Implementation of Paper: Learning to Detect Heavy Drinking Episodes Using Smartphone Accelerometer Data

Nikhil Doifode
SBU ID#: 112714121
Stony Brook University
Stony Brook, NY 11790, USA
ndoifode@cs.stonybrook.edu

Yasha Singh
SBU ID#: 112970310
Stony Brook University
Stony Brook, NY 11790, USA
ysingh@cs.stonybrook.edu

***Abstract*—**We implemented the work demonstrated in paper 'Learning to Detect Heavy Drinking Episodes Using Smartphone Accelerometer Data' by Killian et al[1]. The paper's work is aimed at promoting healthier drinking habits by use of just-in-time adaptive interventions (JITAIs) delivered on mobile platforms just before the onset of heavy drinking episodes. Our project aims to implement the Random Forest Classifier as demonstrated in the paper which can identify if the user is sober or intoxicated based on the data collected from an accelerometer. Our model makes classifications based on 10 second windows of accelerometer data gathered from the user's mobile phone to support the delivery of interventions in real-time. The classifications are done based on transdermal alcohol content (TAC) gathered from 13 subjects participating in an alcohol consumption field study. The TAC readings served as the ground-truth when training the system to make classifications. We identify the user is sober if TAC < 0.08 (g/dl) or intoxicated if TAC ≥ 0.08 (g/dl). We used data for one user and our classifier detected heavy drinking events with 81.4% accuracy.

Keywords— Detecting Heavy Drinking Episodes, TAC, time-series, frequency-domain, feature generation, classifier;

I. Introduction

The paper aims to develop a smartphone-based system to passively track a user's level of intoxication via accelerometer signals to support the delivery of mobile just-in-time adaptive interventions during heavy drinking events. This system's passive nature requires no user action beyond their normal behavior to generate measurements and it uses only raw accelerometer readings rather than highly sensitive user data such as keystrokes, calls, or location. Minimizing the use of sensitive data is of paramount importance for the system's adoption as digital privacy concerns grow.

The study collected smartphone accelerometer data for 13 students participating in a one-day "bar crawl" event where students, as a group, visited all the bars in a certain region on campus. Further, each student wore an ankle bracelet that measured TAC. To the best of our knowledge, this was the first study to both 1) collect data in a field setting and 2) use sensors to measure intoxication rather than self-reports. Thus the findings were 1) applicable to real-world drinking scenarios and 2) classifications are free from user bias. We used data for one user to implement our model.

The paper implements several machine learning classifiers including a convolutional neural network, shallow neural network, random forest, and support vector machine to make classifications. The random forest performs the best from all. We implemented Random Forest Classifier to show our output which achieves the accuracy of 81.4%.

II. Implementation

We analyzed the UCI Bar Crawl: Detecting Heavy Drinking Data Set [2] intended for this paper. From this data repository, we had the time series accelerometer data available to us and we generated the time-domain data and the frequency domain data for the same by ourselves. Given that the data is cleaned and free of any noise, our one major task was to generate the each of features/metrics from the accelerometer signals that will be required for training of the model. In the problem statement of this paper, the prediction task is as follows: given a sample of accelerometer data, classify the sample as corresponding to TAC above or below some preset threshold.

A. Feature Generation

As mentioned in the paper, for each feature we calculated a two-tiered windowed approach has been used to characterize the data as it changed with time. That is for a given metric of 10-second segments, we further segmented the window down to 1-second segments. We then calculated the metric for each small segment, and computed the mean, variance, max, and min of the metric over the 10 smaller segments to characterize how it changed over time along with the mean of the lower third and upper third of sorted values, creating a total of 6 summary statistics per metric.

We were able to generate a total of 17 measures which are described in Table 1. In the original work by Killian et al., they had used a total of 22 measures which in turn had generated 1215 features. In the table, the number in brackets denotes the number of features generated by that measure. In our case, we were able to generate 504 features out of 1215. As mentioned earlier, each metric was calculated over 1 second window and becomes the basis for 6 summarizing statistics for each of the 3 axes over the full 10 second window. Each of the features described in the first 14 rows were calculated per window i.e. 18 features per metric, then used again to find the difference between the current and previous window resulting in double the features i.e. 18 original and 18 new calculated out of difference between the current and previous window, so total of 36 was obtained.

We calculate these features using the two-tiered window approach with PyAudioAnalysis and SciPy. Our time series data of accelerometer reading for x, y and z axis reading which was already in time domain and we had to convert this to frequency domain to generate spectral features. For this we used fft function from Scipy library which applied Discrete Fourier Transform to time domain signal and returned complex values where the real portion is the amplitude of the component cosine waves and the imaginary portion is the amplitudes of the component sine waves. The following diagram explains this process [3].

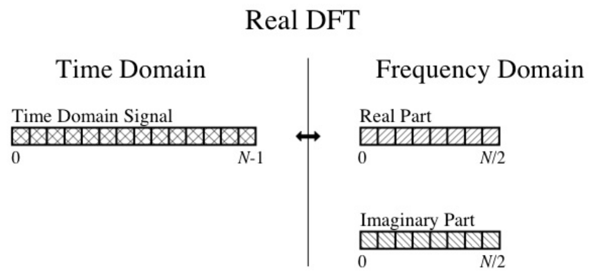
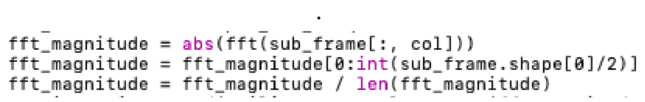


Fig 1.

And since we only need the information for the frequency part we can ignore the sign and remove the repeated values after removing the sign. We also had to normalize the output of fft. The following code snippet shows the process mentioned above.



```
fft_magnitude = abs(fft(sub_frame[:, col]))
fft_magnitude = fft_magnitude[0:int(sub_frame.shape[0]/2)]
fft_magnitude = fft_magnitude / len(fft_magnitude)
```

Fig2.

Here sub_frame[:, col] takes the time domain input of the axis defined by col and converts to frequency domain. Now features like Spectral Entropy, Spectral Centroid and Spectral Spread, Spectral Roll-off and Max Frequency. We used the PyAudioAnalysis [4] library for this purpose from the above generated frequency domain we generated We couldn't use the function from the library as it is since it requires each window has same number of samples but since we cleaned some noise and removed outliers the number of sample points in each window differs. Hence we extracted the features from the library and customized to take variable size input. Other features were easy to generate by using the direct library function either from python or Scipy [6]. Eg. Skewness, Kurtosis, Avg Power over Welch's spectrogram, Mean, Median, Max/Min. etc.
We built and trained random forests using Python's Scikit-Learn using a forest with 700 trees over summarizing features generated on all metrics in a 10 second window of accelerometer data.

B. Classification

We generated our feature set corresponding to one 'pid' namely, 'BK7610' and it had a dimension of 3073 and 504 columns. We had initially generated one dataframe per metric and finally joined these 17 metrics dataframe taking key as 'timestamp' for each dataframe. In order to get a working demonstration we worked on correctly performing the tasks for this 'pid' only. Generating data for all 13 pids, was a challenge in terms of storage. We separated the data in the ratio of 75:25 for the purpose of train and test as originally done in the paper.

Next, we generated the labels for the data entries in our feature set. For this we observed the clean_tac for pid - BK7610 and found out which timestamp had Tac > greater than 0.08 threshold and gave it a label '1'. Now, we mapped our timestamps of features set to the range of timestamps in 'clean_tac' file and found the corresponding labels. Since, we had 504 features in contrast to original 1215 features, so we did not perform dimensionality reduction for the feature set. Finally, as seen from the results in the original paper, out of all classifiers, Random forest classifier was giving the best results, so we chose it. We got an accuracy of 81.4% and a total of 118 False Positive and 25 and false negative out of 769 test labels.

III. Result and Discussion

Using a window length of 10 seconds we had 3073 rows of data each with 504 features. Intoxicated samples (TAC > 0.08) made up about two-third of the data. We randomized the data and split 25% as the test set.

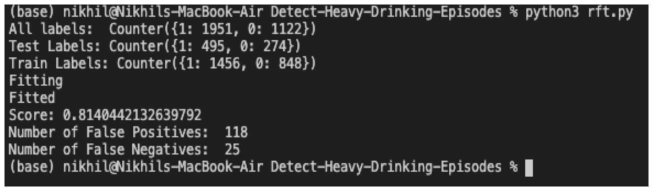
We achieved classification accuracy on an accelerometer / intoxication task, reaching 81.4%. We were able to achieve this result for two main reasons. First, we only used data from one user and we accessed a participant cohort that was sufficiently large and that generated data for a time-span long enough to allow us to collect more than 25,000 observations.
For sober samples that we wrongly called intoxicated (false positives), 50% of samples had TAC between 0.066 to

0.077, which is reasonable. For intoxicated samples that we wrongly classified as sober (false negatives), 50% of samples had TAC between 0.08 to 0.09 which is again understandable.

SN	Feature(s)	Description
1	Mean [36]	Average of raw signal
2	Standard Deviation [36]	Standard deviation of raw signal
3	Median [36]	Median of raw signal
4	Zero Crossing Rate [36]	Number of times signal changed signs
5	Max of Raw [36]	Max of raw signal
6	Min of Raw [36]	Min of raw signal
7	Max of Abs [36]	Max of absolute signal
8	Min of Raw [36]	Min of absolute signal
9	Spectral Entropy [36]	Entropy of energy in the time domain
10	Spectral Entropy [36]	Entropy of energy in the frequency domain
11	Spectral Centroid [36]	Weighted mean of frequencies
12	Spectral Spread [36]	Measure of variance about the centroid
13	Spectral Roll-off [36]	Frequency under which 90% of energy is contained
14	Max Frequency [18]	Max frequency from FFT
15	Skewness [18]	Measure of asymmetry of time-series signal
16	Kurtosis [18]	Heaviness of tail/Fourth moment of time-series signal
17	Average Power [18]	Average power over Welch's power spectrum distribution

Table 1. List of Measure with number of features associated with them

The problem arises where 25% of false negatives sample had TAC of 0.135 - 0.16 where patients would be considered extremely intoxicated, but our classifier missed them and same with 25% of samples where TAC was 0.042 to 0.048 and our classifier mislabelled them as intoxicated. The following screenshot shows our result.



```
(base) nikhil@Nikhils-MacBook-Air Detect-Heavy-Drinking-Episodes % python3 rft.py
All labels:  Counter({1: 1951, 0: 1122})
Test Labels: Counter({1: 495, 0: 274})
Train Labels: Counter({1: 1456, 0: 848})
Fitting
Fitted
Score: 0.8140442132639792
Number of False Positives:  118
Number of False Negatives:  25
(base) nikhil@Nikhils-MacBook-Air Detect-Heavy-Drinking-Episodes %
```

The evaluation of false positives and false negatives is done from the following boxplots shown below.

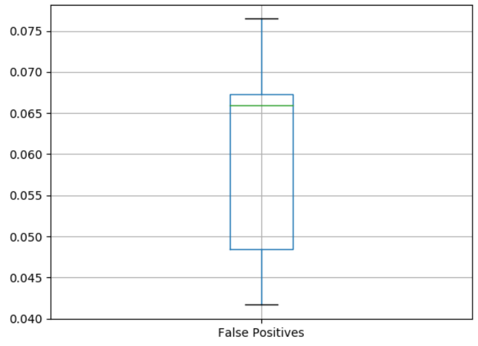


Fig 3. (a)Boxplot showing False Positives

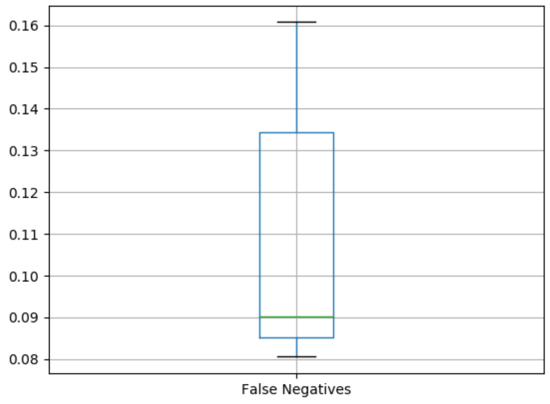


Fig 3. (b) Boxplot showing False Negatives

Figure 3(a) and 3(b) shows the boxplot showing the actual TAC for samples that were misclassified by the random forest, (a) shows the distribution for false positive cases and the (b) for false negative cases. TAC cutoff is at 0.08.

IV. Future Scope

Our plan was to train our classifier on the data collected from all the 13 users and also generated additional features for MFCC covariance, RMS and features related to steps taken like Gait Stretch, Number of Steps, Step time and Cadence. We would also like to train the model by collecting accelerometer data from a Smart watch where the user is not in motion but seated at one position.

V. Conclusion

Although our classifier was trained only on data from one user, it performed admirably well and generated better output. We think the reason might be because we didn't do any dimensionality reduction and data reduction. Also, it can be inferred from the model performance that the features we had selected for implementation were highly relevant features for our model, especially the Spectral features, given that our data was a time series one.

VI. References

[1] Killian, Jackson A. et al. "Learning to Detect Heavy Drinking Episodes Using Smartphone Accelerometer Data." KHD@IJCAI (2019).

[2] Bar Crawl: Detecting Heavy Drinking Data Set

[3] The Scientist and Engineer's Guide to Digital Signal Processing's Table of Content

[4] tyiannak/pyAudioAnalysis: Python Audio Analysis Library: Feature Extraction, Classification, Segmentation and Applications

[5] Smartphone Inference of Alcohol Consumption Levels from Gait

[6] SciPy — SciPy v1.4.1 Reference Guide