Supervised Machine Learning for Detecting Intoxication through Smartphone Accelerometer Data

Stelios Zarifis

15 Φεβρουαρίου 2024

Abstract

Alcohol misuse is a global health concern, contributing to 5.1% of the worldwide disease burden and resulting in 3 million deaths every year. In people aged 20–39 years, approximately 13.5% of total deaths are attributable to alcohol [1]. This project aims to address this issue by developing a novel application that uses supervised machine learning techniques to predict intoxication levels in real-time, based solely on smartphone accelerometer data. I examined various Deep Learning models and architectures......

1 Introduction

The misuse of alcohol is a pervasive issue that poses significant health risks and societal challenges. Current interventions for heavy drinking have limitations, particularly in their timing and effectiveness. This project is motivated by the need for real-time, effective interventions that can be seamlessly integrated into individuals' daily lives.
The objective is to develop a real-time application that predicts intoxication levels using smartphone accelerometer data. This approach leverages the ubiquity of smartphones and the rich, non-sensitive data they can provide. The project employs supervised machine learning techniques as a creative approach to tackle the problem of heavy drinking detection.
I have tested various deep learning architectures, including baseline neural networks, Convolutional Neural Networks (CNN), Long Short-Term Memory networks (LSTM), and a combination of CNN and LSTM. I also explored various hyperparameters to optimize the models.
The potential impact of this project is significant. Achieving high accuracy in detecting heavy drinking episodes can contribute to more effective interventions, potentially reducing the health risks and societal costs associated with alcohol misuse. My work builds upon previous research in this area, but distinguishes itself through its focus on real-time detection and its innovative use of machine learning techniques.
The real-time prediction of intoxication levels has numerous practical applications, from personal health monitoring to public safety initiatives. While our current focus is on detecting heavy drinking episodes, future extensions of our project could incorporate additional features or data sources to enhance model accuracy.

2 Related Work

Several studies have explored the use of smartphone accelerometer data for detecting heavy drinking episodes. For instance, the paper "Learning to Detect Heavy Drinking Episodes Using Smartphone Accelerometer Data" by Killian et al. [2] developed a reliable mobile classifier that uses only non-sensitive accelerometer data to detect periods of heavy drinking. They tested several models and achieved a test accuracy of 77.5% with a random forest.
Another study titled "Alcohol Detection over Long Periods Using Smartphone Accelerometer Data" by Gil-Martín et al. [3] proposed a motion biomarker for alcohol detection using a deep learning approach that processes inertial signals recorded with a smartphone. Their deep learning architecture included three convolutional layers for learning features from the inertial signal spectrum, and several fully connected layers to perform classification and regression tasks.
Matteo Gadaleta et al. [4] have also contributed to this field with their work titled "IDNet: Smartphone-based Gait Recognition with Convolutional Neural Networks". They used a Convolutional Neural Network (CNN) model on accelerometer and gyroscope (inertial) signals. Each axis of the accelerometer data was fed into separate convolutional layers, pooling layers, then concatenated before being interpreted. This approach demonstrates the potential of using deep learning architectures for analyzing accelerometer data.
Ming Zeng et al. [5], in their 2014 paper "Convolutional Neural Networks for Human Activity Recognition using Mobile Sensors", also used a CNN model for accelerometer data. Their work further underscores the effectiveness of using deep learning techniques for activity recognition based on mobile sensor data.

The dataset used in this project is the ”Bar Crawl: Detecting Heavy Drinking” dataset, which is publicly available on the UCI Machine Learning Repository [6]. The dataset was first used in a research paper by Jackson A Killian et al. [2]. It contains accelerometer and transdermal alcohol content data from a college bar crawl, used to predict heavy drinking episodes via mobile data. The data was originally collected from 19 participants, but the TAC readings of 6 participants were deemed unusable.

3 Dataset and Features

Dataset Information The dataset used in this project is titled ”Bar Crawl: Detecting Heavy Drinking” and is publicly available on the UCI Machine Learning Repository. The data was originally collected from 19 participants, but the TAC readings of 6 participants were deemed unusable. The data included is from the remaining 13 participants, comprising $715 TAC$ readings and $14,057,567$ accelerometer readings.

Additional Information The study decomposed each time series into 10-second windows and performed binary classification to predict if windows corresponded to an intoxicated participant ($TAC \geq 0.08$) or sober participant ($TAC < 0.08$). The study tested several models and achieved a test accuracy of 77.5% with a random forest.

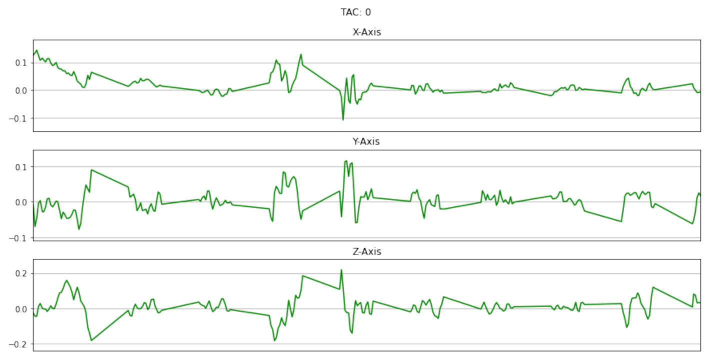
Relevant Information All data is fully anonymized. Accelerometer data was collected from smartphones at a sampling rate of $40Hz$. The file contains 5 columns: a timestamp, a participant ID, and a sample from each axis of the accelerometer. Data was collected from a mix of 11 iPhones and 2 Android phones. TAC data was collected using SCRAM ankle bracelets and was collected at 30-minute intervals.

Number of Instances The dataset includes $14,057,567$ accelerometer readings, 715 TAC readings, and data from 13 participants.

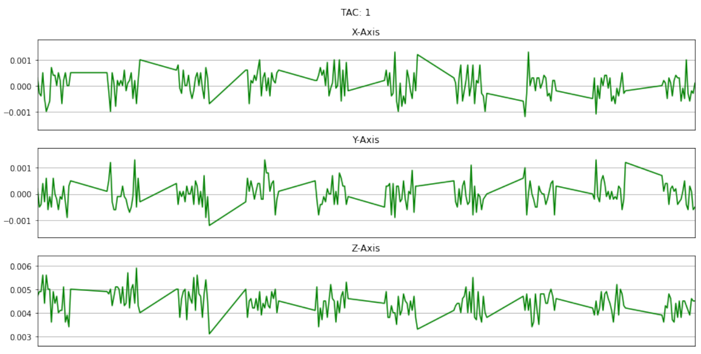
Number of Attributes The dataset includes time series data from 3 axes of accelerometer data, 1 phone-type feature, and 1 time series of TAC for each of the 13 participants.

Target Distribution TAC is measured in g/dl where 0.08 is the legal limit for intoxication while driving. The mean TAC is 0.065 ± 0.182 and the maximum TAC is 0.443.

The following figures illustrate the differences in accelerometer patterns between a sober and an intoxicated individual.



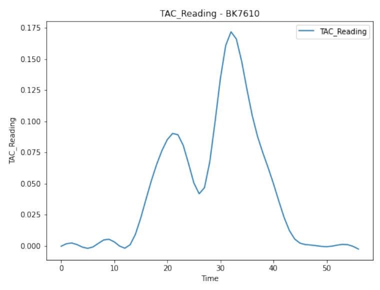
(α´) Accelerometer time series for a sober person.



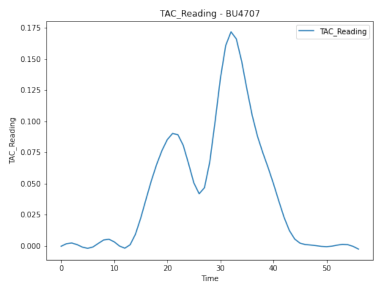
(β´) Accelerometer time series for an intoxicated person.

Figure 1: Comparison of accelerometer time series for sober person 1α´ and intoxicated person 1β´.

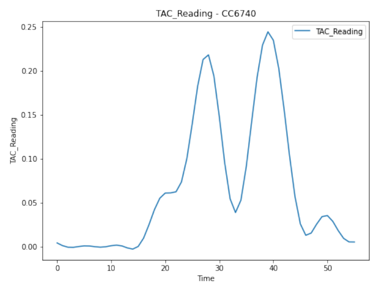
Gaussian Distribution of TAC Values Another interesting observation in the dataset is the distribution of TAC values. As shown in the Figure below, TAC values exhibit a Gaussian distribution (mixture of Gaussians over time). This characteristic can be advantageous in the classification process, providing a clear representation of the separation between sober and intoxicated states.



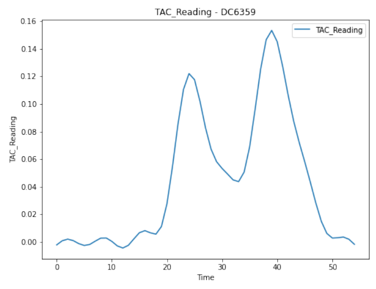
(α´) BK7610



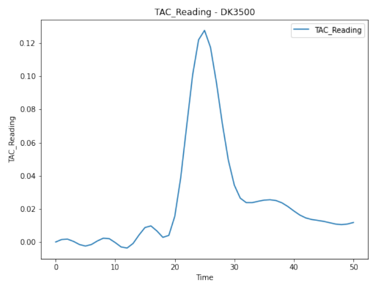
(β´) BU4707



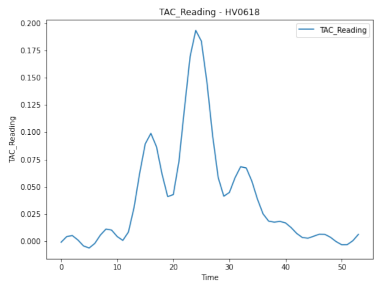
(γ´) CC6740



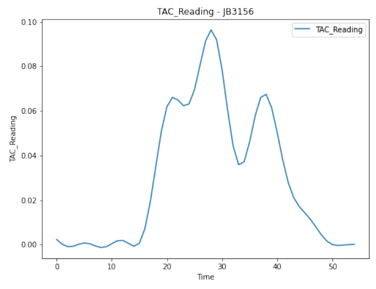
(δ´) DC6359



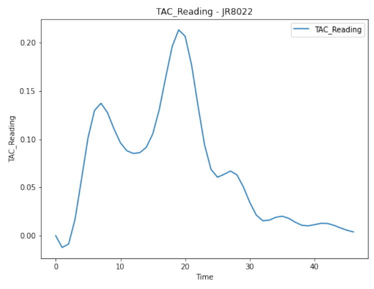
(ε´) DK3500



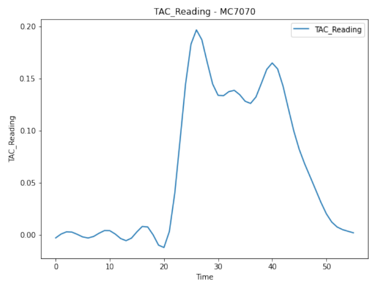
(στ´) HV0618



(ζ´) JB3156



(η´) JR8022



(θ´) MC7070

Figure 2: Time series of TAC values for 9 of the participants in a 3 $\times$ 3 grid.

4 Data Preprocessing

Time Unit Conversion Initially I converted the recorded accelerometer samples from milliseconds to seconds. This conversion enabled a standardized time unit for subsequent processing steps.

Class Creation Two classes are established to define the target variable for the machine learning model. '1' denotes intoxication when the TAC value is greater than or equal to 0.08, and '0' indicates sobriety when TAC is below 0.08.

Unique Timestamps and Refinement Then I extracted the unique timestamps and refined the dataset in order to include only the first 20 instances of accelerometer measurements per second. This step simplifies the dataset while retaining essential temporal information.

Dataset Segmentation The dataset is then divided into subdatasets, distinguishing between sober and intoxicated instances. This step will be useful in the Class Balancing procedure, later.

Column Selection To optimize the dataset for model training, irrelevant columns were dropped. Only relevant features, including `tac` values, `time_acc` timestamps, `x`, `y`, and `z` measurements were retained.

Class Balancing Addressing class imbalance, a balanced dataset was created to ensure an equal number of samples for both sober and intoxicated instances. This step prevents bias toward the majority class during model training. Imbalances may result in the model being proficient at predicting instances of the prevalent class while neglecting the minority class. Creating a balanced dataset ensures that the model learns patterns and features representative of both sober and intoxicated instances, contributing to a more robust model.

Feature-Target Separation The dataset is then divided into features (`X`) and targets (`y`), setting the stage for subsequent model training.

Z-Score Standardization To standardize features, z-score standardization was employed, involving the removal of the mean and scaling to unit variance. The process involves rescaling the features such that they have the properties of a standard normal distribution with $\mu = 0$ and $\sigma = 1$, where μ is the mean and σ is the standard deviation from the mean. The formula for calculating the z-score of a value x in a distribution is:

$$z = \frac{x - \mu}{\sigma}$$

The primary purpose of z-score standardization is to make different features comparable and to ensure that no particular feature dominates others due to differences in their scales.

Train-Test Split The dataset is split into train set and test set (80% - 20%, respectively). This separation allows the evaluation of the model on unseen data to ensure that it can generalize well and has not merely memorized the training set.

5 Methodology

In my analysis of the Bar Crawl dataset, I employed a diverse set of machine learning models to effectively capture and predict heavy drinking episodes based on accelerometer data. I started with classic models such as Multi-Layer Perceptrons (MLPs) and Random Forests, leveraging their flexibility and scalability. Subsequently, I explored ensemble models like AdaBoost, benefitting from their adaptive learning and robustness against overfitting. Gaussian Naive Bayes (GNBs) provided a probabilistic approach, assuming normal distribution of features and Support Vector Machines (SVMs) proved effective in capturing non-linear relationships. Moving towards more specialized models for sequential data, Convolutional Neural Networks (CNNs) and Long Short-Term Memory Networks (LSTMs) performed well in capturing long-term dependencies in sequential data.

5.1 Multi-Layer Perceptrons (MLPs)

Multilayer Perceptrons (MLPs) or feedforward neural networks, can be a suitable choice for analyzing complex data for several reasons:

Universal Approximation Theorem: MLPs are capable of approximating any continuous function given a sufficient number of hidden units. This means they have the potential to model complex patterns in the data.

Flexibility: MLPs do not make strong assumptions about the underlying data distribution, making them a flexible choice for our real-world data which do not necessarily follow a specific distribution.

End-to-End Learning: MLPs learn to map inputs to outputs which means they can automatically learn useful representations from raw data without the need for manual feature engineering.

Non-Linearity: MLPs introduce non-linearity into the model through activation functions. This allows them to capture non-linear relationships in the data, which are important when modeling complex data.

Scalability: MLPs can be scaled to handle large datasets and high-dimensional input spaces.

5.1.1 Data Preparation

Cleaning: Several columns that were not necessary for the analysis were removed from the data frame. These columns are `pid`, `window10`, `win_10_x_FFT_spectral_centroid_spread`, `win_10_y_FFT_spectral_centroid_spread`, `win_10_z_FFT_spectral_cen`
`x_FFT_spectral_centroid_spread`, `y_FFT_spectral_centroid_spread`, `z_FFT_spectral_centroid_spread` and `key`. Also, rows with missing values were dropped.

5.1.2 Model Architecture

The first MLP architecture I implemented was the default of the sklearn library, with 100 neurons in 1 hidden layer. Then, I performed randomized grid search with various architectures:

- Hidden sizes: $\{(50,),(100,),(50,50),(100,50),(100,100),(200,100),(100,100,50),(100,50,50)\}$
- Activation functions: $\{\text{ReLU}, \tanh\}$
- Learning Rates $\{0.001, 0.0001\}$

5.1.3 Model Training and Results

For the training, I performed 10-fold cross validation. The resulting accuracies are shown in the following table. The average accuracy is 71.71%, which is a good metric of the model's performance.

5.1.4 Further Model Training

As discussed, various hyperparameters were tested with the grid search method but no superior model architectures were revealed, indicating the efficacy of the initial choice. In the following table the accuracy of the best models of the grid search is shown:

5.2 Random Forests Classifier

Random Forests are a powerful and versatile machine learning method that can be a good choice for analyzing data for several reasons:

Fold	Accuracy
1	59.14%
2	84.94%
3	81.98%
4	90.98%
5	94.62%
6	49.36%
7	26.57%
8	63.52%
9	71.88%
10	94.06%

Table 1: 10-Fold Cross-Validation Accuracies for MLP with 100 Neurons in 1 Hidden Layer

Hidden Layers	Accuracy
(100, 100)	71.70%
(200, 100)	71.70%

Table 2: Best models' Accuracies after the Grid Search

Handling of High Dimensionality: Random Forests can handle high-dimensional spaces as well as large numbers of training examples, making them suitable for complex datasets.

Robustness to Overfitting: By constructing multiple decision trees and outputting the mode of the classes for classification, Random Forests mitigate the risk of overfitting which is a common problem in single decision tree models.

Feature Importance: Random Forests provide measures of feature importance, which is beneficial in understanding which features are driving the prediction. This can be useful for real-time predictions, where feature importance can be used to prioritize calculations and reduce computational overhead.

Non-Linearity and Interaction Effects: Random Forests can model non-linear decision boundaries thanks to their hierarchical structure.

5.2.1 Data Preparation

The data perparation was the same as for the MLP.

5.2.2 Model Architecture

The first Random Forest model I tested had 8 nodes maximum depth and 100 estimators. After that, I performed randomized grid search with the following architectures:

- Number of estimators: $\{50, 100, 200\}$
- Maximum depth: $\{4, 8, 16\}$
- Minimum samples split: $\{2, 5, 10\}$

5.2.3 Model Training and Results

The accuracy of the first model (Depth: 8, Estimators: 100) is 85.72%. We observe that the Random Forest Classifier shows dramatic improvement in performance over the MLPs.

5.2.4 Further Model Training

After training for different hyperparameters with the grid search method I showcase the best models I trained in the following table: Notably, the Random Forest model achieved an outstanding accuracy of 99.51% on unseen data, indicating its suitability and power as a classifier for our application. The exceptional performance of the model can be attributed to the consistent behavior of the human body across participants. The distribution of TAC values seems to follow a normal distribution (as seen in Figure 2), enabling the model to easily discern the characteristics responsible for accurate predictions. This uniformity in physiological responses enhances the model's ability to generalize well to unseen data and capturing relevant features for predicting intoxication levels.

Random Forest Model	Accuracy
Depth: 8, Estimators: 100	85.72%
Depth: 16, Estimators: 200	99.51%

Table 3: Best Models' Accuracies after the Grid Search

5.3 AdaBoost Classifier

AdaBoost, short for Adaptive Boosting, is a powerful ensemble machine learning algorithm known for its efficiency and high level of accuracy. Here are some reasons why it could be a good choice for analyzing the Bar Crawl data:

Combination of Weak Learners: AdaBoost works by combining multiple weak learners to create a strong learner. This makes the model robust and improves the overall predictive accuracy.

Adaptive Learning: AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. This means it places more emphasis on the instances it has trouble with, improving the performance over iterations.

Handling of Overfitting: AdaBoost avoids overfitting because it combines a lot of weak learners, each of which is only slightly better than random guessing.

No Need for Feature Scaling: AdaBoost, like other tree-based methods, does not require feature scaling to work properly, simplifying the data preprocessing steps.

Feature Importance: Like Random Forests Classifier, AdaBoost can provide estimates of feature importance from the trained model, which is beneficial in understanding which features are driving the prediction.

5.3.1 Data Preparation

The data perparation was the same as for MLP and Random Forest.

5.3.2 Model Architecture

The first AdaBoost architectures I explored were with 50 and 100 estimators. Then, I tried randomized grid search with the following architectures:

- Number of Estimators: $\{50, 100, 150, 200\}$
- Learning Rate: 0.001

5.3.3 Model Training and Results

For the training, I performed 10-fold cross validation. The resulting accuracies are shown in the following table. The average accuracy for the model with 50 estimators is 67.82% and the one with 100 estimators is 65.90%.

Fold	Accuracy
1	0.4693
2	0.8790
3	0.7432
4	0.7420
5	0.8961
6	0.5245
7	0.4047
8	0.6603
9	0.6919
10	0.7705

Table 4: 10-Fold Cross-Validation Accuracies for Adaboost Model with 50 Estimators

Fold	Accuracy
1	0.4847
2	0.8933
3	0.7394
4	0.7715
5	0.8951
6	0.5259
7	0.4028
8	0.6266
9	0.7097
10	0.5412

Table 5: 10-Fold Cross-Validation Accuracies for Adaboost Model with 100 Estimators

5.3.4 Further Model Training

Also, more hyperparameters were tested with the grid search method but no superior model architectures were revealed:

Fold	Accuracy
1	51.84%
2	89.92%
3	70.53%
4	76.22%
5	89.41%
6	51.55%
7	43.43%
8	64.00%
9	70.45%
10	38.49%
Average Accuracy	64.58%

Table 6: Best model's Accuracy after the Grid Search

5.4 Gaussian Classifiers (GNBs)

A Gaussian Classifier (Gaussian Naive Bayes) is a probabilistic model that is often used for simple classification tasks:

Assumption of Normal Distribution: This Classifier assumes that the features follow a normal distribution. This is often reasonable as many real-world data do follow normal distributions.

Efficiency: Gaussian Classifier is computationally efficient, making it suitable for high-dimensional data and large datasets.

Robustness to Irrelevant Features: In practice, Gaussian Classifier is quite robust to irrelevant features. This means that if the dataset contains features that are not useful for predicting the targets, they will have less of an impact on the performance.

5.4.1 Data Preparation

The data perparation was the same as for MLP, Random Forest and AdaBoost.

5.4.2 Model Architecture

For the Gaussian Naive Bayes classifier, the architecture involves using just the Gaussian distribution to model the likelihood of features given the class.

5.4.3 Model Training and Results

Again, I used 10-fold cross validation and the results are shown below. The average accuracy is low (29.98%), as expected, because the Gaussian classifier is a very simple predictor, unable to model complex relationships.

Fold	Accuracy
1	57.73%
2	15.05%
3	18.02%
4	9.00%
5	5.38%
6	50.64%
7	73.43%
8	36.49%
9	28.12%
10	5.95%

Table 7: 10-Fold Cross-Validation Accuracies for Gaussian Naive Bayes Model

5.5 Support Vector Machines (SVMs)

A Support Vector Machine (SVM) is a supervised machine learning algorithm, used for both classification and regression tasks:

Effective in High Dimensional Spaces: SVMs are particularly effective in high dimensional data, like in our case.

Maximal Margin Classifier: SVMs are maximal margin classifiers which handle both linearly separable and non-linearly separable data. They find the hyperplane that maximizes the margin between classes, which often leads to better generalization.

Kernel Trick: SVMs can also perform non-linear classification using the kernel trick, mapping their inputs into high-dimensional feature spaces. This allows the algorithm to model complex relationships in data.

Robustness: SVMs are robust against overfitting, as they classify using the maximal margin hyperplane, especially in high-dimensional spaces.

Sparse Solution: The solution of the SVMs depends only on support vectors which are a subset of the data, making the model memory efficient.

5.5.1 Data Preparation

The data perparation was the same as for MLP, Random Forest, AdaBoost and GNB.

5.5.2 Model Architecture

Firstly I employed a simple SVM. After that, I tried using a Gaussian Kernel (radial basis function - RBF), to extend the basic SVM by mapping input data into a higher-dimensional space.

5.5.3 Model Training and Results

Again, utilizing 10-fold cross-validation, the performance of the Support Vector Machine (SVM) model is summarized below. The average accuracy across folds is 71.56%.

Fold	Accuracy
1	57.66%
2	84.95%
3	81.98%
4	91.00%
5	94.62%
6	49.36%
7	26.57%
8	63.51%
9	71.88%
10	94.06%

Table 8: 10-Fold Cross-Validation Accuracies for SVM Model

5.6 Logistic Regression

Logistic Regression is a statistical model that uses a logistic function to model a binary dependent variable:

Probabilistic Approach: Logistic Regression estimates the probability of an event occurrence, which is a good feature for decision-making processes. For instance, the application could set thresholds based on these probabilities to categorize predictions into different risk levels, allowing for a more informed response to potential heavy drinking episodes.

Efficiency: Logistic Regression is less computationally intensive than other classification methods like Neural Networks and SVMs, making it a good choice for our real-time application.

Easy to Implement and Interpret: Logistic Regression is straightforward to understand and explain and it can be regularized to avoid overfitting.

No Assumption of Linearity: The method also does not require a linear relationship between the dependent and independent variables. This allows modeling complex relationships.

5.6.1 Data Preparation

The data perparation was the same as for MLP, Random Forest, AdaBoost, GNB and SVM.

5.6.2 Model Architecture

Logistic Regression utilizes the sigmoid activation function to map input features to a probability score between 0 and 1. The decision boundary is set at 0.5, and the model is trained by minimizing the logistic loss function.

5.6.3 Model Training and Results

The 10-fold cross validation method yields the following results. The average accuracy of the logistic regression model is 73.02%.

Fold	Accuracy
1	58.84%
2	84.95%
3	81.98%
4	91.00%
5	94.62%
6	49.36%
7	26.57%
8	63.51%
9	71.88%
10	94.06%

Table 9: 10-Fold Cross-Validation Accuracies for Logistic Regression Model

5.7 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are well-suited for analyzing sequential data for various reasons:

Ability to Capture Local Dependencies: CNNs have filters that convolve over the input data. These filters are able to capture local dependencies in the data (for example relationships between consecutive accelerometer readings or TAC values). This is particularly useful in time-series data where temporal dependencies exist.

Parameter Sharing: The same filters (weights and bias) are used across the entire data, which allow the CNN to recognize patterns at different positions in the time series.

Automatic Feature Learning: CNNs can automatically learn and extract features from raw data, proving to be more effective than manually engineered features, given the inherent challenge for humans to capture intricate patterns within complex datasets.

Hierarchical Feature Learning: Lower layers of a CNN might learn to detect simple changes in movement patterns, while deeper layers could learn to recognize more complex patterns over longer periods of time.

Robustness to Noise and Variations: CNNs are known to be robust to noise in the input data and to variations in the input. This is important as real-world accelerometer and TAC data can be noisy and can vary greatly between different individuals and devices.

Due to their ability to automatically and hierarchically learn features from raw sequential data, their robustness to noise and variations, and their ability to capture local dependencies, CNNs are a good choice for analyzing your sequential data.

5.7.1 Data Preparation for the CNN

Reshaping for 2D Convolutional Model To train a 2D Convolutional Model, the training and testing data are reshaped. For each participant (`PID`), accelerometer data was transformed to have dimensions 80×3, which means that the CNN gets windows of 80 records of accelerometer values triples. Target variables remain unchanged.

5.7.2 Model Architecture

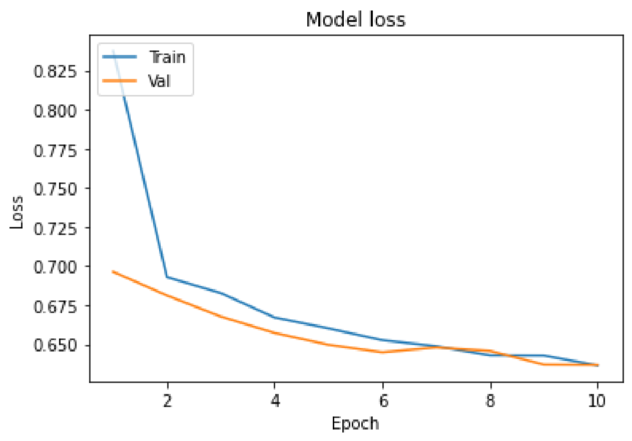
The following architecture is my initial attempt to learn the dataset. It comprises various layers designed to capture spatial and temporal features from the accelerometer data. I break down the model:

1. **Conv2D Layers:**
   - **Conv2D:** The first layer convolves over the input data with 16 filters, each of size 2×2.
   - **Conv2D:** The second Conv2D layer further processes the data with 32 filters of size 1x1.
2. **Flatten Layer:**
   - **Flatten:** This layer flattens the 3D output from the previous Conv2D layers into a 1D array.
3. **Dense Layers:**
   - **Dense:** The first dense layer consists of 128 units, contributing to the hierarchical feature learning process.
   - **Dense:** The second and final dense layer also comprises 128 units, providing the model's output.
4. **Dropout Layers:**
   - **Dropout:** A dropout layer is introduced after the first Conv2D layer to prevent overfitting. It has no impact on the shape of the data.
   - **Dropout:** Another dropout layer follows the dense layer with 128 units, aiding in regularization.

The total number of parameters in the model is 338,288, with all of them being trainable.

5.7.3 Model Training and Results

The CNN model exhibits learning behavior as seen in the training and validation loss plot (left). As the epochs progress, both losses decrease, indicating effective learning. The accuracy plot (right) shows that the model achieves an accuracy of 63.42% when trained on data from a single participant (`PID`).



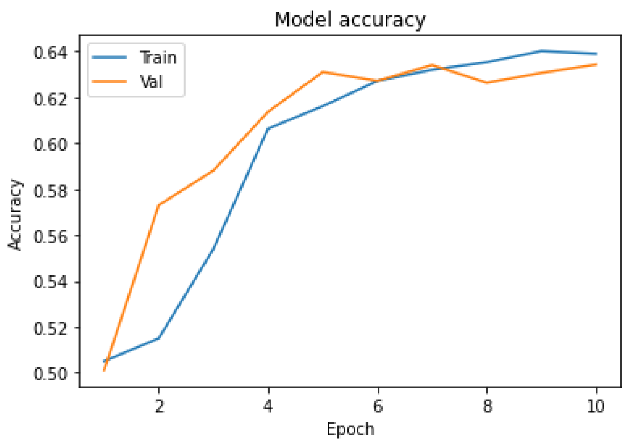


Figure 3: CNN Model Training Progress for One Participant

5.7.4 Further Model Training

To increase the accuracy of the model I combined the datasets of the participants. I trained the same CNN on 12 participants' datasets and tested it on the 13th. The training results can be seen below: From the following graphs and the table 10, we can see that the CNN trained on the combined dataset outperforms the first model. To evaluate the model on unseen data, the last dataset (`PID SF3079`) was used, yielding an accuracy of 80.34% on the test set.

Participant ID	Training Accuracy	Validation Accuracy
BK7610	0.8690	0.8535
BU4707	0.9103	0.8561
CC6740	0.8444	0.8100
DC6359	0.8706	0.8359
DK3500	0.8940	0.8288
HV0618	0.8706	0.8005
JB3156	0.7951	0.7535
JR8022	0.8221	0.7581
MC7070	0.7550	0.7410
MJ8002	0.7905	0.7586
PC6771	0.7571	0.7380
SA0297	0.7773	0.7529

Table 10: Training Results for Different Participants

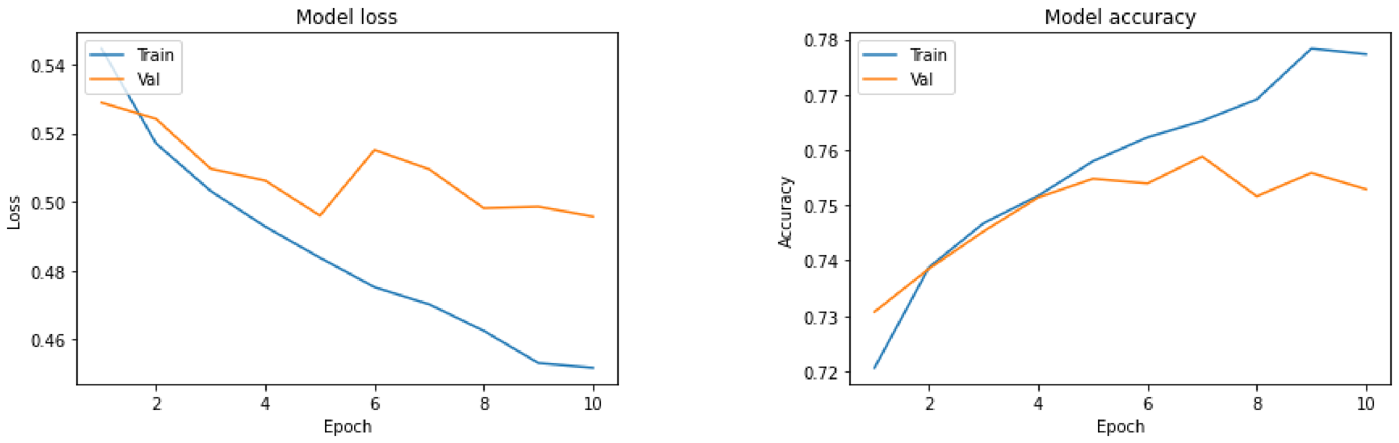


Figure 4: Training Progress on 12/13 Participants - Loss (left) and Accuracy (right)

5.8 Long Short-Term Memory Networks (LSTMs)

Long Short-Term Memory Networks (LSTMs) present another compelling approach for the analysis of sequential data, offering distinct advantages tailored to the sequential nature of the dataset:

Capturing Long-Term Dependencies: Unlike traditional neural networks, LSTMs are designed to capture long-term dependencies in sequential data. This is particularly advantageous in scenarios where intricate patterns unfold over extended time periods.

Sequential Memory Mechanism: LSTMs are equipped with a sequential memory mechanism, allowing them to selectively store and retrieve information over time. This feature is essential when dealing with dynamic data patterns, ensuring the model can learn and remember relevant context.

Handling Variable-Length Sequences: The accelerometer and TAC data exhibit variable-length sequences due to differing durations of the participant being sober or intoxicated. LSTMs excel in handling such variability, providing adaptability to sequences of different lengths.

Natural Handling of Time-Series Data: LSTMs are inherently well-suited for time-series data, making them effective in capturing temporal dependencies between consecutive data points. This aligns with the time-sensitive nature of our accelerometer and TAC measurements.

Reducing Vanishing Gradient Problem: Also, I chose LSTM over RNN as its architecture mitigates the vanishing gradient problem encountered in deep networks, enabling more effective training on long sequences. This is crucial for a good performance over extended periods of data.

The LSTM's ability to capture long-term dependencies, handle variable-length sequences, and naturally process time-series data makes it a promising choice for our sequential data analysis.

5.8.1 Data Preparation for the LSTM

As explained in Section 4, the Z-score Standardization technique is applied to the dataset. This standardization process ensures that different features are comparable and prevents any single feature from dominating others due to differences in their scales.

5.8.2 Model Architecture

The following architecture represents the LSTM model, refined after extensive fine-tuning for optimal performance. The layers are designed to effectively capture temporal dependencies in the accelerometer data. The breakdown of the model is as follows:

1. **LSTM Layers:**
   - **LSTM:** The first LSTM layer has 256 units and processes input sequences of length 80.
   - **LSTM:** The second LSTM layer follows with 32 units, extracting deeper temporal features from the sequence.
2. **Dense Layer:**
   - **Dense:** The dense layer with a single unit produces the model's output, indicating the prediction for heavy drinking episodes.

The total number of parameters in the LSTM model is 303,265, and all of them are trainable.
TRY 2 DENSE LAYERS 32 -> 32 OR 16 -> 1

5.8.3 Model Training and Results

The LSTM model's training progress is depicted in the loss and accuracy plots below. The model was trained over four epochs, and the training and validation loss decrease over epochs, indicating that the model is learning. The model achieves a test accuracy of 80.14%.

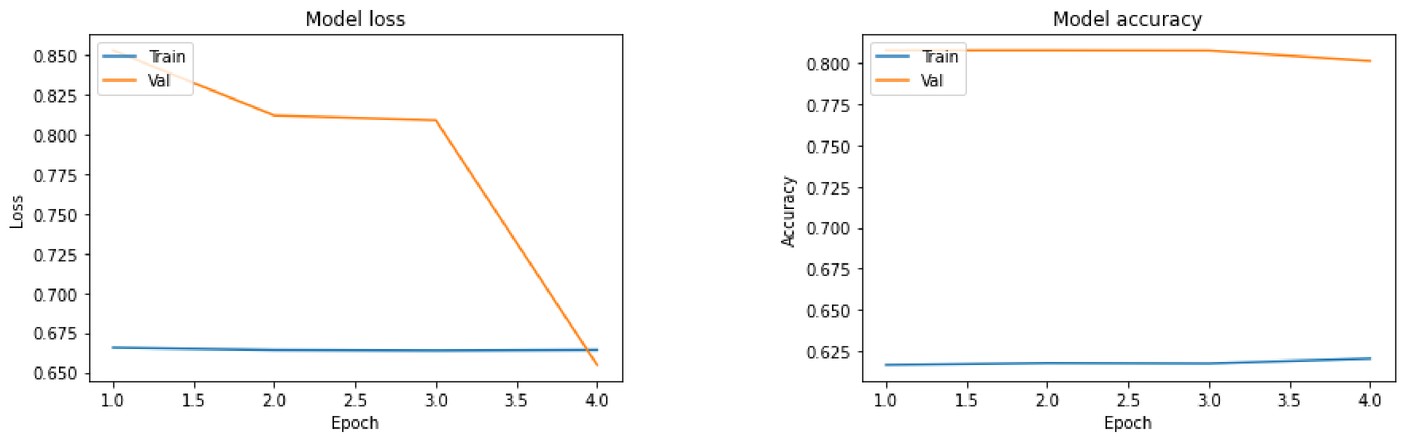


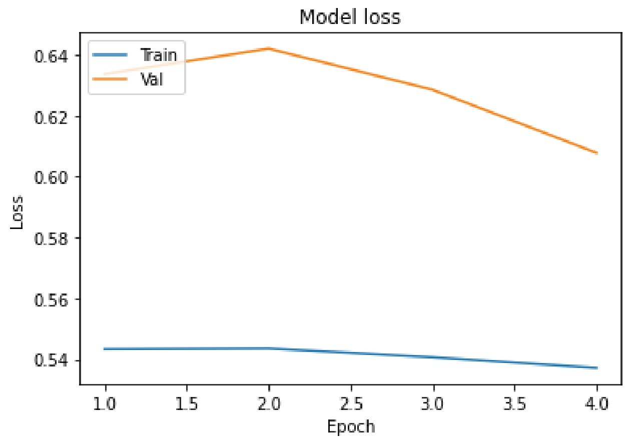
Figure 5: LSTM Model Training Progress for One Participant

5.8.4 Further Model Training

To enhance the accuracy of the model, the datasets of 12 participants were combined, and the LSTM was trained on the aggregated data. From Table 11, it is observed that the LSTM model trained on the combined dataset shows varied performance across participants. The worst performance was 67.65% validation accuracy and the best performance was 97.73% validation accuracy. The following figures represent the training progress for the LSTM model on the aggregated dataset. To evaluate the LSTM model's generalization on unseen data, the last dataset (`PID SF3079`) was used, resulting in an accuracy of **73.01%** on the test set.

Participant ID	Training Accuracy	Validation Accuracy
BK7610	0.6577	0.7426
BU4707	0.6780	0.6765
CC6740	0.6254	0.9773
DC6359	0.6630	0.9616
DK3500	0.6990	0.9355
HV0618	0.7124	0.9742
JB3156	0.6844	0.8352
JR8022	0.7022	0.8661
MC7070	0.7124	0.8518
MJ8002	0.7148	0.8499
PC6771	0.7220	0.8133
SA0297	0.7330	0.7290

Table 11: Training Results for Different Participants using LSTM



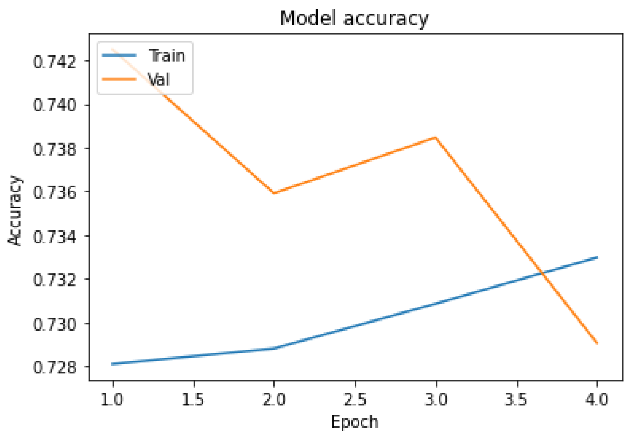


Figure 6: Training Progress on 12/13 Participants using LSTM - Loss (left) and Accuracy (right)

6 Results

The performance summary of various models is presented in the table below: These results demonstrate the Random Forest model's

Model	Average Accuracy
MLP	71.71%
Random Forest	**99.51%**
AdaBoost	67.82%
Gaussian Naive Bayes (GNB)	29.98%
SVM	71.56%
SVM with RBF Kernel	[pending]
Logistic Regression	73.02%
CNN	80.34%
LSTM	80.14%

Table 12: Summary of Model Performance

exceptional accuracy of 99.51%, indicating its strong ability to capture and generalize patterns in the data. Other simpler models, such as GNB, exhibit lower accuracy. Finally, more complex models that capture time dependencies, like CNN and LSTM were expected to be superior as the data are of sequential nature. However, their performance was surpassed by the Random Forest model, as it turned out that the data are not very complex and the time dependencies are not the strongest characteristic to decide the prediction.

7 Conclusion

8 Contributions

References

[1] World Health Organization. Alcohol, 2022. Accessed: February 8, 2024.

[2] Jackson A Killian et al. Learning to detect heavy drinking episodes using smartphone accelerometer data. 2020.

[3] María Gil-Martín et al. Alcohol detection over long periods using smartphone accelerometer data. 2020.

[4] Matteo Gadaleta et al. Idnet: Smartphone-based gait recognition with convolutional neural networks. 2017.

[5] Ming Zeng et al. Convolutional neural networks for human activity recognition using mobile sensors. 2014.

[6] Bar crawl: Detecting heavy drinking. UCI Machine Learning Repository, 2020. DOI: https://doi.org/10.24432/C5TK6G.