We used 5% and 1% (the suggested value for outlier identification), and computed the results ordered by the absolute value of the residuals (Table 2). Our values were closely aligned with those reported in the ROUT paper.[2]

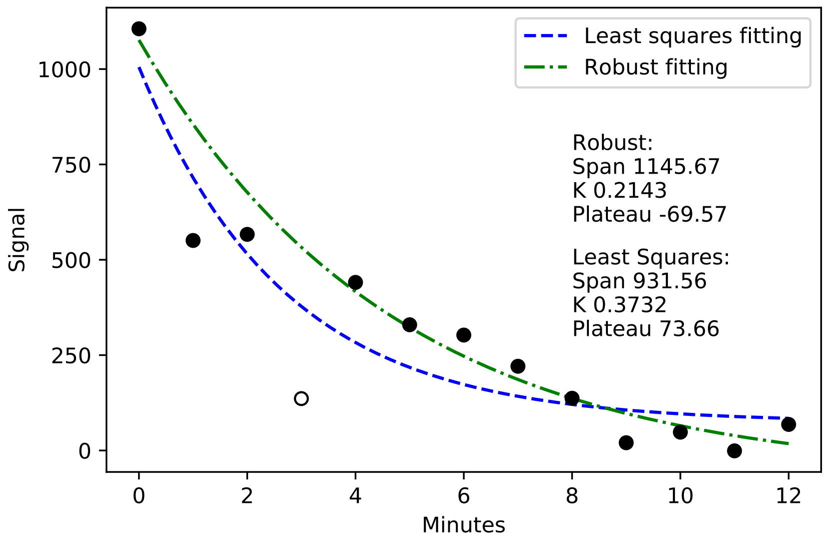


**Fig. 2 Exponential decay fitted to various points using least squares and robust fitting. The point at time 3 has been flagged as an outlier.**

Following a robust fit using the Lorentzian distribution (coefficients = [1076.10, -69.57, 0.214]), ROUT proceeds in an iterative fashion by examining the top 30% of points away from the curve (highest residuals) and checking p-values against the threshold for the set value of Q. Those points where the p-value is less than the threshold are flagged as outliers. In the case of the exponential decay graph, the point at time 3 is shown as the outlier along with the modified robust fit curve. This curve pays less attention to this presumed outlier during the fit. The p-value for this point (0.0004) is less than the threshold (0.0008) and is hence flagged as an outlier.

Following this testing, we developed the functions required to implement a standard

**Table 2 Important values from the worked example. At time 3, it can be seen that the p-value drops below the Q value threshold of 1% and hence this point is flagged as an outlier.**

| Time | Residual | t ratio | P-value | Threshold Q=5% | Threshold Q=1% |
|---|---|---|---|---|---|
| 8 | 0.19 | 0.00 | 0.9981 | 0.0500 | 0.0100 |
| 5 | 6.81 | 0.09 | 0.9317 | 0.0462 | 0.0092 |
| 10 | –16.80 | 0.22 | 0.8326 | 0.0423 | 0.0085 |
| 4 | 24.13 | 0.31 | 0.7617 | 0.0385 | 0.0077 |
| 0 | 29.60 | 0.38 | 0.7102 | 0.0346 | 0.0069 |
| 7 | 34.79 | 0.45 | 0.6628 | 0.0308 | 0.0062 |
| 11 | –39.88 | 0.52 | 0.6177 | 0.0269 | 0.0054 |
| 12 | 50.34 | 0.65 | 0.5302 | 0.0231 | 0.0046 |
| 6 | 55.50 | 0.72 | 0.4899 | 0.0192 | 0.0038 |
| 9 | –76.62 | 0.99 | 0.3457 | 0.0154 | 0.0031 |
| 2 | –110.12 | 1.42 | 0.1854 | 0.0115 | 0.0023 |
| 1 | –304.50 | 3.93 | 0.0028 | 0.0077 | 0.0015 |
| 3 | –396.76 | 5.12 | 0.0004 | 0.0038 | 0.0008 |

nonlinear curve fitting procedure to apply to the medical data being studied. This familiar function is given in Eq. 3 along with the gradient vector in Eq. 4.

$$y = ax^2 + bx + c \tag{3}$$

$$\nabla f(a,b,c) = \begin{bmatrix} \frac{\partial}{\partial a}(ax^2+bx+c) \\ \frac{\partial}{\partial b}(ax^2+bx+c) \\ \frac{\partial}{\partial c}(ax^2+bx+c) \end{bmatrix} = \begin{bmatrix} x^2 \\ x \\ 1 \end{bmatrix} \tag{4}$$

Human subject data can be highly variable. Hence the application of an automated outlier removal system is highly desirable. To illustrate the application of ROUT to the data, consider the case of the delta simple response time (SRT) scores for the Defense Automated Behavioral Assessment (DANA) that are part of the data set we examined. (This test is explained in more detail in Section 2.3.3.) The scatter plot of the x-y pairs is shown in Fig. 3 and the table of relevant ROUT values (for the top 30% of the points away from the curve) is in Table 3. In this case, none of the points qualified as outliers, but the shapes of the least squares fit and robust fit curves are interesting nonetheless. The top three contenders as possible outliers are distinctly "pulling" the curve in their directions for least squares fitting, but are ignored more with the robust fit.

The modified LM algorithm generates a robust mean of the data when supplied

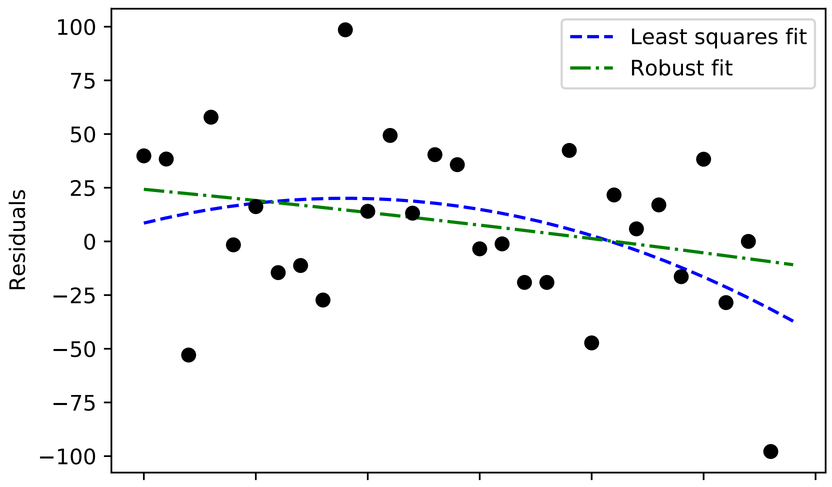


**Fig. 3** **Delta DANA SRT values plotted against their respective residuals. None of the points qualified as outliers and hence none were removed from the data set.**

**Table 3** **Delta DANA SRT values for the ROUT algorithm. None of the points have a p-value less than their threshold, hence none are classified as outliers.**

| X | Residual | t ratio | P-value | Threshold Q=1% |
|---|---|---|---|---|
| 3 | 36.66 | 0.95 | 0.3497 | 0.0031 |
| 11 | 36.99 | 0.96 | 0.3453 | 0.0028 |
| 19 | 39.76 | 1.03 | 0.3111 | 0.0024 |
| 8 | 43.07 | 1.12 | 0.2733 | 0.0021 |
| 25 | 43.58 | 1.13 | 0.2678 | 0.0017 |
| 20 | 48.59 | 1.26 | 0.2180 | 0.0014 |
| 2 | 75.15 | 1.95 | 0.0617 | 0.0010 |
| 9 | 83.90 | 2.18 | 0.0385 | 0.0007 |
| 28 | 88.40 | 2.30 | 0.0299 | 0.0003 |

with the function and gradient listed in Eqs. 5 and 6. When we have nothing but the delta values, this function is probably the best to supply ROUT and is graphically easy to comprehend with just a simple scatter plot and horizontal line showing the computed robust mean from ROUT. This style of chart for the delta SRT values is shown in Fig. 4.

$$y = x * 0 + a \tag{5}$$

$$\nabla f(a) = \left[\frac{\partial}{\partial a}(x * 0 + a)\right] = \left[1\right] \tag{6}$$

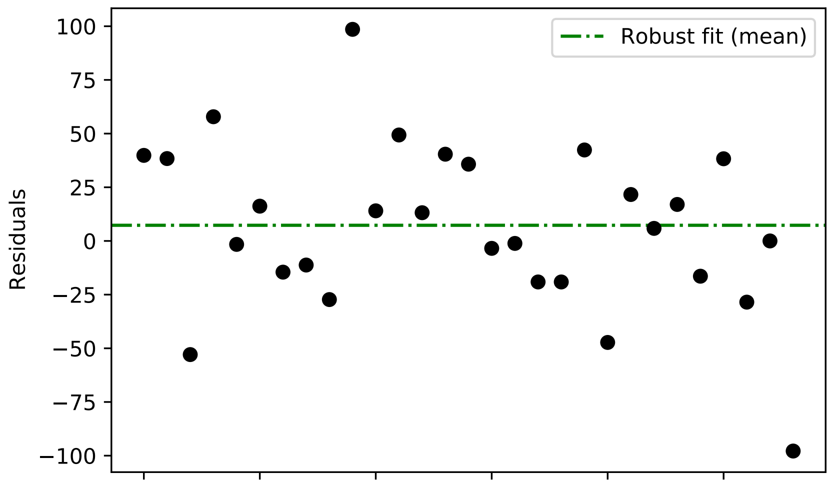


**Fig. 4 A scatter plot of the delta DANA SRT values shown along with the horizontal line of the computed robust mean. None of the points qualified as outliers and hence none were removed from the data set.**

## 2.3 Characteristics of the Data

### 2.3.1 Participants

The data involves 29 (n = 29) active duty United States Army personnel. Data was captured during explosives training exercises known as breaching events and was provided as a set of three spreadsheet files, each containing either data or a summary of the data. These files were converted to CSV files that allow for easy parsing by Python and moved to one of the supercomputers for processing. One change was made to the data files. Two subjects had entries recorded as “ND,” which we assume means no data in their post exposure questionnaire. To simplify processing and not have to remove these subjects from the pool, we replaced these values with a zero (0) to match their response from before testing. Other than that, no changes were made to the data. The application of the outlier removal method ROUT was used on this data “as-is.” A summary of the data can be seen in Table 4.

**Table 4 Summary demographics and data characteristics for the 29 participants (ages ranging from 21 to 43), blast overpressures with standard deviations (psi and kPa), and impulse with standard deviations (psi and kPa)**

| **Subject and blast exposure characteristics** | | |
|---|---|---|
| Total number of subjects (n) | 29 | |
| Age (years) | | |
| Mean (SD) | 29.0 (5.08) | |
| Range [Min-Max] | 21-43 | |
| Peak Pressure | psi | kPa |
| Per Incident Mean (SD) | 4.35 (0.49) | 30.01 (3.40) |
| Cumulative Mean (SD) | 8.71 (0.99) | 60.03 (6.80) |
| Impulse | psi $\times$ time (milliseconds) | |
| Per Incident Mean (SD) | 11.74 (1.13) | |
| Cumulative Mean (SD) | 23.48 (2.26) | |

### 2.3.2 Self-reporting for mTBI/Concussion

Self-reporting and assessment is a common way to gauge subject brain health. This symptomatology was gathered by giving the study participants questionnaires to complete, once before (pre-) the explosive training events, and a second time after (post-) completing the breaching exercises. The response choices were 0 - "Not experienced at all," 1 - "No more of a problem than before training started," 2 - "A mild problem–present but don't really notice and doesn't concern me," 3 - "A moderate problem–I can continue what I am doing but I notice the problem," and 4 - "A severe problem–constantly present, feels like it could affect my performance." The questions were similar to but more numerous and tailored than the ones commonly seen in the Rivermead questionnaire.[6] These questions and how the respondents reported are shown in Table 5. The symptoms have been sorted from highest to lowest based on the symptoms that increased by the larger percentages. This type of data was used for the development and testing of neural networks that is discussed later in this report.

### 2.3.3 Neurocognitive Performance

While the results from self-reporting are subjective, a battery of tests was also given to the participants both before and after breaching event exposure in an effort to be more quantitative. The DANA tool was administered 8 h before and 1 h after blood tests (see Section 2.3.4) were drawn from the participants. DANA runs on a handheld device and consisted of three tests designed to gauge different cognitive functions.[7] The first test was the SRT test where subjects had to tap on the screen target as quickly as possible. Test two was the procedural reaction time (PRT) test

**Table 5 Changes (post- vs. pre-event) in self-reported symptomatology for participants in the breaching exercises**

| Symptom | Increase | No Change | Decrease |
|---|---|---|---|
| Headaches | 15 (52%) | 13 (45%) | 1 (3%) |
| Taking longer to think | 12 (41%) | 17 (59%) | 0 (0%) |
| Feelings of dizziness | 9 (31%) | 19 (66%) | 1 (3%) |
| Slowed thinking | 8 (28%) | 21 (72%) | 0 (0%) |
| Poor concentration | 8 (28%) | 20 (69%) | 1 (3%) |
| Feeling anxious or tense | 5 (17%) | 24 (83%) | 0 (0%) |
| Blurred vision | 5 (17%) | 21 (72%) | 3 (10%) |
| Ringing in ears | 5 (17%) | 19 (66%) | 5 (17%) |
| Being irritable or easily angered | 4 (14%) | 24 (83%) | 1 (3%) |
| Easily upset by loud noise | 4 (14%) | 23 (79%) | 2 (7%) |
| Feeling frustrated or impatient | 3 (10%) | 26 (90%) | 0 (0%) |
| Light-headedness | 3 (10%) | 25 (86%) | 1 (3%) |
| Fatigue; tiring more easily | 3 (10%) | 23 (79%) | 3 (10%) |
| Pain in ears | 2 (7%) | 27 (93%) | 0 (0%) |
| Poor coordination/clumsiness | 2 (7%) | 27 (93%) | 0 (0%) |
| Difficulty localizing sound source | 2 (7%) | 26 (90%) | 1 (3%) |
| Difficulty getting organized | 2 (7%) | 26 (90%) | 1 (3%) |
| Forgetfulness; poor memory | 2 (7%) | 24 (83%) | 3 (10%) |
| Easily upset by bright lights | 1 (3%) | 28 (97%) | 0 (0%) |
| Fullness in ears | 1 (3%) | 28 (97%) | 0 (0%) |
| Loss of balance | 1 (3%) | 28 (97%) | 0 (0%) |
| Easily overwhelmed by things | 1 (3%) | 28 (97%) | 0 (0%) |
| Feeling disoriented | 1 (3%) | 28 (97%) | 0 (0%) |
| Numbness or tingling in body parts | 1 (3%) | 28 (97%) | 0 (0%) |
| Difficulty making decisions | 1 (3%) | 27 (93%) | 1 (3%) |
| Restlessness | 1 (3%) | 26 (90%) | 2 (7%) |
| Sleep disturbance | 1 (3%) | 24 (83%) | 4 (14%) |
| Feeling depressed or sad | 0 (0%) | 29 (100%) | 0 (0%) |
| Change in taste/smell | 0 (0%) | 29 (100%) | 0 (0%) |
| Nausea and/or vomiting | 0 (0%) | 28 (97%) | 1 (3%) |
| Double vision | 0 (0%) | 28 (97%) | 1 (3%) |
| Loss of or increased appetite | 0 (0%) | 28 (97%) | 1 (3%) |

where a numerical digit was displayed for a brief time with subjects then required to remember and identify the displayed number. The third test was the Go-no-Go (GNG) test that is a forced reaction time test. "Friends" and "foes" are presented to the subjects and they are instructed to only press a "Fire" button when a foe appears. Attention and impulsivity are measured by this test. A more thorough treatment of DANA usage and analysis is available.[8]

Prior to any in-depth analysis, we thought it prudent to try to understand the data by way of visualization techniques. One way to do this is a pair-wise plot to see general relationships between the data. Such a plot is shown in Fig. 5. The figure, which shows the changes in reaction times from before and after with the three DANA tests, was generated before any outlier removal and analysis to visually identify what could be outliers, and this approach also helps in visually picking up on trends of interaction between variables. The diagonal represents a histogram of the individual data element.

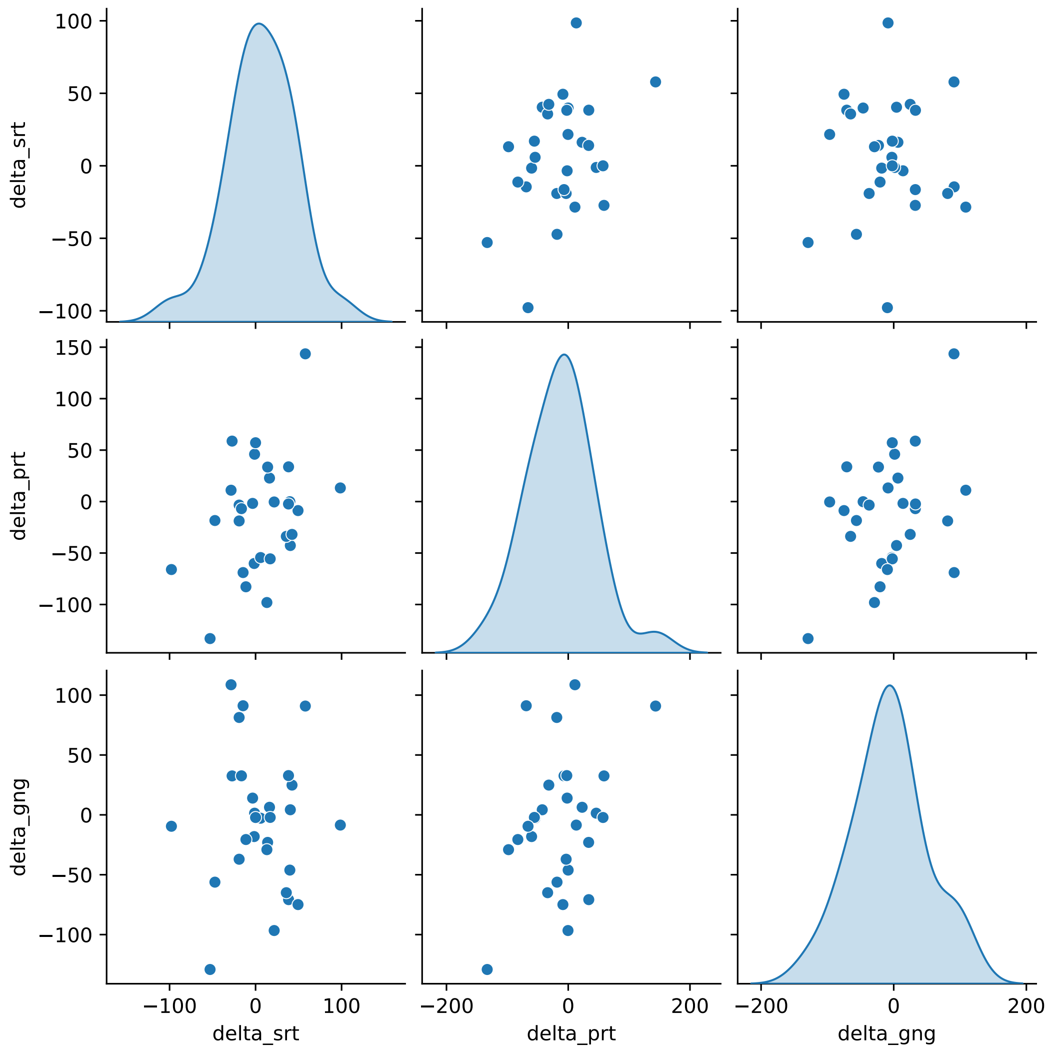


**Fig. 5 Pair-wise plot of DANA metrics**

Neurocognitive performance data was converted into delta values (difference in time taken after exposure versus before exposure) and ROUT (Q = 1%) applied to identify and remove outliers. The remaining values were checked for normality using both the Shapiro–Wilk test and the D'Agostino's k-squared test. Significance was tested using one-sided Wilcoxon tests. There are several ways to report the interquartile range (IQR).[9] We use an interpolation technique for when the IQR does not exactly fall on an experimental value. Boutté et al.[3] noted using the Graphpad Prism tool[10] for their analysis, and this tool seems to use this interpolation approach as well. This behavior is available in Python with a newer version of NumPy (version 1.22), where there is an option to pick how this range is computed. Supplying the optional argument “weibull” to the NumPy percentile function selects a continuous result and provides this approach. The p-values and the IQRs for the data are shown in Table 6.

**Table 6 Values for the DANA assessment**

| DANA (ms) | Pre-Exposure | | Post-Exposure | | % Change | p-value |
|---|---|---|---|---|---|---|
| | Median | IQR | Median | IQR | | |
| SRT | 268.95 | 256.63–298.41 | 290.58 | 263.76–343.47 | 8.04 | 0.164 |
| PRT | 636.88 | 592.94–672.14 | 628.03 | 587.67–680.20 | –1.39 | 0.066 |
| GNG | 601.33 | 553.97–646.23 | 589.73 | 561.13–661.80 | –1.93 | 0.241 |

### 2.3.4 Blood Biomarkers

The US Food and Drug Administration has been approving certain blood tests to assist in the identification of mTBI injuries.[11] The samples in the data set consist of nervous-system related proteins including glial fibrillary acidic protein (GFAP), ubiquitin carboxy-terminal hydrolase L1 (UCH-L1), neurofilament light chain (Nf-L), Tau, and amyloid beta peptides Aβ-40 and Aβ-42. In the data set, UCH-L1 had numerous values recorded as 0.0, and based on the IQR reported in the paper we were reviewing,[3] probably were removed from the sample. However, since we did not possess any background data on this, we left the data in as-is. Also, UCH-L1 failed normality tests. Accordingly, we chose to implement a two-sided Wilcoxon test making no assumptions to the direction of the data trends. Given all of this, any results for our processing of UCH-L1 should be fairly inconclusive. A pair plot of this data is available in Fig. 6 and basic data characteristics shown in Table 7.

**Table 7 Values for the blood biomarker assessment. UCH did not follow a normal distribution and was tested with a two-sided Wilcoxon test.**

| Biomarker (pg/mL) | Pre-Exposure | | Post-Exposure | | % Change | p-value |
|---|---|---|---|---|---|---|
| GFAP | 59.20 | 45.25–70.20 | 52.10 | 43.95–70.20 | –11.99 | 0.051 |
| UCH-L1 | 0.00 | 0.00–3.08 | 0.00 | 0.00–4.04 | 0.00 | 0.841 |
| Nf-L | 4.88 | 3.75–6.59 | 5.22 | 3.72–6.81 | 6.97 | 0.091 |
| Tau | 0.07 | 0.00–0.22 | 0.12 | 0.00–0.23 | 71.43 | 0.071 |
| Aβ-40 | 126.00 | 92.05–160.00 | 140.00 | 104.50–164.50 | 11.11 | 0.147 |
| Aβ-42 | 5.09 | 2.46–6.55 | 5.19 | 3.79–7.01 | 1.96 | 0.045* |

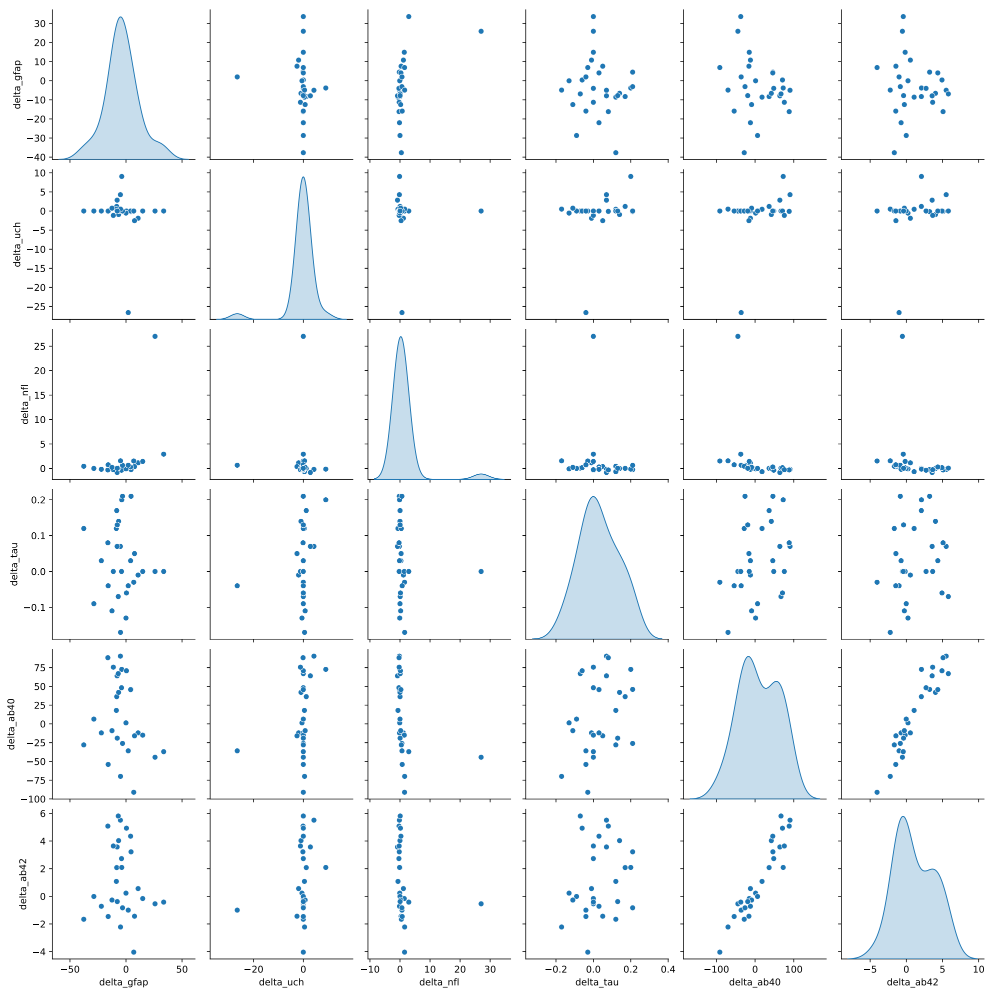


**Fig. 6 Pair-wise plot of blood biomarker metrics**

## 2.4 Analysis

Analysis proceeds by looking at the various ways to pair data and look for patterns, both hidden and obvious. Unsupervised learning approaches such as clustering did not seem overly appropriate for this data set as most of the data was labeled and too small to extract any significant meaning from something like $k$-means clustering. Even though we performed $k$-means clustering with Scikit-learn, we do not report those results here.

### 2.4.1 Neurocognitive Performance

The Seaborn library was used to produce a visualization of the blood biomarkers and DANA metrics data. The data used for this portion was the difference between the post- and pre-measures, or the delta subset of the data, which was stored in a Pandas DataFrame. The Seaborn pair-plot function displayed scatter plots of the data to visualize the relationships between the biomarkers as well as the distribution of the data and spot any potential outliers. Figure 7 shows the pair-plot of the blood biomarkers and DANA metrics. Although the axis labels are somewhat indiscernible in these charts, they are included for illustrative purposes only to show how they build on the histogram and scatter plots and combine them into one. The histogram can be seen on the diagonal and shows the distribution of a single variable. The scatter plots in the rest of the diagram show the relationships between all pairings of two variable combinations. These charts can provide a quick visual assessment of interactions that might be taking place.

The SciPy.stats library was used to perform one-sided Spearman's rank correlation. This library has a spearmanr function that can be used for performing a two-sided or one-sided t-test and also calculates the corresponding p-value for the correlation. This function was used to calculate Spearman's rank of the delta subset of the blood biomarkers and DANA metrics. The results can help determine the strength of the correlation between the data sets. To display the results from the Spearman's rank correlation, a Pandas DataFrame was used. To reproduce the median and interquartile ranges of the blood biomarkers and DANA metrics, the NumPy and statistics libraries were used. The statistics library has a median function, and the NumPy library has a percentile function to calculate Q1 and Q3. To calculate the iIQRs outright, SciPy stats has an IQR function. Both the pre- and post-measures were analyzed. These results were also displayed using a Pandas DataFrame.

Heatmaps were also produced using the Seaborn library to visualize the data. Both the blood biomarker data and DANA metrics data were used for the heatmap. These figures can be helpful, especially in dense data graphs, for rapidly calling the eye to important areas of data by shading the scalar values being studied. Figure 7 shows the heatmap of the blood biomarkers with the DANA metrics data. In Fig. 8, we show a heat map of the correlation (p-values) for the delta blood biomarkers. Low p-values are shaded significantly darker to highlight them. For more detailed analysis of biomarkers as indicators, readers should consult the analysis by Boutté et al.[3]

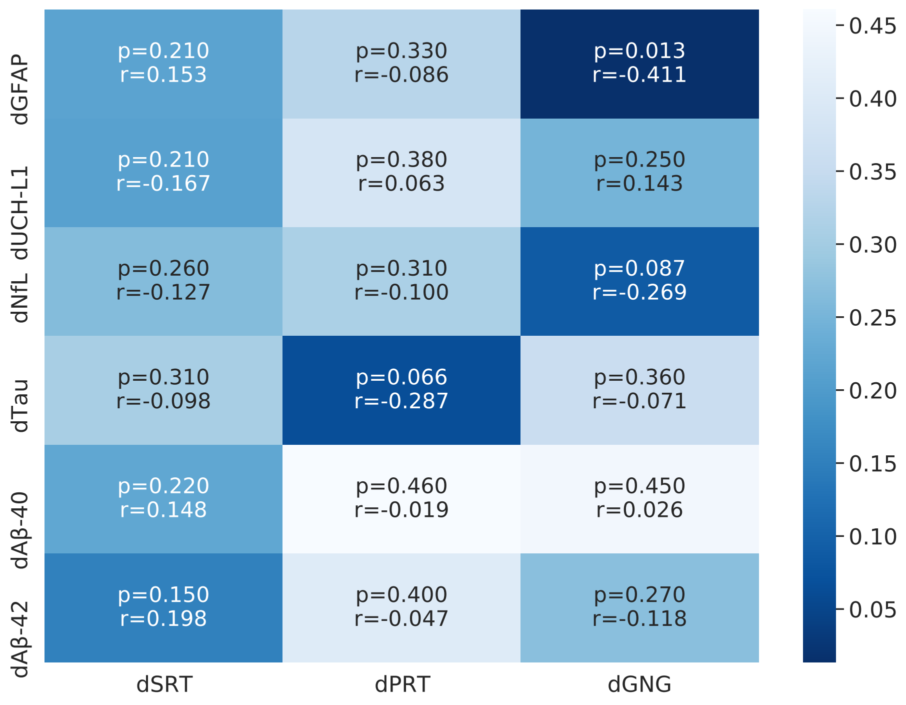


**Fig. 7** Heat map of the blood biomarkers and DANA metrics. Color shading is based on the p-value with darker shades representing areas of higher significance.

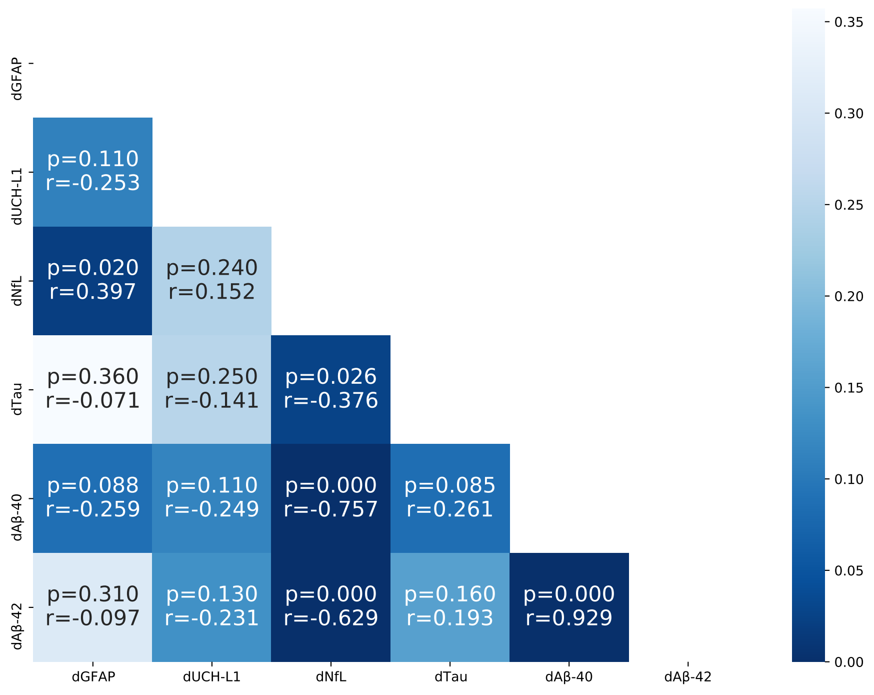


**Fig. 8** Heat map of the correlation between blood biomarker changes. Color shading is based on the p-value with darker shades representing areas of higher significance.

### 2.4.2 Self-reporting and Blood Biomarkers

To illustrate the box and whisker charts in Seaborn, we also looked at the relationship of the blood biomarkers with symptomology. We only look at one relationship: the change in GFAP in regard to increased levels of dizziness. Here there is a correlation where elevated levels of Aβ-40 are associated with an increase in dizziness after breaching events (decrease or no change: n = 20, range = –91.1 to 88.2, median = –13.5; increase: n = 9, range = –44.4 to 90.3, median = 48.2; p = 0.038). The box and whisker chart is shown in Fig. 9.

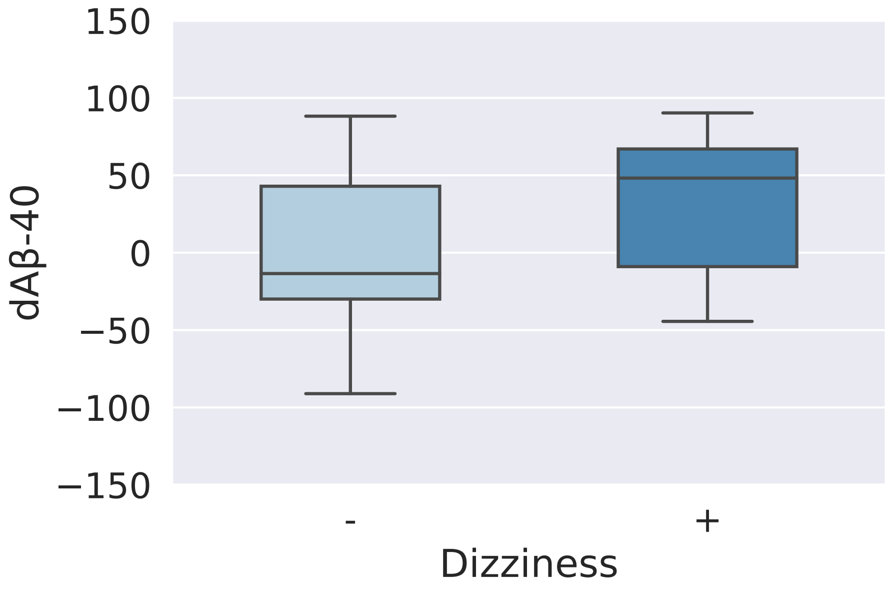


**Fig. 9 Relationship between delta Aβ-40 and dizziness. Data is displayed as a box-whisker plot (median along with 5%–95% range). Participants who showed no change or decrease (-) along with those who showed an increase (+) are shown.**

## 3. Deep Neural Networks for mTBI

Additional data became available from WRAIR in the form of a series of cohort studies with varying number of participants and was provided to us in spreadsheets containing the following information for each event:

- Blast overpressure sensor data provided by the Blast Gauge System.
- Symptoms and demographic data provided by questionnaires.
- Brain health data provided by the DANA software for brain health screening.

This new data provided for a larger sample size and hence more of an opportunity to explore using more advanced ML processes to analyze the data.

## 3.1 Data Storage, Ingest, and Access

The ETL process can be extremely burdensome for data scientists. The time spent preprocessing and getting data into appropriate formats can easily dwarf that of the actual time spent analyzing the data. We developed a process flow that uses multiple tools for this task. Predominately, this process involves data ingest, cleanup, and providing availability to the requesting analytics services. The process flow view of the architecture of the system can be seen in Fig. 10.

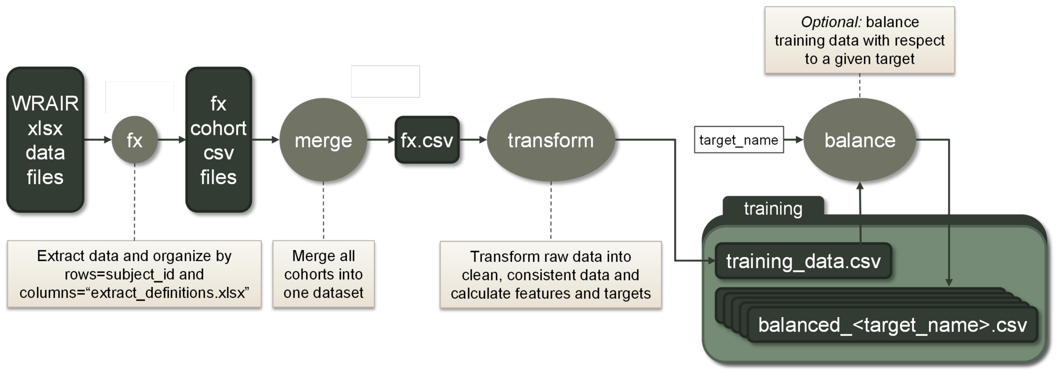
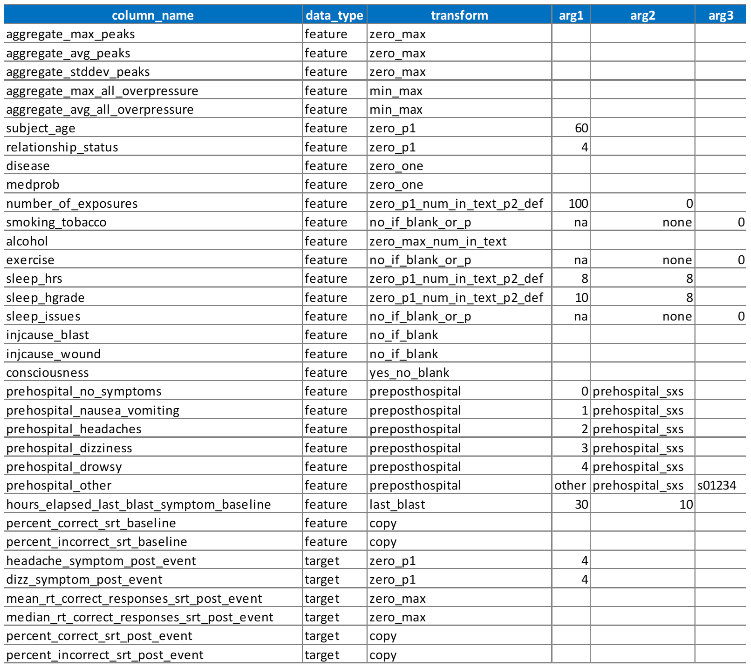

WRAIR xlsx data files
fx
fx cohort csv files
merge
fx.csv
transform
target_name
balance
Optional: balance training data with respect to a given target
training
training_data.csv
balanced_<target_name>.csv
Extract data and organize by rows=subject_id and columns="extract_definitions.xlsx"
Merge all cohorts into one dataset
Transform raw data into clean, consistent data and calculate features and targets

**Fig. 10 The architecture for data ingest and access**

To prepare data for analysis in this work, a series of Python ETL jobs was implemented. A separate job was initiated for the ingest, clean, and feature extraction steps. The software framework described previously was used to tie the jobs together into one workflow. There was much variability across the cohort data sets due to differences in the semantics of data variables collected and, possibly, human error or inconsistencies. As a result, a separate non-generalizable ETL job for the ingest, clean, and extraction steps had to be written for each cohort data set. Using this, the raw data sets were unified into one feature table that contained the desired summary statistics and other engineered values across all cohorts. For this work, the data was not stored in the data store, but was simply converted into a CSV file. If a particular feature was not present in all the cohorts, that data was not included in the data used to train the ML models.

Next, this file (fx.csv) is transformed into another CSV file by way of a transform process as specified by a function file. This file provides the instructions required to transform the data (such as how to respond to blank fields) as it is being loaded. An excerpt from one of these function files is shown in Fig. 11. Overall, this type of process is often required due to inconsistencies between cohort data and to fix data that is entered in free form.



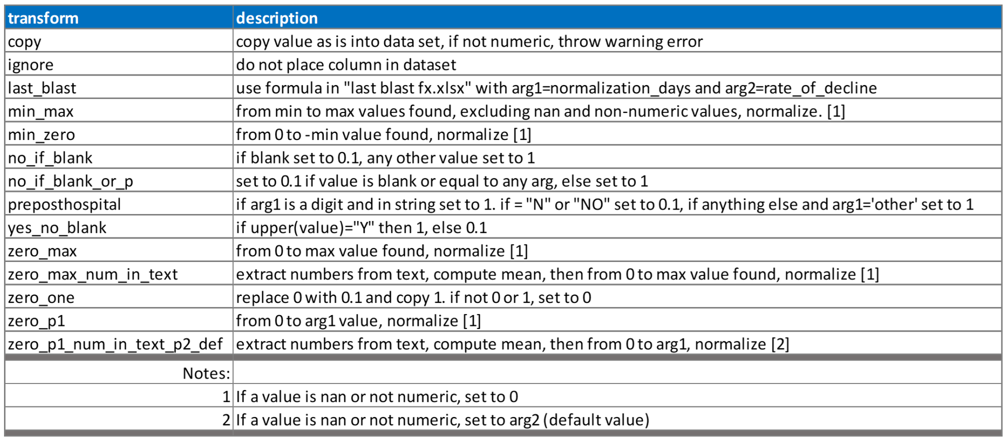
| column_name | data_type | transform | arg1 | arg2 | arg3 |
|---|---|---|---|---|---|
| aggregate_max_peaks | feature | zero_max | | | |
| aggregate_avg_peaks | feature | zero_max | | | |
| aggregate_stddev_peaks | feature | zero_max | | | |
| aggregate_max_all_overpressure | feature | min_max | | | |
| aggregate_avg_all_overpressure | feature | min_max | | | |
| subject_age | feature | zero_p1 | 60 | | |
| relationship_status | feature | zero_p1 | 4 | | |
| disease | feature | zero_one | | | |
| medprob | feature | zero_one | | | |
| number_of_exposures | feature | zero_p1_num_in_text_p2_def | 100 | 0 | |
| smoking_tobacco | feature | no_if_blank_or_p | na | none | 0 |
| alcohol | feature | zero_max_num_in_text | | | |
| exercise | feature | no_if_blank_or_p | na | none | 0 |
| sleep_hrs | feature | zero_p1_num_in_text_p2_def | 8 | 8 | |
| sleep_hgrade | feature | zero_p1_num_in_text_p2_def | 10 | 8 | |
| sleep_issues | feature | no_if_blank_or_p | na | none | 0 |
| injcause_blast | feature | no_if_blank | | | |
| injcause_wound | feature | no_if_blank | | | |
| consciousness | feature | yes_no_blank | | | |
| prehospital_no_symptoms | feature | preposthospital | 0 | prehospital_sxs | |
| prehospital_nausea_vomiting | feature | preposthospital | 1 | prehospital_sxs | |
| prehospital_headaches | feature | preposthospital | 2 | prehospital_sxs | |
| prehospital_dizziness | feature | preposthospital | 3 | prehospital_sxs | |
| prehospital_drowsy | feature | preposthospital | 4 | prehospital_sxs | |
| prehospital_other | feature | preposthospital | other | prehospital_sxs | s01234 |
| hours_elapsed_last_blast_symptom_baseline | feature | last_blast | 30 | 10 | |
| percent_correct_srt_baseline | feature | copy | | | |
| percent_incorrect_srt_baseline | feature | copy | | | |
| headache_symptom_post_event | target | zero_p1 | 4 | | |
| dizz_symptom_post_event | target | zero_p1 | 4 | | |
| mean_rt_correct_responses_srt_post_event | target | zero_max | | | |
| median_rt_correct_responses_srt_post_event | target | zero_max | | | |
| percent_correct_srt_post_event | target | copy | | | |
| percent_incorrect_srt_post_event | target | copy | | | |

**Fig. 11 Excerpt from the function file**

Each column name can be either a feature or a target. A feature is used as an input to the ML model, and the targets are what we want the model to learn. Another term common in ML is a label. A label is the true outcome of the target. Labeled data is used in supervised learning and is known to the training stage but hidden from the testing phase.

The transform provides the name of the function that is used to convert the data, and the arguments are used by the function. The function descriptions can be seen in Fig. 12. All of these functions convert the data provided by the extraction and cleaning process into values between 0.0 and 1.0. A value of 0.0 means that the data was empty or unrecognizable. Otherwise, the range of value provided by the data is normalized within the range of 0.1 to 1.0

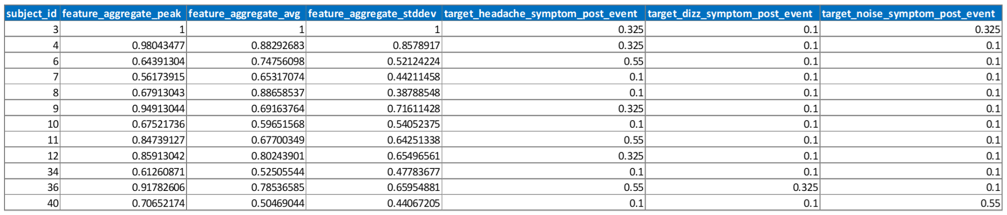
Any data that does not meet the minimum requirements (such as no blast overpressure data) is not included in the training data set. Also, each cohort was found to



| transform | description |
|---|---|
| copy | copy value as is into data set, if not numeric, throw warning error |
| ignore | do not place column in dataset |
| last_blast | use formula in "last blast fx.xlsx" with arg1=normalization_days and arg2=rate_of_decline |
| min_max | from min to max values found, excluding nan and non-numeric values, normalize. [1] |
| min_zero | from 0 to -min value found, normalize [1] |
| no_if_blank | if blank set to 0.1, any other value set to 1 |
| no_if_blank_or_p | set to 0.1 if value is blank or equal to any arg, else set to 1 |
| preposthospital | if arg1 is a digit and in string set to 1. if = "N" or "NO" set to 0.1, if anything else and arg1='other' set to 1 |
| yes_no_blank | if upper(value)="Y" then 1, else 0.1 |
| zero_max | from 0 to max value found, normalize [1] |
| zero_max_num_in_text | extract numbers from text, compute mean, then from 0 to max value found, normalize [1] |
| zero_one | replace 0 with 0.1 and copy 1. if not 0 or 1, set to 0 |
| zero_p1 | from 0 to arg1 value, normalize [1] |
| zero_p1_num_in_text_p2_def | extract numbers from text, compute mean, then from 0 to arg1, normalize [2] |
| Notes: | |
| 1 | If a value is nan or not numeric, set to 0 |
| 2 | If a value is nan or not numeric, set to arg2 (default value) |

**Fig. 12 FX transform function descriptions**

have different data. For a complete catalog, all data was placed in the FX file, but only columns that exist in all cohorts are included in the training data since missing data would confuse the ML models. The result is a training data set ready for the training phase. An excerpt is in Fig. 13.



| subject_id | feature_aggregate_peak | feature_aggregate_avg | feature_aggregate_stddev | target_headache_symptom_post_event | target_dizz_symptom_post_event | target_noise_symptom_post_event |
|---|---|---|---|---|---|---|
| 3 | 1 | 1 | 1 | 0.325 | 0.1 | 0.325 |
| 4 | 0.98043477 | 0.88292683 | 0.8578917 | 0.325 | 0.1 | 0.1 |
| 6 | 0.64391304 | 0.74756098 | 0.52124224 | 0.55 | 0.1 | 0.1 |
| 7 | 0.56173915 | 0.65317074 | 0.44211458 | 0.1 | 0.1 | 0.1 |
| 8 | 0.67913043 | 0.88658537 | 0.38788548 | 0.1 | 0.1 | 0.1 |
| 9 | 0.94913044 | 0.69163764 | 0.71611428 | 0.325 | 0.1 | 0.1 |
| 10 | 0.67521736 | 0.59651568 | 0.54052375 | 0.1 | 0.1 | 0.1 |
| 11 | 0.84739127 | 0.67700349 | 0.64251338 | 0.55 | 0.1 | 0.1 |
| 12 | 0.85913042 | 0.80243901 | 0.65496561 | 0.325 | 0.1 | 0.1 |
| 34 | 0.61260871 | 0.52505544 | 0.47783677 | 0.1 | 0.1 | 0.1 |
| 36 | 0.91782606 | 0.78536585 | 0.65954881 | 0.55 | 0.325 | 0.1 |
| 40 | 0.70652174 | 0.50469044 | 0.44067205 | 0.1 | 0.1 | 0.55 |

**Fig. 13 Excerpt from the training data set**

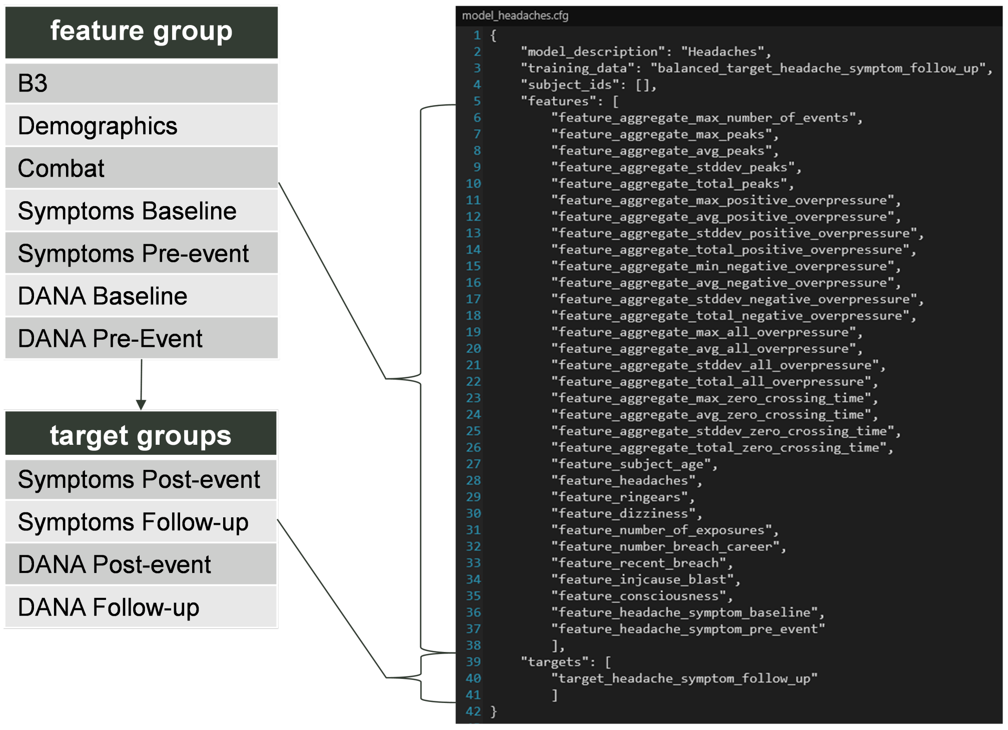
There is also an optional process called balancing that can be useful. This balance approach is used to deal with the sparse target issue. A sparse target is one that does not exist frequently in the training data. ML models have difficulty learning sparse targets. When a model is developed that has one target, the training data can be balanced using that target. This only applies to targets that represent classes, such as the questionnaire answers that are limited to only the values 0, 1, 2, 3, or 4. Balancing works by randomly adding $\alpha$ records of the classes that are less frequent ($nclass$), where $\alpha = \frac{nmax-nclass}{2}$. The class with maximum samples is nclass. For example, if target 0 occurred 100 times and target 1 occurred 80 times, 10 randomly selected samples would be added to the training data for target 1. The result is a more balanced data set that will be easier to learn.

## 3.2 ML Approach

We created a flexible Python software application where a user can select the following:

- Inputs (features)
- Outputs (what you want to predict, i.e., targets)
- Train a model
- Visualize the results in various formats

The application is controlled by a model configuration file (Fig. 14) that defines what the model will learn (the targets) and which features it will use to learn those targets. Figure 14 also provides a top-level view of the many possible combinations of features and targets. Any pairing of features and targets can be used to create a model using a model configuration file.



```
{
    "model_description": "Headaches",
    "training_data": "balanced_target_headache_symptom_follow_up",
    "subject_ids": [],
    "features": [
        "feature_aggregate_max_number_of_events",
        "feature_aggregate_max_peaks",
        "feature_aggregate_avg_peaks",
        "feature_aggregate_stddev_peaks",
        "feature_aggregate_total_peaks",
        "feature_aggregate_max_positive_overpressure",
        "feature_aggregate_avg_positive_overpressure",
        "feature_aggregate_stddev_positive_overpressure",
        "feature_aggregate_total_positive_overpressure",
        "feature_aggregate_min_negative_overpressure",
        "feature_aggregate_avg_negative_overpressure",
        "feature_aggregate_stddev_negative_overpressure",
        "feature_aggregate_total_negative_overpressure",
        "feature_aggregate_max_all_overpressure",
        "feature_aggregate_avg_all_overpressure",
        "feature_aggregate_stddev_all_overpressure",
        "feature_aggregate_total_all_overpressure",
        "feature_aggregate_max_zero_crossing_time",
        "feature_aggregate_avg_zero_crossing_time",
        "feature_aggregate_stddev_zero_crossing_time",
        "feature_aggregate_total_zero_crossing_time",
        "feature_subject_age",
        "feature_headaches",
        "feature_ringears",
        "feature_dizziness",
        "feature_number_of_exposures",
        "feature_number_breach_career",
        "feature_recent_breach",
        "feature_injcause_blast",
        "feature_consciousness",
        "feature_headache_symptom_baseline",
        "feature_headache_symptom_pre_event"
        ],
    "targets": [
        "target_headache_symptom_follow_up"
        ]
}
```

**Fig. 14 Features and targets for the ML model showing the layers**

Some combinations of targets are too sparse for a ML model to learn them. In the case when only one target is specified, the user can choose to balance the data set. This balance process ensures that each class of targets has the same number of samples in the training data. This is accomplished by randomly selecting and copying samples until all classes have the same number of samples.

The models were developed using Google TensorFlow version 2.7 and commonly used ML packages that were previously described, to include NumPy and Pandas. The ML models are created with four layers of neurons (see Fig. 15) as follows:

- Inputs layer: N neurons where N is the number of input features specified
- Output layer: U neurons where U is the number of targets specified
- Hidden layer 1: 2N neurons
- Hidden layer 2: mean of 2N and U neurons

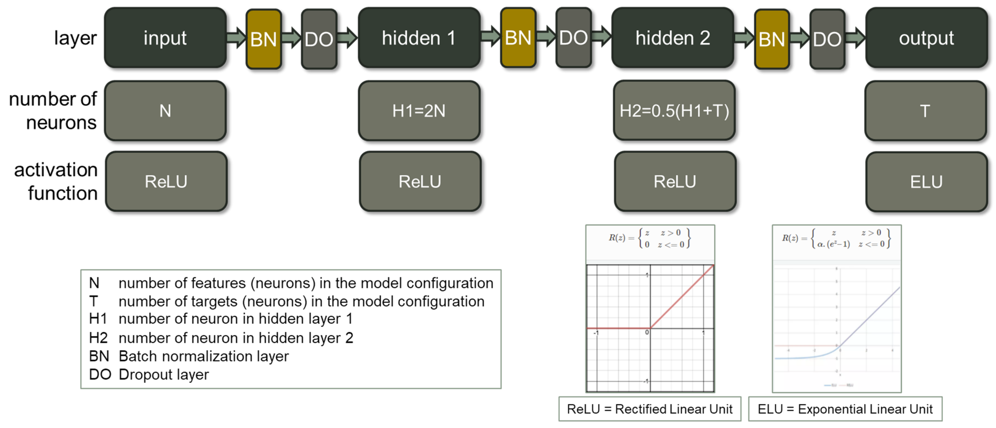

layer
input
BN
DO
hidden 1
BN
DO
hidden 2
BN
DO
output
number of neurons
N
H1=2N
H2=0.5(H1+T)
T
activation function
ReLU
ReLU
ReLU
ELU
N number of features (neurons) in the model configuration
T number of targets (neurons) in the model configuration
H1 number of neuron in hidden layer 1
H2 number of neuron in hidden layer 2
BN Batch normalization layer
DO Dropout layer
ReLU = Rectified Linear Unit
ELU = Exponential Linear Unit

**Fig. 15 Features and targets for the ML model**

Using the example configuration file in Fig. 14, the "Headaches" model has 32 input neurons, 64 hidden 1 neurons, 32 hidden 2 neurons, and 1 output neuron. Between each layer of neurons are batch normalization and drop out (DO) layers. For the models shown in this report, the DO layers were set to 0, meaning no drop out. When larger data sets become available, the DO layer will be used.

The activation functions ReLU and ELU were found to be most effective for this data. The training process used the Adam optimizer, a mean absolute error loss function, and a batch size of four samples. When more data is available, the sample size could be increased.

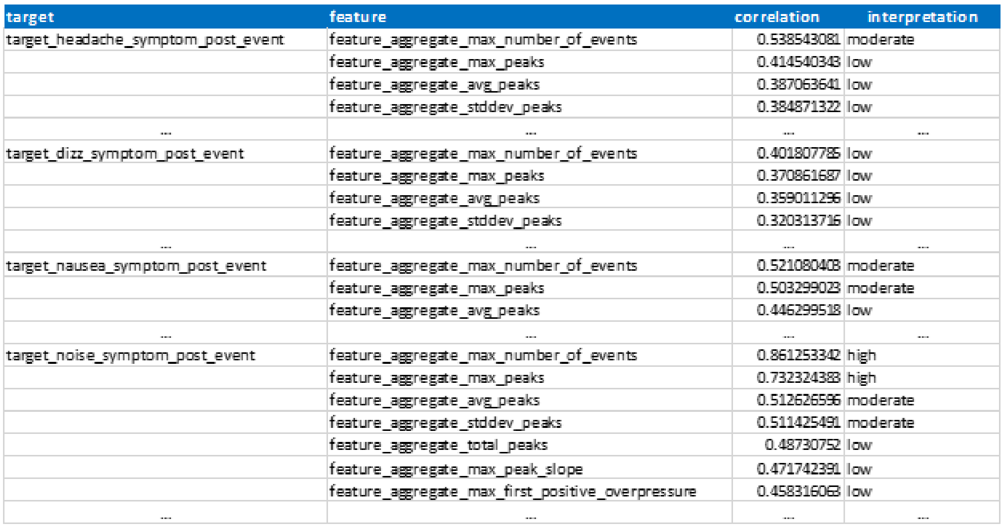
## 3.3 ML Tools

We developed two tools that aid in the process of selecting features and targets to include in models: Pearson-R correlations and feature-target box plots. The Pearson-R correlations are stored in a CSV file for every feature-target pair and they are interpreted based on the p-value calculations using the values shown in Table 8.

**Table 8 Pearson-R p-value interpretations**

| Size of Correlation | Interpretation |
|---|---|
| 0.90 to 1.00 (–0.90 to –1.00) | Very high positive (negative) correlation |
| 0.70 to 0.90 (–0.70 to –0.90) | High positive (negative) correlation |
| 0.50 to 0.70 (–0.50 to –0.70) | Moderate positive (negative) correlation |
| 0.30 to 0.50 (–0.30 to –0.50) | Low positive (negative) correlation |
| 0.00 to 0.30 (0.00 to –0.30) | Negligible correlation |

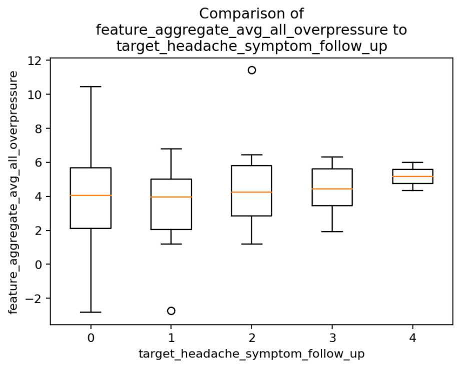
An excerpt of the correlation file is shown in Fig. 16. It is sorted by interpretation (very high to low and negligible values are not included), and then by correlation.



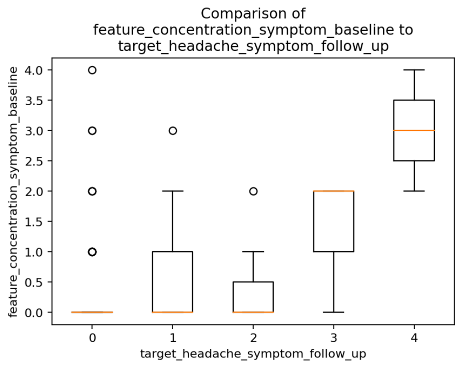
| target | feature | correlation | interpretation |
|---|---|---|---|
| target_headache_symptom_post_event | feature_aggregate_max_number_of_events | 0.538543081 | moderate |
| | feature_aggregate_max_peaks | 0.414540343 | low |
| | feature_aggregate_avg_peaks | 0.387063641 | low |
| | feature_aggregate_stddev_peaks | 0.384871322 | low |
| ... | ... | ... | ... |
| target_dizz_symptom_post_event | feature_aggregate_max_number_of_events | 0.401807785 | low |
| | feature_aggregate_max_peaks | 0.370861687 | low |
| | feature_aggregate_avg_peaks | 0.359011296 | low |
| | feature_aggregate_stddev_peaks | 0.320313716 | low |
| ... | ... | ... | ... |
| target_nausea_symptom_post_event | feature_aggregate_max_number_of_events | 0.521080403 | moderate |
| | feature_aggregate_max_peaks | 0.503299023 | moderate |
| | feature_aggregate_avg_peaks | 0.446299518 | low |
| ... | ... | ... | ... |
| target_noise_symptom_post_event | feature_aggregate_max_number_of_events | 0.861253342 | high |
| | feature_aggregate_max_peaks | 0.732324383 | high |
| | feature_aggregate_avg_peaks | 0.512626596 | moderate |
| | feature_aggregate_stddev_peaks | 0.511425491 | moderate |
| | feature_aggregate_total_peaks | 0.48730752 | low |
| | feature_aggregate_max_peak_slope | 0.471742391 | low |
| | feature_aggregate_max_first_positive_overpressure | 0.458316063 | low |
| ... | ... | ... | ... |

**Fig. 16 Excerpt from Pearson-R correlation results**

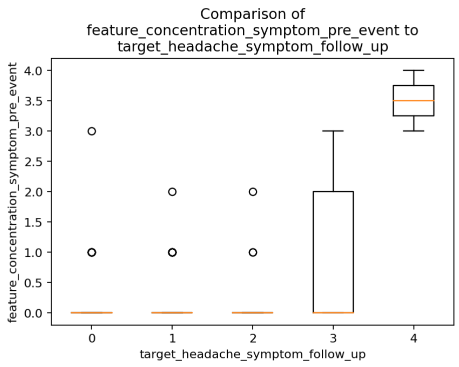
The feature-target box and whisker plots provide a visualization of the range of feature values in the training data set for each target value. Figure 17 shows the feature plots for the post-event headache symptom. Overlaps in feature values across different targets means the feature has a lower probability of being a discriminator for prediction.



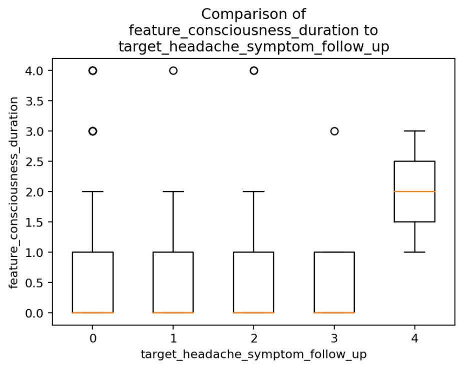
(a)



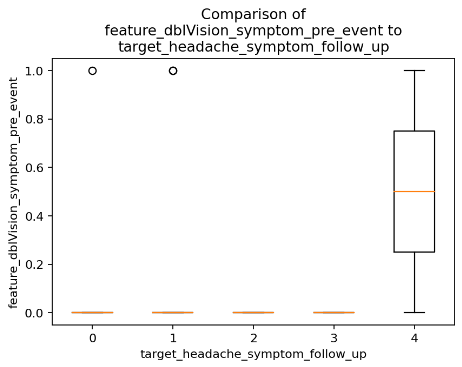
(b)



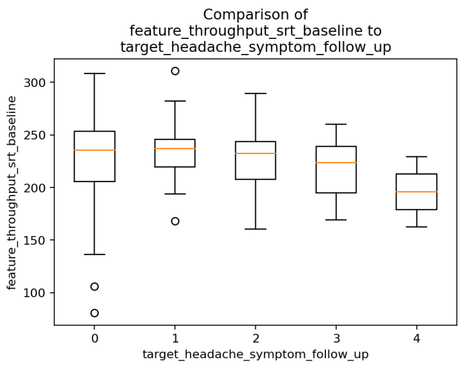
(c)



(d)



(e)



(f)

**Fig. 17 Feature-target correlation box and whisker plots for headache symptom correlated with (a) average of all overpressure readings, (b) baseline concentration, (c) pre-event concentration, (d) consciousness duration, (e) pre-event double vision, and (f) baseline throughput DANA SRT**

## 3.4 ML Results

The ML software application is a flexible one that uses the data ingested by the feature extraction process to create models and visualize their predictive capabilities as shown in Fig. 18. When new data is received, it is ingested and the feature extraction process creates the fx.csv file, which contains data from all cohorts for every subject that participated in the event. The merge and transform processes are run once to create a comprehensive training data set. Specific target-based training data sets can be created with the balance process, if required when sparse targets are present. These tools only need to be run once to generate a great deal of information to aid in the creation of model configurations. Once a model configuration is defined, the train process creates a model that learns how to generate the targets using the features specified in the model configuration. Then predictions can be run to generate plots showing the model's ability to relate the features to the targets. From this, an analyst can determine levels that correspond to probable target responses.

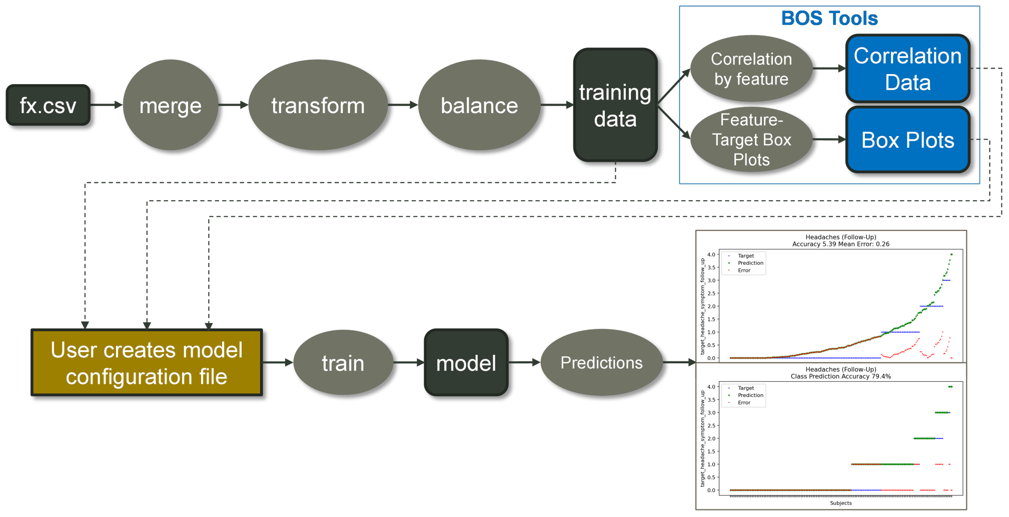


**Fig. 18 ML software application workflow**

To demonstrate how this process works, we have created three example models:

1. Headaches (target_headache_symptom_follow_up)
2. Long think (target_longThink_symptom_post_event)
3. Sleep (target_sleep_symptom_follow_up)

Each model predicts if a subject will experience the target specified using all of the features that have Pearson-R statistical significance interpreted as low, moderate, high, or very high. See Appendixes A, B, and C for the model configuration files.

The headaches, long think, and sleep model predictions are shown in Figs. 19–24. Prediction plots are created by running each subject through the model and generating a prediction. The data is then sorted by the actual target response and then by the predicted response and plotted along with the error between the target and predicted values calculated as $error = |(target - prediction)|$.

The models use nonlinear regression and output values between 0 and 1 that are converted to the target value ranges. The targets for the example models are subject responses that are in the range of 0 to 4 (see Table 5 for the complete list of questions). Figures 19, 21, and 23 show the raw output of the models and Figs. 20, 22, and 24 show the class outputs, which are simply rounding the raw output to the nearest class value.

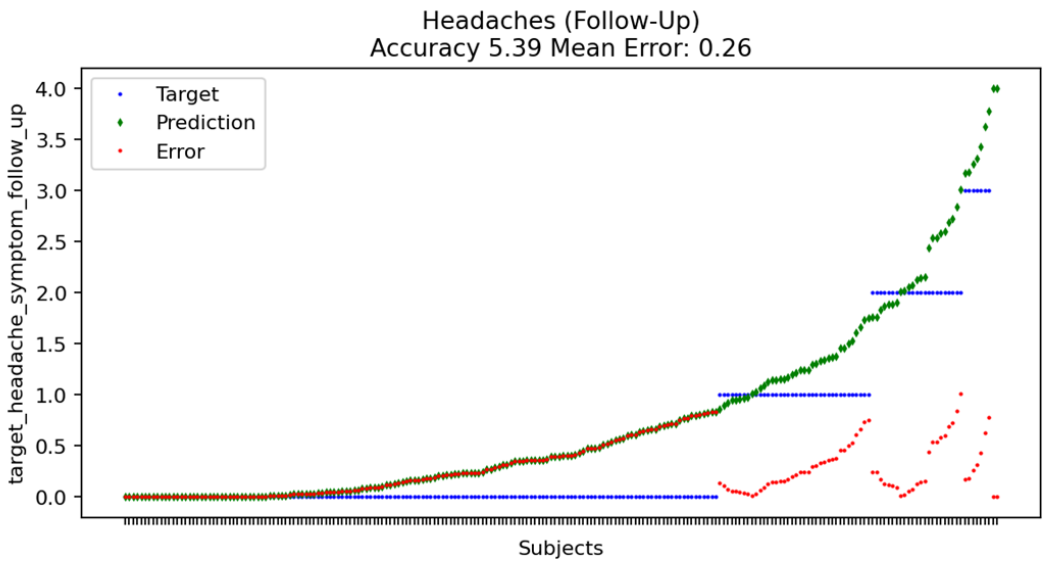


**Fig. 19 Headaches–raw prediction**

Each raw prediction displays prediction accuracy ($a$) and mean error ($e$), and the class prediction displays class prediction accuracy ($c$), calculated as shown in Eq. 7, where $t$ is the target, $p$ is prediction, and $N$ is the number of subjects. These models show that responses to the post-event or follow-up surveys can be predicted with an accuracy of 79%–98% using this approach.

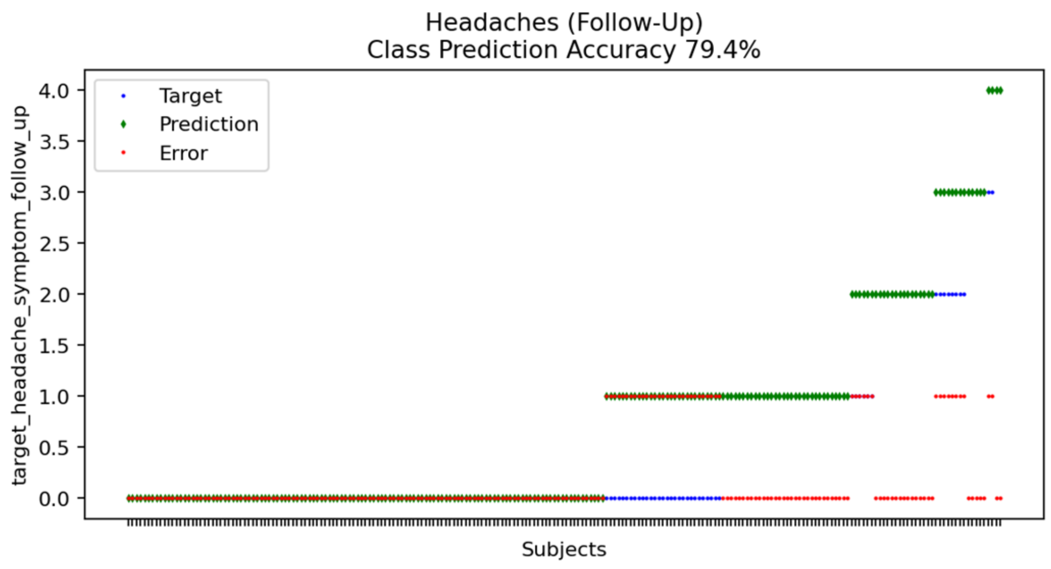


**Fig. 20 Headaches–class prediction**

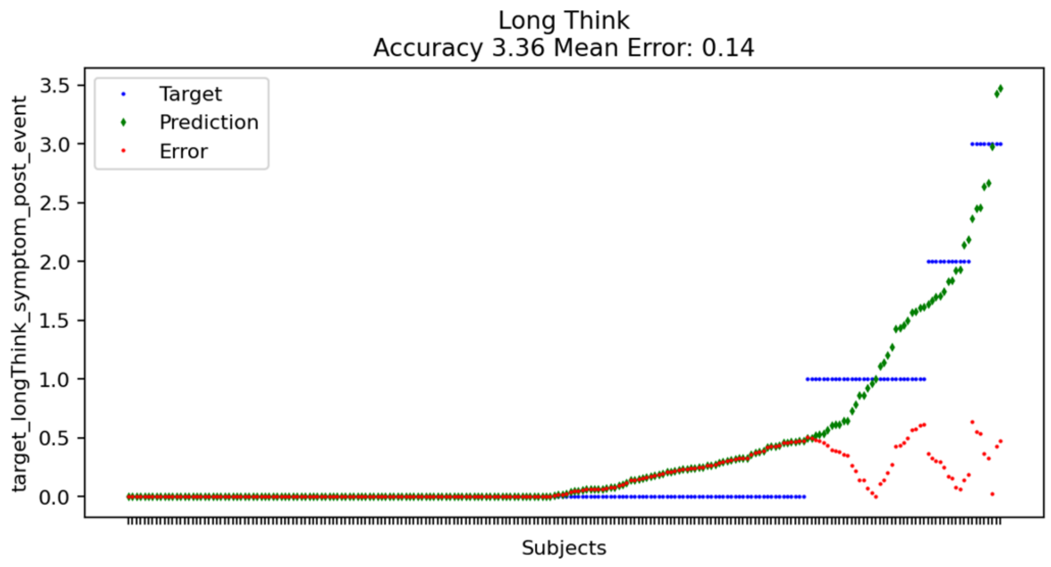


**Fig. 21 Long think–raw prediction**

$$
\begin{aligned}
a &= \sqrt{\sum (t-p)^2} \\
e &= \frac{1}{N}\sum |(t-p)| \\
c &= (1 - \frac{\sum min(|(t-p)|), 1)}{N}) \times 100
\end{aligned} \tag{7}
$$

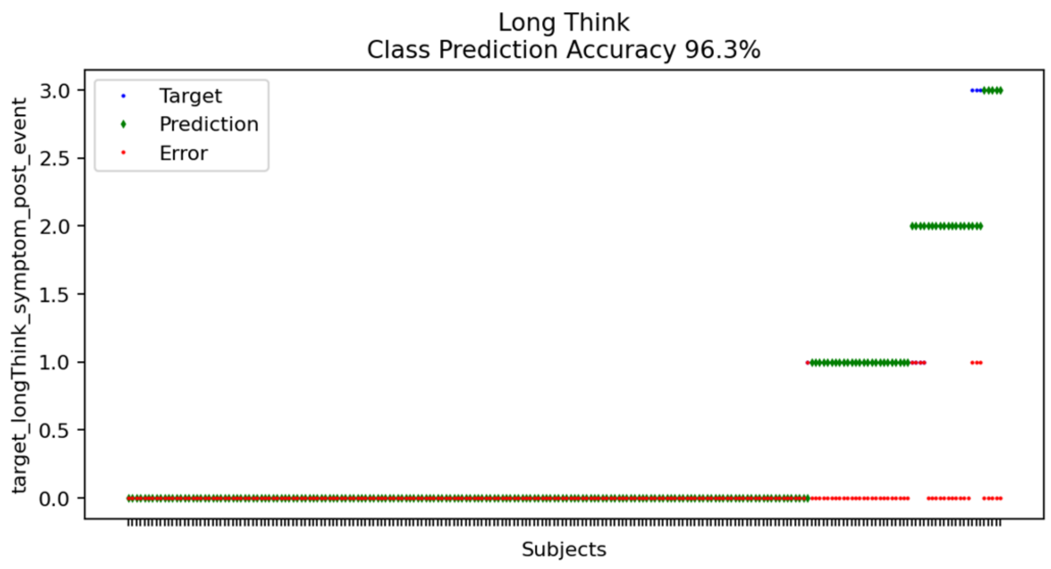


**Fig. 22  Long think–class prediction**

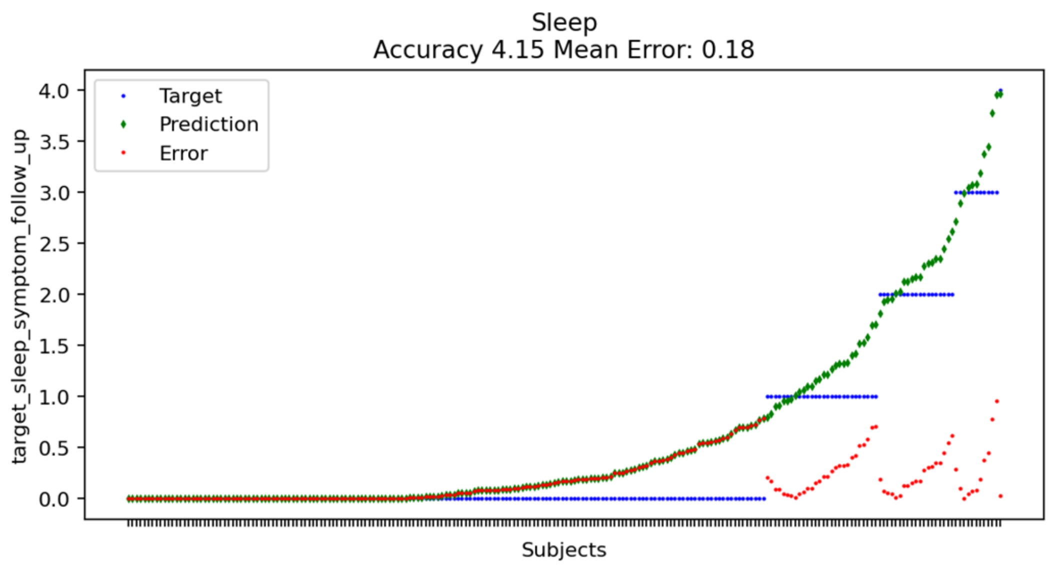


**Fig. 23  Sleep–raw prediction**

## 4. Conclusions, Summary, and Recommendations

We began this project with the underlying objective of understanding and replicating state-of-the-art mTBI analysis methods. Where appropriate, we wanted to explore how these might be enhanced with AI/ML approaches and automated where possible, and how these tasks could be accomplished in an open source frame-

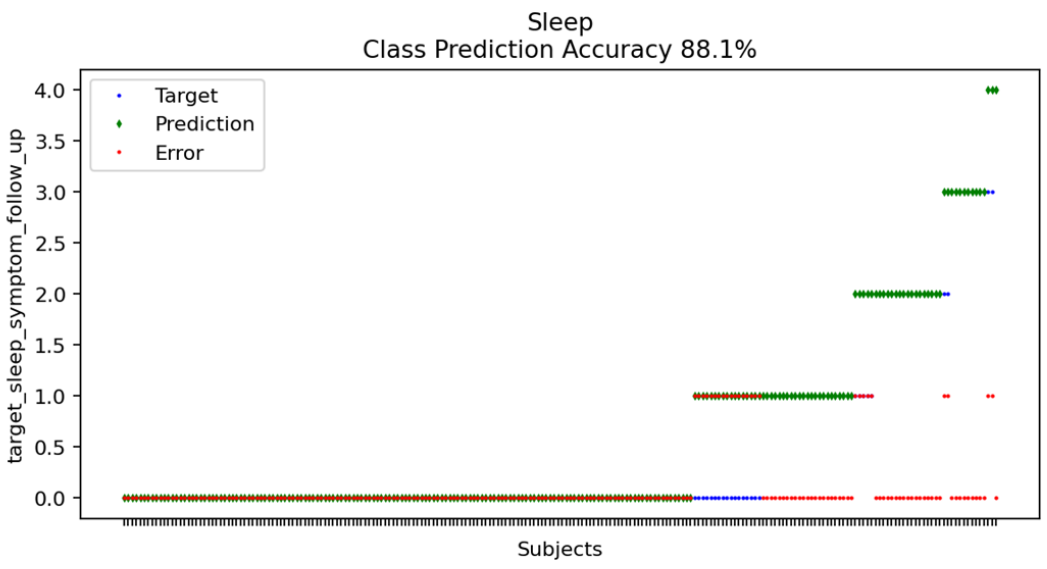


**Fig. 24 Sleep–class prediction**

work. While commercial-based tools have their place, we were also interested in how these processes might be replicated and expanded in areas common to data sciences. The ability to perform automated outlier detection and statistical analysis, coupled with newer ML techniques, paves the way for rapid evaluation of emerging data sets that can also be expanded to scalable computing resources, such as parallel supercomputers, should other more compute-costly processes such as neuroimaging (positron emission tomography [PET], magnetic resonance imaging [MRI], computed tomography [CT], etc.) be added.

The suite of data warehousing, statistical, and ML tools resulting at the conclusion of this effort provides a baseline capability for unbiased, automated analysis of exposure associated with outcome (also known as "dose-response"). This capability can be expanded and enhanced with the following work:

- Adding new data sources. Other existing data can be added to the process, specifically, exposures from other breaching environments and heavy weapons use and other outcome variables, such as biosample-based physiological markers of neurotrauma (especially amyloid-β [Aβ-42]). Predictive model validation can be executed by some existing data withheld from the ML process (expressly for validation purposes) or by the collection of new data.

- Creating blast overpressure studies (BOS) AI for analysts. BOS AI in its current state is a collection of Python programs that must be run manually and in the right sequence. An integrated application that hides all the algorithms and processes and is easy to use would be very useful for anyone analyzing this data. All of the data manipulation and AI would be obfuscated and the user would have a clean UI to build their own models and test hypotheses based on their individual expertise.
- Integration of analysis models with the lakehouse. While the data was ingested, cleaned, and features extracted using the developed software stack, the analysis itself did not interface or pull data directly from the lakehouse. Integrating the AI algorithms into the automated workflow would be an important step in allowing for application integration. Further, once the lakehouse is deployed onto the high-performance computing systems, the AI would scale and automatically refine as more data gets ingested through the pipeline.
- Adding further ML techniques. The current models all use deep neural networks. Other ML techniques such as decision trees, random forests, or gradient boosting may produce better models depending upon the features and targets available.
- Including neuroimaging in the workflow. Although not described in this narrative, one of the early items investigated was the use of tools like Scikit-Learn for neuroimaging. None of the data sets from WRAIR had these types of images, but early work we did with the Haxby data set showed the capabilities of Scikit-Learn to integrate with the rest of the tools and achieve some small amount of speedup using parallel computing assets.[12]
- Designing the tenants of a military personnel blast exposure monitoring program. The ultimate goal of these efforts is to protect our military personnel. Programs that monitor each person exposed to blasts could predict when a person should be removed from such missions. Repeated exposure of various magnitude can lead to long-lasting medical issues. This program would develop a tool that could predict the probability of issues in advance of an actual exposure.

In conclusion, by developing and coupling these capabilities, an expandable and streamlined research and development environment will emerge for mTBI allowing for longitudinal studies and enhanced Soldier protection. Focusing on open source methods and tools will allow for the quickest insertion of customized methods that could also be optimized for large-scale computing resources should the need arise. Bringing effective data management and processing together is the proper way forward for BOS and analysis, and this project demonstrated the right building blocks to achieve this goal.

## 5. References

1. Zaharia M, Ghodsi A, Xin R, Armbrust M. Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. In: Proceedings of the 11th Conference on Innovative Data Systems Research, CIDR 2021, virtual event, 2021 Jan 11–15, online proceedings; www.cidrdb.org; 2021.

2. Motulsky HJ, Brown RE. Detecting outliers when fitting data with nonlinear regression – a new method based on robust nonlinear regression and the false discovery rate. BMC Bioinformatics. 2006;7(1):123.

3. Boutté AM, Thangavelu B, LaValle CR, Nemes J, Gilsdorf J, Shear DA, Kamimori GH. Brain-related proteins as serum biomarkers of acute, subconcussive blast overpressure exposure: a cohort study of military personnel. PLOS ONE. 2019;14(8):1–16.

4. Top 10 Python libraries for data science. 2019 Dec [https://towardsdatascience.com/top-10-python-libraries-for-data-science-cd82294ec266.

5. Babich R. Levenberg Marquardt algorithm in C. 2008 May [accessed 2022 Mar 28]. https://github.com/leechwort/levenberg-maquardt-example/blob/master/levmarq.c.

6. King NS, Crawford S, Wenden FJ, Moss NE, Wade DT. The rivermead post concussion symptoms questionnaire: a measure of symptoms commonly experienced after head injury and its reliability. Journal of Neurology. 1995;242(9).

7. Lathan C, Spira JL, Bleiberg J, Vice J, Tsao JW. Defense Automated Neurobehavioral Assessment (DANA)—psychometric properties of a new field-deployable neurocognitive assessment tool. Military Medicine. 2013;178(4):365–371.

8. LaValle CR, Carr WS, Egnoto MJ, Misistia AC, Salib JE, Ramos AN, Kamimori GH. Neurocognitive performance deficits related to immediate and acute blast overpressure exposure. Frontiers in Neurology. 2019;10.

9. Hyndman R, Fan Y. Sample quantiles in statistical packages. The American Statistician. 1996;50:361–365.

10. On-line GraphPad Prism manual pages. n.d. [accessed 2022 Jan 31]. https://www.graphpad.com/guides/prism/latest/statistics/stat_percentiles_and_the_median.htm.

11. Phillips C. Army announces FDA clearance of field deployable TBI blood test. 2021 Mar [https://health.mil/News/Articles/2021/03/12/Army-Announces-FDA-Clearance-of-Field-Deployable-TBI-Blood-Test.

12. Haxby J, Gobbini M, Furey M, Ishai A, Schouten J, Pietrini P. Distributed and overlapping representations of faces and objects in ventral temporal cortex. Science. 2001;293:2425–30.

## Appendix A. model_headache.cfg Configuration File

```
{
        "model_description": "Headaches␣(Follow-Up)",
        "training_data": "balanced_target_headache_symptom_follow_up",
        "subject_ids": [],
        "features": [
                "feature_aggregate_max_number_of_events",
                "feature_aggregate_max_peaks",
                "feature_aggregate_avg_peaks",
                "feature_aggregate_stddev_peaks",
                "feature_aggregate_total_peaks",
                "feature_aggregate_max_peak_slope",
                "feature_aggregate_max_first_positive_overpressure",
                "feature_aggregate_avg_first_positive_overpressure",
                "feature_aggregate_stddev_first_positive_overpressure",
                "feature_aggregate_total_first_positive_overpressure",
                "feature_aggregate_max_positive_overpressure",
                "feature_aggregate_avg_positive_overpressure",
                "feature_aggregate_stddev_positive_overpressure",
                "feature_aggregate_total_positive_overpressure",
                "feature_aggregate_min_negative_overpressure",
                "feature_aggregate_avg_negative_overpressure",
                "feature_aggregate_stddev_negative_overpressure",
                "feature_aggregate_total_negative_overpressure",
                "feature_aggregate_max_all_overpressure",
                "feature_aggregate_avg_all_overpressure",
                "feature_aggregate_stddev_all_overpressure",
                "feature_aggregate_total_all_overpressure",
                "feature_aggregate_max_zero_crossing_time",
                "feature_aggregate_avg_zero_crossing_time",
                "feature_aggregate_stddev_zero_crossing_time",
                "feature_aggregate_total_zero_crossing_time",
                "feature_subject_age",
                "feature_relationship_status",
                "feature_disease",
                "feature_medprob",
                "feature_meningitis",
                "feature_concussion",
                "feature_headaches",
                "feature_ringears",
                "feature_deafness",
                "feature_dischargingears",
                "feature_fullears",
                "feature_nosesinusthroat",
                "feature_fainting",
                "feature_dizziness",
                "feature_coordination"
                ],
        "targets": [
                "target_headache_symptom_follow_up"
                ]
}
```

## Appendix B. model_longThink.cfg Configuration File

```
{
        "model_description": "Long␣Think",
        "training_data": "balanced_target_longThink_symptom_post_event",
        "subject_ids": [],
        "features": [
                "feature_aggregate_max_number_of_events",
                "feature_aggregate_max_peaks",
                "feature_aggregate_avg_peaks",
                "feature_aggregate_stddev_peaks",
                "feature_aggregate_total_peaks",
                "feature_aggregate_max_peak_slope",
                "feature_aggregate_max_first_positive_overpressure",
                "feature_aggregate_avg_first_positive_overpressure",
                "feature_aggregate_stddev_first_positive_overpressure",
                "feature_aggregate_total_first_positive_overpressure",
                "feature_aggregate_max_positive_overpressure",
                "feature_aggregate_avg_positive_overpressure",
                "feature_aggregate_stddev_positive_overpressure",
                "feature_aggregate_total_positive_overpressure",
                "feature_aggregate_min_negative_overpressure",
                "feature_aggregate_avg_negative_overpressure",
                "feature_aggregate_stddev_negative_overpressure",
                "feature_aggregate_total_negative_overpressure",
                "feature_aggregate_max_all_overpressure",
                "feature_aggregate_avg_all_overpressure",
                "feature_aggregate_stddev_all_overpressure",
                "feature_aggregate_total_all_overpressure",
                "feature_aggregate_max_zero_crossing_time",
                "feature_aggregate_avg_zero_crossing_time",
                "feature_aggregate_stddev_zero_crossing_time",
                "feature_aggregate_total_zero_crossing_time",
                "feature_subject_age",
                "feature_relationship_status",
                "feature_disease",
                "feature_medprob",
                "feature_meningitis"
                ],
        "targets": [
                    "target_longThink_symptom_post_event"
                    ]
}
```

## Appendix C. model_sleep.cfg Configuration File

```
{
        "model_description": "Sleep",
        "training_data": "balanced_target_sleep_symptom_follow_up",
        "subject_ids": [],
        "features": [
                "feature_aggregate_max_number_of_events",
                "feature_aggregate_max_peaks",
                "feature_aggregate_avg_peaks",
                "feature_aggregate_stddev_peaks",
                "feature_aggregate_total_peaks",
                "feature_aggregate_max_peak_slope",
                "feature_aggregate_max_first_positive_overpressure",
                "feature_aggregate_avg_first_positive_overpressure",
                "feature_aggregate_stddev_first_positive_overpressure",
                "feature_aggregate_total_first_positive_overpressure",
                "feature_aggregate_max_positive_overpressure",
                "feature_aggregate_avg_positive_overpressure",
                "feature_aggregate_stddev_positive_overpressure",
                "feature_aggregate_total_positive_overpressure",
                "feature_aggregate_min_negative_overpressure",
                "feature_aggregate_avg_negative_overpressure",
                "feature_aggregate_stddev_negative_overpressure",
                "feature_aggregate_total_negative_overpressure",
                "feature_aggregate_max_all_overpressure",
                "feature_aggregate_avg_all_overpressure",
                "feature_aggregate_stddev_all_overpressure",
                "feature_aggregate_total_all_overpressure",
                "feature_aggregate_max_zero_crossing_time",
                "feature_aggregate_avg_zero_crossing_time",
                "feature_aggregate_stddev_zero_crossing_time",
                "feature_aggregate_total_zero_crossing_time",
                "feature_subject_age",
                "feature_relationship_status",
                "feature_disease",
                "feature_medprob",
                "feature_meningitis",
                "feature_concussion",
                "feature_headaches",
                "feature_ringears",
                "feature_deafness",
                "feature_dischargingears",
                "feature_fullears",
                "feature_nosesinusthroat",
                "feature_fainting",
                "feature_dizziness",
                "feature_coordination",
                "feature_trvlsick",
                "feature_psychological",
                "feature_depanxstress",
                "feature_rxndrug",
                "feature_backache",
                "feature_avoid",
                "feature_memory"
                ],
        "targets": [
                "target_sleep_symptom_follow_up"
                ]
}
```

## List of Symbols, Abbreviations, and Acronyms

| | |
|---|---|
| AI | artificial intelligence |
| BOS | blast overpressure studies |
| CSV | comma separated value |
| DANA | defense automated behavioral assessment |
| DO | drop out |
| ETL | extract, transform, and load |
| FDR | false discovery rate |
| GFAP | glial fibrillary acidic protein |
| GNG | go-no-go |
| IQR | inter-quartile range |
| LM | Levenberg-Marquardt |
| mTBI | mild traumatic brain injury |
| Nf-L | neurofilament light chain |
| PRT | procedural reaction time |
| ROUT | robust regression and outlier removal |
| SQL | standard query language |
| SRT | simple response time |
| TBI | traumatic brain injury |
| UCH-L1 | ubiquitin carboxy-terminal hydrolase L1 |
| UI | user interface |
| WRAIR | Walter Reed Army Institute of Research |

|  |  |
|---|---|
| 1<br>(PDF) | DEFENSE TECHNICAL<br>INFORMATION CTR<br>DTIC OCA |
| 1<br>(PDF) | DEVCOM ARL<br>FCDD RLD DCI<br>TECH LIB |
| 1<br>(PDF) | DEVCOM ARL<br>FCDD RLC S<br>D SHIRES |