Learning to Detect Heavy Drinking Episodes Using Smartphone Accelerometer Data

Ananya Navelkar, Jainam Soni
M.Sc., Computer Science, Stony Brook University

***Abstract*—Implementation of [1] for CSE 512 Project. We have tried to extract nearly 200 out of 1214 features prescribed in the paper. Here, we show our results and understanding of the paper and also present some of our own methodologies to extract features and build a classification model.**

***Index Terms*—TAC, Accelerometer, Random Forest, MLP, SVM, Frequency-Time Domain feature extraction**

I. Introduction

Paper [1] aims at reducing the number of heavy drinking habits among college students by proposing a mobile intervention technique.

Many of the research works done in the past suggest that just-in-time adaptive interventions are more effective in reducing the number of heavy drinking cases than sending messages frequently. For example in one of the papers, when messages which were sent to a person while approaching a bar, it helped alert the person to avoid landing up in a risky situation.
Hence, using smart phone accelerometer data along with TAC sensor data can help us build a model to detect heavy drinking episodes.

A. *Important terminologies used in the paper:*

The paper suggests two methods of detecting a drinking event:

1) Blood Alcohol Content (BAC) - Measure of persentage of alcohol in the bloodstream when a person is drinking.
2) Transdermal Alcohol Content (TAC) - Measures the level of ethanol present in the perspiration produced by the body during drinking. (TAC readings)

B. *Limitations of existing methods to measure BAC or TAC:*

1) Users have to enter details manually in order to get BAC reading - not feasible due to selection bias and requires active user input.
2) Smartwatches measure TAC - expensive.

In order to overcome the above limitations the paper has proposed a solution in which the user's level of intoxication is measured with the help of accelerometer signals and interventions are made accordingly.

Advantages of this method -
1. No user input is needed, only normal behaviour of the user is required.
2. Excessive interference avoided, privacy of data is maintained.

II. Literature Survey

As mentioned in [1], the approach of collecting accelerometer data from a field test and merging it with TAC data is a novel approach. They use sensors to establish the ground truth which ensures that the model would be generalizable and reliable.
From [2], we understood why SCRAM devices are used to detect Transdermal alochol content in the body without collecting body fluids. [2] posits a reliable correlation between SCRAM Transdermal Alcohol Concentration and conventional breath test results over varying periods of time and under varying degrees of alcohol intake.

III. Methodologies Implemented

A. *Support Vector Machine*

[3] says that in a case of a nonlinear classifier a kernel function based on the dataset is used for determining hyperplanes. We tried with keeping $kernel = Linear$ but it took a lot of time and didn't converge. Hence for our task, paper suggests to use radial basis functions which is a popular learning algorithm used in various kernelized learning algorithms. The hyperparameters that we will use with rbf svm are gamma and C.
The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. [1] suggests to keep the gamma parameter as 2. But since we are not using libsvm, we will keep gamma as "scale" which is the default value. C behaves as the regularization parameter in the SVM. We keep C in between 0.5 and 0.7.

B. *Random Forest Trees*

As Random forest learns is a boosting technique, it turns out to be an ideal choice for our classification task. We will be trying RF with different values of n_estimators and max_depth to find suitable hyperparameters. Since we have generated many features, we would determine the importance of each feature by training Scikit-Learn's Random Forest Classifier on randomized subsets of the data, then use feature_importances function to evaluate the importance of each feature calculated as the normalized reduction of the Gini impurity brought by that feature.

C. *Multilayer Perceptron Classifier*

MLP is expected to work well if there are good number of training samples. Since we reduce the number of rows in 2nd approach, we can notice difference in performance for the both approaches. [1] prescribed to build a shallow MLP model but we didn't get good results with it. We have built a MLP network with 4 dense layers with increasing hidden neurons size starting from 64. We add a dropout layer with 0.5 to avoid overfitting between two hidden layers. Adam optimizer with learning rate as 0.001 works best for our case. Our loss function would be binary_crossentropy because our target variable has only two values 0 and 1.

IV. Dataset Description

The dataset consists of data from 13 college students.
1. Training Data - Samples of triaxial accelerometer readings (x,y,z) were taken at a sampling rate of 40Hz using a smartphone accelerometer.
2. Expected Values - A SCRAM ankle bracelet was used to measure the TAC readings which were sampled every 30 minutes for a duration of 18 hours.

Accelerometer data for every person is in one file which has around 13 million rows and TAC reading of corresponding person is kept in a separate file. We will be using a windowing approach to calculate features in time-domain. As mentioned in [1], we will use combination of 1 sec and 10 sec windows to extract features over that time. Also, we will follow a Two tier approach where in window size for first tier would be a combination of 1sec and 10 sec. And for second tier, we will keep a window size of 10 sec and drop all 1 sec window features.
While making windows, we have to make sure that we are not making windows with 2 different pid's in it. And hence, we choose to separate the accelerometer data for every pid and calculated their features separately. Additionally, we merge every pid's features with their TAC reading from the clean tac directory. Though,the timestamps in both the files are in seconds, they do not match exactly. So we have wriiten a code which sorts these timestamps and store the TAC reading timestamps in a dictionary. While iterating on the accelerometer data, we check if it's timestamp value is in between the two TAC readings, and if yes, we assign the target value of the first reading. Furthermore, we converted this problem into binary classification where every TAC reading above 0.08 is termed as 1. Our dataset now has 1:3 balance ratio for target variable where one third of the dataset readings have target as 1 which indicates that the person was drunk enough. We plotted the drinking period of each of the 13 volunteers, and found that their TAC level was above 0.8 during the middle of the event. There was no such thing that they were only drunk at the end of the event. Their drinking periods were well separated with some of them having 2-3 periods of high TAC reading. Also, we observed one person with TAC reading above 0.8 during most of the day, which maybe due to the noise in the data. Also, one person never had TAC reading above 0.8 and hence none of his rows had target variable $= 1$.

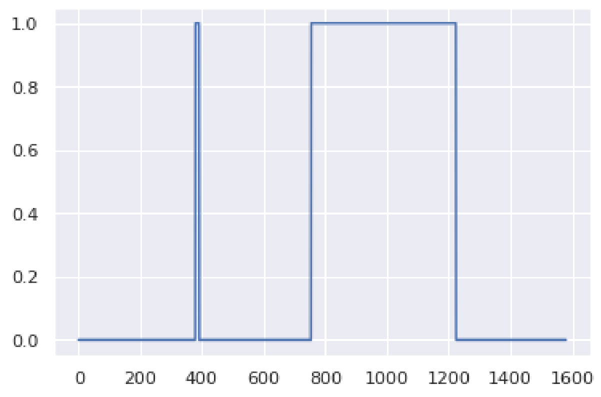

Fig. 1. Target TAC Reading fluctuation during the event for pidBU4707

V. Implementation

A. *Feature Extraction*

*1) **Segmentation**:* Segmentation was done in 2 ways:
1. Every participants data of 18 hours was divided into window sizes of 1 second and 10 seconds.
2. Every participants data of 18 hours was divided into window sizes of 10 seconds.

*2) **Calculating features**:* To calculate 1 second window, we use step_size as a function which takes all the rows in one window until it notices a change in 4th last position of timestamp. The last 3 digits are milliseconds and hence when we see a change in 4th last position we assume that one second is over.
For accelerometer data of every person, we find mean(3), standard deviation(3), median(3), zero-crossing rate(3), max(3) and min(3) of raw signal over all the 3 axes. This gives us around 18 features. We calculate this for every 1 second window throughout the data. Simultaneously, we keep a check for a 10 second window. We calculate 6 summary statistics - mean, variance, max, min, mean of lower third and upper third of the sorted values for every feature extracted in the last 10 seconds. For example, variance of 1-second features like mean(3), standard deviation(3), median(3), zero-crossing rate(3), max(3) and min(3) of raw signal over all the 3 axes. So, variance gives us 18 features. Additionally for 10 second windows, we calculate difference with the previous window for all the 6 summary statistics. So we doubled the variance feature to 36 features. We will do this for all the 6 summary statistics and by this we will reach close to 210-220 features.

B. *Method Formulation*

We tried all the 3 classifiers for our two tiered approach. For 10 seconds approach, we have around 36969 rows and 147 columns. For a combination of 1 second and 10 second approach, we have around 369597 rows and 96 columns. While building data for 10 sec, we drop all the 1-second features like x, y, z axes accelerometer data along with their means, std dev, median, etc. We found that these values null values in 10-15 rows. Since this is a small number as compared to the length of dataset, we chose to drop these rows straightaway without any imputation. After dropping the 1-sec specific columns, we calculated difference between present and previous values for our remaining 72 columns. Hence, we doubled our features by this approach as prescribed in the paper. Along with this 144 columns, we have encoded pid's using LabelEncoder, timestamp value in seconds and target variable. We choose the timestamp value to be the first value in the 10 seconds window.

C. *Parameter Tuning*

We will change two parameters and observe the change for Random Forest model:- 1) threshold TAC value and 2) number of imp features to keep (λ)

1) ***Tuning Thresholds of TAC reading:*** Classifications of sober vs intoxicated were done with different TAC threshold values - 0.04, 0.08, 0.1. Results were as shown in TableII. It was expected that the classification would improve as the differences would become more prominent between sober and intoxicated classes. But we observed that the performance decreased gradually and increased after a certain TAC threshold value indicating that student behaviours are distinct. Similar observations were made in the paper [1] as well.

TABLE I
PARAMETER TUNING OF TAC-CUTOFF TO MAKE CLASSIFICATIONS OF SOBER (TAC<THRESHOLD)VS. INTOXICATED (TAC≥THRESHOLD)

Sr.No.	TAC-cutoff	Accuracy	Precision	Recall
1	0.04	96.79%	99.95%	99.97%
2	0.08	91.15%	87.87%	91.18 %
3	0.1	94.03%	92.71%	94.03 %

2) ***Tuning Lambda for number of important features to be selected:*** : We used "feature_importances" to calculate important features for our random forest and sorted them. Results are represented in Table 2. We did this for our 10 second window approach where we had 147 columns.

TABLE II
PARAMETER TUNING OF LAMBDA FOR RANDOM FOREST

Sr.No.	λ	Accuracy
1	0.2	95.91%
2	0.4	94.18%
3	0.8	92.3%

VI. RESULTS

Since, we have 1:3 balance of target variable in our dataset, accuracy won't be a good measure to judge the performance of the model. Hence, we calculate the precision and recall to ensure that our model is correctly classifying true positives and true negatives. To avoid overfitting, we split the dataset into 70-30 using train-test-split library.

A. ***Using 1 second and 10 seconds window:***

Our Random Forest model turns out to be the best model in this case. Since, it has around 96 features to learn with around 3 lakh rows. From Fig. 2, we can observe that AUC is nearly 1 which says that we have correctly classified true positives and true negatives. Apart from high accuracy, RF also gives good precision and recall values.

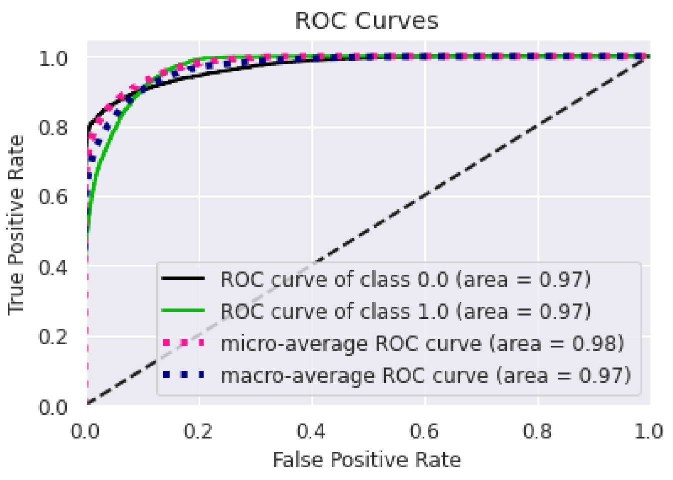

Fig. 2. ROC Curves for Random Forest Classifier with 10 sec window

TABLE III
ACCURACY, PRECISION AND RECALL OF THE CLASSIFIERS FOR TIER -1

Sr.No.	Classifier	Accuracy	Precision	Recall
1	MLP	74.25%	–	–
2	SVM	76.71%	38.35%	76.71%
3	RF	99.89%	99.89%	99.89%

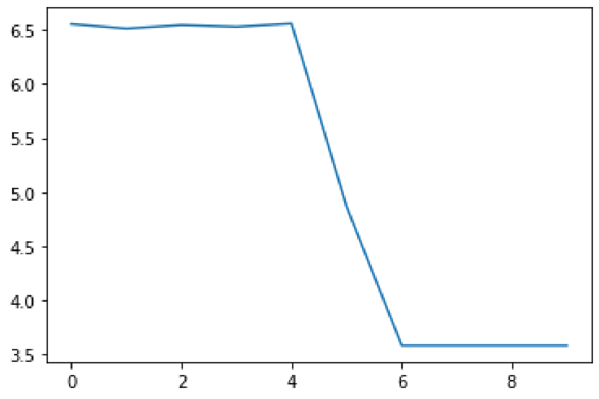

Fig. 3. Loss Function for MLP

The above loss function graph shows that our model has finished learning and converged at epoch 8. Just as [1], even our MLP model gives the lowest accuracy which says that neural nets is not a good choice for this classification task. The loss function does not decrease gradually and saturated after a while. The features are not rich enough to learn and hence adding more dense layers won't do any good.

B. Using only 10 seconds window:

We tried adjusting hyperparameters and performed parameter tuning as described above on each of these classifiers. For SVM with Radial Basis Function as the kernel, we found 0 true negatives and 0.5 area under the curve. Thus, we can conclude that SVM is not able to capture true negatives and should be avoided.
As neural network learns better if there are more number of training samples but in this approach we had only around 36,000 rows. And therefore we can see a decrease in accuracy from MLP tier-1 to MLP tier-2

TABLE IV
ACCURACY, PRECISION AND RECALL OF THE CLASSIFIERS FOR TIER -2

Sr.No.	Classifier	Accuracy	Precision	Recall
1	MLP	70.94%	–	–
2	SVM	76.97%	38.48%	76.97%
3	RF	91.15%	87.87%	91.18%

We can see a decrease in accuracy for Random Forest when we removed our 18 one-second window features. When we calculated feature importances and sorted them, 10-second window features had greater importance than their corresponding 1-second features. Additionally, the original x, y, and z tri-axial accelerometer data had 0 importance which tells us that calculating features from our dataset was important to learn relevant information.

VII. LIMITATIONS

Some of the extracted features like number of steps had to be dropped because they had very similar values and also their difference column was filled with 0s. Also, we started with spectral entropy feature but we were not able to complete it. We calculated many raw features but were not able to calculate frequency domain features. Though we got pretty good accuracy but our model may be error-prone in detecting real-life drinking scenarios. Just-in-time adaptive interventions should be accurate because we are dealing with human lives. Overall, because of time restrictions, we were not able to completely understand the time series signal and extract it's corresponding features in frequency domain. Also, we could not apply CNN model because we did not calculate MFCC covariance features which were important summary statistics.

VIII. CONCLUSION

We developed a model to classify data to determine mobile interventions in case of heavy drinking events. In the process we learnt how to handle accelerometer time series data and calculate various features by performing segmentation over the time domain. We performed parameter tuning and observed that Random forest was the best classifier among the 3. SVM failed to classify true negatives, whereas MLP's loss did not decrease gradually. We also learned the importance of precision and recall when you have class imbalance in your target variable. More work is needed to better understand the outcome of every feature.

REFERENCES

[1] J. A. Killian, K. M. Passino, A. Nandi, D. R. Madden, and J. Clapp, "Learning to detect heavy drinking episodes using smartphone accelerometer data."

[2] J. S. Hawthorne and M. H. Wojcik, "Transdermal alcohol measurement: A review of the literature," *Canadian Society of Forensic Science Journal*, vol. 39, no. 2, pp. 65–71, 2006.

[3] A. D. Ojha, S. Yerremreddy, A. Navelkar, J. Soni, and P. Bide, "A two stage approach for detection of invasive and cervical intra-epithelial neoplasia using machine learning and image processing methodologies," in *2019 1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing Communication Engineering (ICATIECE)*, 2019, pp. 144–150.