Predicting Worker Fatigue Using Machine Learning

Team 3

Swapnil Junarrkar

Husna Kalim

Akshat Kothari

Toma La Cara

Mohamed Zidan

Table of Contents

Executive Summary

Machine learning is becoming more and more prominent in our world today, influencing every global industry. Our projects focus is on the safety industry as we created a machine learning model that can predict whether an individual can carry out certain tasks. Our machine learning model paired with the Muse 2 device will provide a cost-effective solution for when we need to determine whether an individual is feeling fatigued. Our data collection process followed up by an in-depth data preprocessing procedure that consisted of 4 different approaches made it easier for us to tune our machine learning model. In the end, we were able to reach an accuracy score of 99.84% when differentiating between fatigue vs normal. By improving on our current work, we can expand the applications of our machine learning model to countless other industries. With that being said, this project should be used as a steppingstone to create a reliable solution that cannot just improve efficiency within companies but prevent serious injuries to people and their environments.

Problem

Mental fatigue deteriorates the ability to perform daily activities. This is made even more apparent when you consider dangerous jobs or vocations where physical fatigue can be hazardous to workers and people around them. This problem is further amplified when you consider there is currently no viable way to predict someone's "tiredness" or "fatigue" reliably. For example, if a construction worker or pilot fulfills their duties without adequate sleep, their actions can put a lot of others at risk.

There are some solutions that exist in today's market; however, they are either too expensive, time consuming or not practical. Some of these solutions include eye tracking devices for truck drivers such as PERCLOS, studies and brain games for employees such as the Oxford Sleep Resistance Test (OSLER test), and the Psychomotor Vigilance Test (PVT).

For eye tracking solutions, they can be easily tricked by the subject as they use external stimuli to measure fatigue. Once the subject has understood what makes the device go off, they can consciously control their body's movements to cheat the system and its sensors. On the other hand, the tests and studies done on employees are only conducted over a long period of time, take extremely long to do for each employee, and is not a very subjective way to measure fatigue as people's resilience to it can differ from one another. Therefore, we decided to utilize an EEG approach and build a predictive model to measure fatigue.

Solution

To tackle this imperative issue, our team developed a machine learning model that can predict the state of the brain based on EEG activity. Our model has been trained on both "Normal" or fresh brain waves as well as "Fatigued" or tired brain waves and is able to predict the difference between them. Our solution also does away with a lot of the limitations described in the above section as it is not reliant on external stimuli against which a user can be trained to deceive it. It is also extremely fast and objective in its predictions.

This model has numerous real-world solutions that can help improve the quality of life for the users as well as their safety and the safety of their environment. Applications for this product include elevated risk jobs such as construction, long haul drivers such as truck drivers and pilots, and even everyday commercial applications such as firms who want to monitor the health of their employees. Additional details on how this model was created and how we achieved the accuracy will be discussed in the sections below.

Data Collection

To collect our datasets, we used the Muse 2 headband in conjunction with the Mind Monitor app. Once the participant has the Muse device setup on their head, it automatically sends real-time data to the app, which can then be recorded and uploaded online. As mentioned previously, we want our model to be able to predict whether someone is fatigued, or not, and to narrow down our scope for this project we decided on collecting fatigued data post exercise, and not fatigue(normal) data 1-2 hours after someone has woken up. Data collection was done by 2 individuals, and over a course of 2 days where 1 set of fatigued and normal data were collected each day. We also standardized the duration for which we collected the data to 5 minutes, and the individual had to be doing nothing else for that period. These restrictions and standardizations were implemented for us to control/limit the number of variables that could influence our data.

From the mind monitor app, the recorded data is automatically converted into a csv file which made it easy for us to import it into jupyter notebook. In total, we collected 8 sets of data (4 fatigue, 4 normal), which equated to 4129 rows of datapoints. An example of the files is shown below; datapoints are recorded roughly every second and there are 38 features, some of which are of no use to us for our model.

In terms of challenges, after the first set of data was collected, we noticed that the headband would stop collecting data for the period that the user had their jaws clenched or eyes closed. This resulted in missing values within our datasets; however, we were still able to collect enough data points within the 5 minutes that we decided to simply keep those datasets and not redo the experiment.

Data Preprocessing

Four different approaches were used to pre-process the eight collected datasets before combining all the pre-processed datasets into 4 large datasets for further training and testing using machine learning models. The figure below shows an overview of this process.

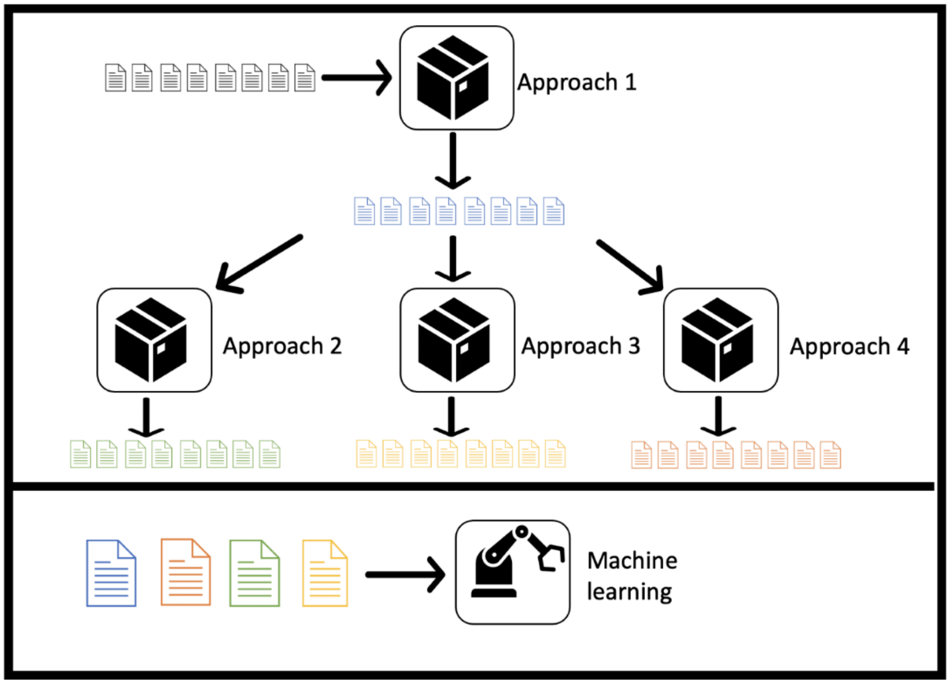

Figure 1: Big picture of the data pre-processing and machine learning stages

Approach 1: Outlier Detection and Removal and Imputation of Missing Values

This approach pre-processes the eight collected datasets with twenty features each by applying Gaussian mixture-based outlier detection and removal algorithm, followed by replacing the missing values resulting from the outlier algorithm using a linear interpolation-based imputer.

Also, three distributions of a Gaussian mixture model are weighted to maximize the product of the probabilities of observing attribute values. A data point would be classified as an outlier if the probability of occurring in the fitted combination of distributions is lower than 0.0005.

The figure below shows a sample of a dataset pre-processed using this approach.

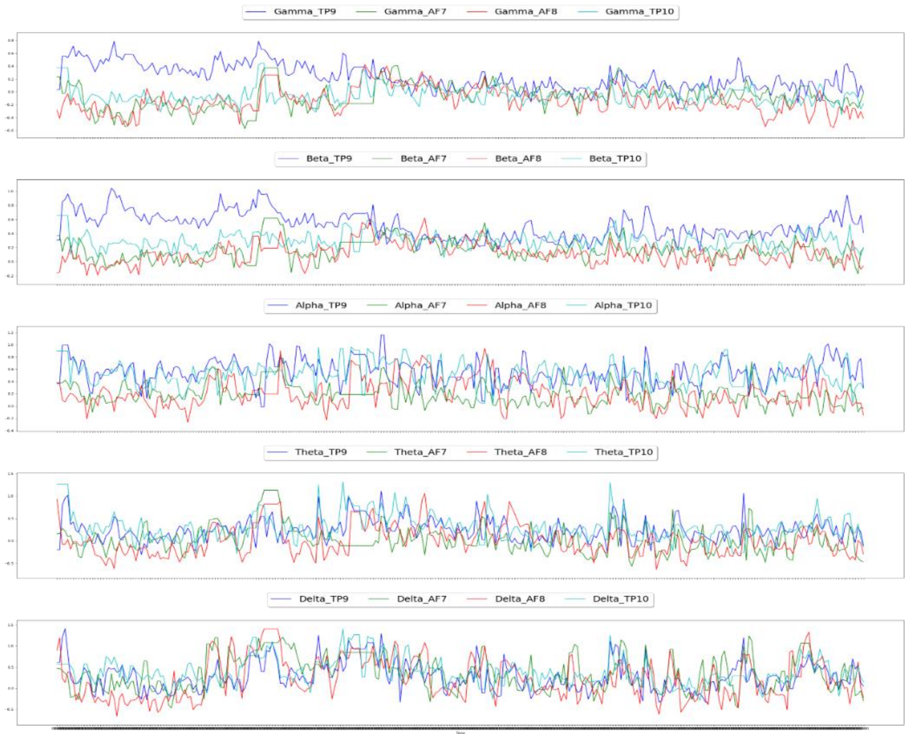

Figure 2: dataset pre-processed using Approach 1

Approach 2: Principal Component Analysis (PCA)

PCA is often used to reduce the dimensions of the data to make it possible to visualize data in 2D or 3D. However, this approach uses PCA as a feature engineering tool to add 5 PCA feature components to each dataset obtained using Approach 1. These added PCA features components could be useful for the machine learning binary classification task. With this approach, the total number of features was increased to 25.

Also, the elbow method was used to select how many PCA feature components to add to each dataset. Explained variance ratios were calculated for each one of the eight collected datasets and then the elbow method was applied to each dataset with the most common value selected. The figure below shows the explained variance ratios for the eight collected datasets.

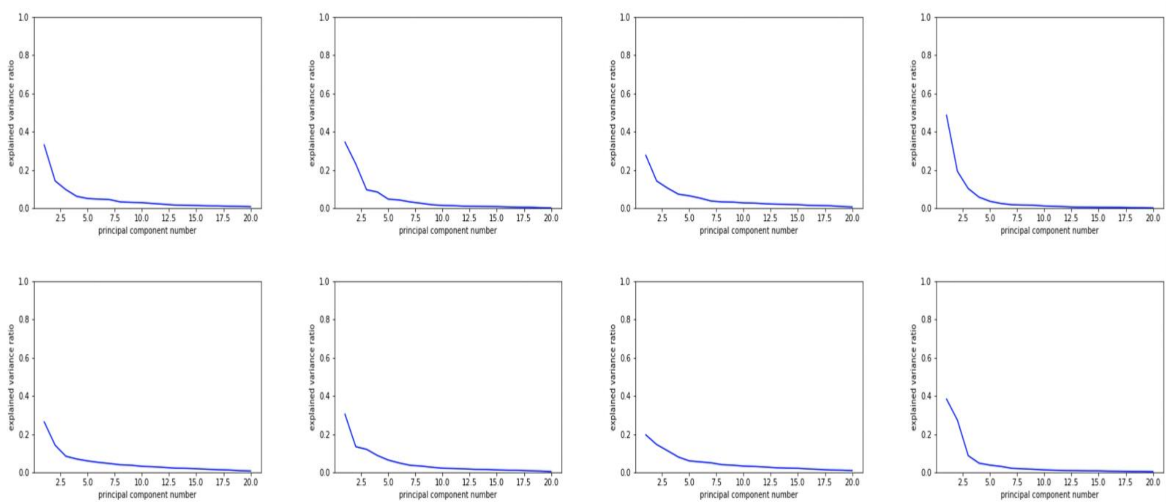

Figure 3: Explained variance ratios for the eight collected datasets

The figure below shows a sample of a dataset pre-processed using this approach.

Figure 4: dataset pre-processed using Approach 2

Approach 3: Fast Independent Component Analysis (Fast ICA)

Similar to Approach 2, Approach 3 adds 5 ICA features components to each dataset obtained using Approach 1. The reason why ICA was selected is that it's an advanced version of PCA and it's usually used for raw EEG artifact removal. With this approach, the total number of features was increased to 25. The figure below shows a sample of a dataset pre-processed using this approach.

Figure 5: dataset pre-processed using Approach 3

Approach 4: Independent Component Analysis (ICA) and Time and Frequency Domain Feature Extraction

In addition to Fast ICA, this approach also extracted time and frequency domain features. In particular, 5 Fast ICA feature components, 120 temporal-based features, and 100 frequency-based features, were added to each one of the eight datasets obtained using Approach 1. With this approach, the total number of features was increased to 245.

The process that was used to obtain the time and frequency domain features is summarized as follows:

A dataset is divided into windows. For each window, several metrics in the time and frequency domains are calculated. These metrics summarize the datapoints in that specific window. Then, only one point is kept per window. Even though this process adds a significant number of new features, it decreases the total number of datapoints in a dataset. To address this issue, an overlap of 50% is allowed between windows, which are all of sizes of 2 seconds.

The table below summarizes the time and frequency domain features obtained with each window, and the figure below shows a sample of a dataset pre-processed using this approach.

Feature	Domain	Description
Mean	Time	Mean value among the datapoint values in a dataset
median	Time	Median value among the datapoint values in a dataset
Minimum	Time	Minimum data point value in a window
Maximum	Time	Maximum data point value in a window
Standard deviation	Time	Standard deviation based on the datapoints in a window
Slope	Time	Line slope when fitting a linear regression to the data points in a window
Maximum frequency	Frequency	The frequency with the highest amplitude. This metric provides an indication of the most important frequency in the windows under consideration.
Frequency weighted signal average	Frequency	This metric provides information on the average frequency observed in the window (given the amplitudes) and might shed a light on the entire spectrum of frequencies. It is calculated by multiplying the frequencies with their amplitude and normalizing them by the total sum of the amplitudes.
Power spectral entropy	Frequency	This metric represents how much information is contained within the signal. In other words, the power spectral entropy determines whether there are one or a few discrete frequencies standing out of all others. To calculate the metric, the power spectral density is first calculated (squaring the amplitude and normalizing by the number of frequencies), normalize the values to a total sum of 1 such that we can view it as a probability density function and compute the entropy via the standard entropy calculation.

Table 1: Time and frequency domain features with each window [2]

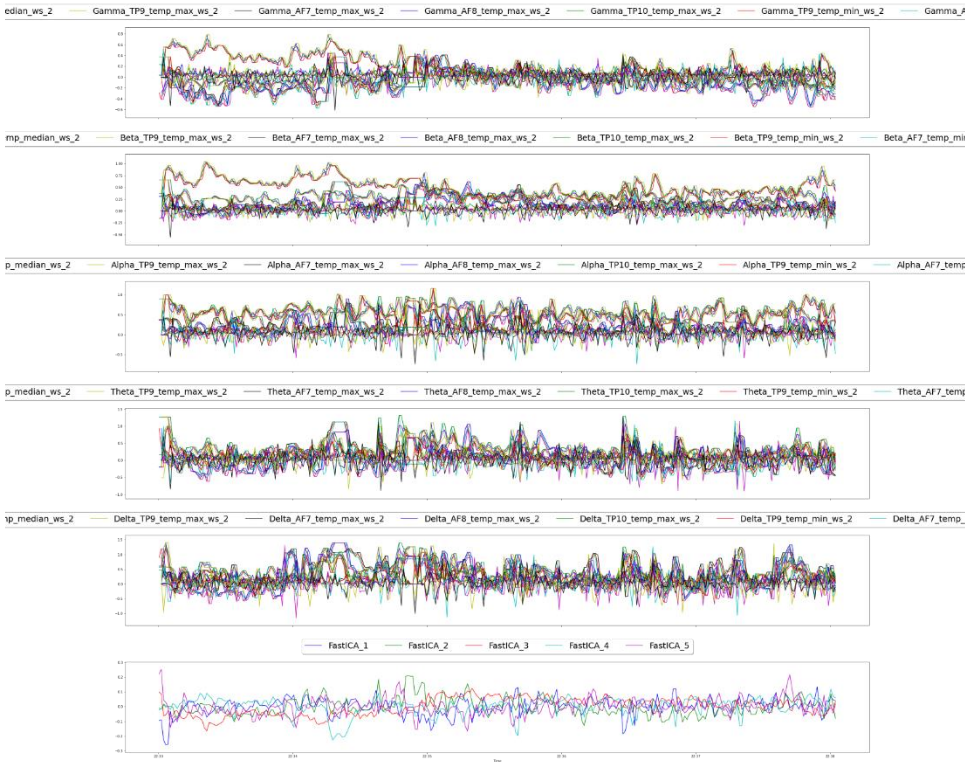

Figure 6: Some of the new time-domain features of a dataset pre-processed using Approach 4

Building the Model

To build our Machine Learning model, we first used the AutoML package [1] on all 4 pre-processed datasets mentioned above to determine which set of data gives us the highest accuracy. The complete AutoML pipeline usually consists of data preprocessing, feature engineering, feature selection, model training, hyperparameter tuning, and algorithm selection. This package uses various ML algorithms, namely Baseline, Linear, Random Forest, Decision Tree, XGBoost, and Neural Network. It performs hyper-parameter tuning to find the best model and provides a detailed Markdown report for each model. A summarized version of the AutoML evaluations for the various datasets are given in the table below:

Preprocessing Approach	AutoML Best Model	Accuracy	Time
Standard Data Preprocessing	XGBoost	94.76%	4.12s
PCA	Neural Network	99.12%	1.16s
Fast ICA	XGBoost	96.07%	4.43s
Fast ICA and Temporal Frequency Extraction	XGBoost	95.29%	27.76s

Table 2: Summary of AutoML evaluations

Based on the above evaluations, we see that the highest accuracy was achieved with the PCA data preprocessing, using the Neural Network Deep Learning Model (99.12%). We then used TensorFlow to build our ANN Classifier model, with 25 input parameters, and 16 parameters in the hidden layer, based on the Markdown report provided by AutoML. We computed the model's confusion matrix to evaluate the accuracy of the classification and got a 99.84% accuracy.

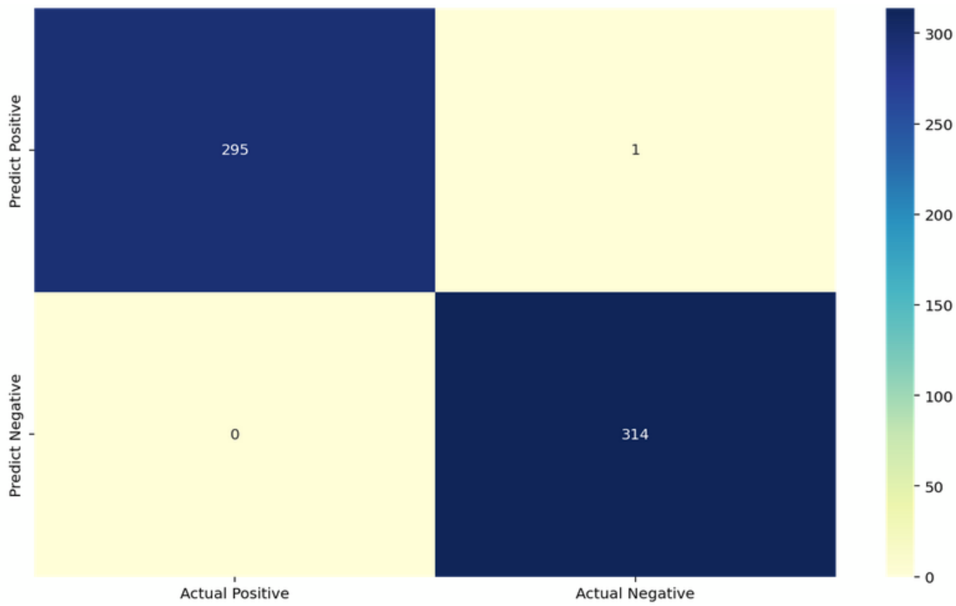

Figure 7: Confusion matrix

Challenges

It is difficult to differentiate between fatigue and normal data.

The collected EEG signals are amplified, digitized, and then sent to a computer or mobile device for storage and data processing. But there is no proper nomenclature on the data stored to differentiate between the different states of the brain unless the person himself explains or the data is labelled before any other process begins.

Alignment issues of Muse device(headband) leading to null values.

Muse devices must be worn very tightly such that it is intact with our scalp throughout the data collection process. But every person does not have the same shape of the head, even density of hairs on the scalp and type of skin(oily/dry), etc. as these features vary from person to person. So, it may tend to slip down of device leading to null values during the data collection process and it cannot be accomplished on the same baseline for two different people.

Data fetching issues with involuntary activities like blinking of eyes etc. The device also records when you move, fidget, or open your eyes.

As we know Muse is a wearable brain sensing headband. The device measures brain activity via 4 electroencephalography (EEG) sensors. As the EEG scans are performed by placing EEG sensors on our scalp. They pick up and record the electrical activity in our brain that is not limited to the state of the brain, like if a person is tired and tries to know his state of mind while grabbing some snacks, the device may not give accurate results. The results get varied with such small activities and cause ambiguity and misinterpretation of the brain.

The brain sensing is not 100 percent error free.

The primary purpose of the device is to have complete freedom and know the state of the brain, but when we use the device to collect the data, we are subconsciously aware and cautious about the device during the data collection process, which may result in unreliable signal generation and processing them may produce erroneous results.

Future Work

Collect more data

Our dataset is formed by capturing 5 minutes of our brain waves after some activity at two different conditions (normal and fatigue) by different individuals and finally concatenated it. We can expand our dataset by collecting data for a longer time duration and increase the number of sample data by increasing the number of people. Through this, we can increase the accuracy and reliability of the test.

Real time Predictions

The application of this device is way beyond just determining whether normal or a fatigue condition. This device with minimal enhancements can be used in many fields like education, manufacturing industry, aviation, transport, etc. For instance, it can be used in the screening of people who are in the effect of Marijuana.

Conclusion

Through this project, we intended to bring a solution to the problem where mental fatigue can cause a massive loss and could be destructive to humans and their surroundings. To resolve this, it required us to distinguish between the diverse state of mind in a cost-effective, transparent, effortless, and user-friendly way. This could be achieved using the Muse 2 device and mind monitor app. Using this we arrived with a solution to deal with problems associated with the diverse state of the brain and distinguish between normal condition and mental fatigue. We implemented different ML algorithms and data processing approaches on the dataset collected under different conditions like after physical exercise, two hours after waking up. The data processing approaches include Outlier Detection and Removal and Imputation of Missing Values, Principal component analysis (PCA), Fast Independent Component Analysis (Fast ICA), and Time & Frequency Domain Feature Extraction. Finally, though tuning our machine learning model, we got an accuracy of 99.84% using the ANN classifier and the PCA data.

As a future scope of work, we can collect more data and we wish to investigate the impact on performance due to the variations in a dataset collected in diverse conditions and the possibility of using the device in real-time predictions.

References

[1] https://github.com/mljar/mljar-supervised

[2] https://link.springer.com/book/10.1007/978-3-319-66308-1