Learning to Detect Heavy Drinking Episodes Using Smartphone Accelerometer Data

Karthik Govindappa (NetID: kg667)

Abstract

Excessive alcohol consumption is a significant health problem that has led to a huge impact on the intellectual and social lives of students on campuses. It has been a cause of death worldwide and an especially severe risk on college campuses. An estimated 88,0005 people (approximately 62,000 men and 26,000 women) die from alcohol-related causes every year, making alcohol the third leading preventable cause of death in the United States. Around 80% of college students consume alcohol to some degree and at least half of the students engage in binge drinking. Binge drinking refers to consuming too much alcohol in too little time. These students are more likely to become involved in adverse activities resulting in negative consequences.

We can promote healthier drinking habits by delivering adaptive interventions on mobile platforms just in time during the heavy drinking episodes. Such interventions can be helpful in avoiding health issues, drunk driving, violence, and alcohol poisoning due to excessive alcohol consumption. However, it is difficult to predict the beginning of heavy drinking episodes and we aim to build a machine learning model that will classify whether a person is heavily intoxicated or not. We would be implementing this based on work done in paper 'Learning to Detect Heavy Drinking Episodes Using Smartphone Accelerometer Data' by Killian et al[1] and will be doing it on Streaming data of multiple mobile devices

Introduction

There are two ways to identify alcohol intake activity: first, by directly measuring the blood alcohol content (BAC) and second, is a proxy such as transdermal alcohol content (TAC). TAC is a standard measure to calculate the amount of alcohol consumed. There are some mobile applications which allow users to input height, weight, age, and the number of drinks consumed in a given period of time to give an estimate measure of BAC. This can be ineffective, inaccurate and biased because it is heavily reliable on the user input. To decrease the dependency on user input there have been studies which constantly sense and store user data even when not involved in drinking events. They make use of multitude of mobile data, such as keystroke speed, sent/received calls, location and more. Such data collection methods can be a threat to user's private data and hence, may not be a good way to solve the problem. Also, not many people would like their activity data to get recorded at all the time and such methods don't necessarily have a future scope. An additional issue with this

can be that certian users might not be able to provide information to the apps because of excessive alcohol consumption.

Recent work has used machine learning to classify the levels of intoxication and how smartphones data can be used to classify user drinking behavior, but current methods lack generalizability or make extensive use of private user information. The classification of intoxication level of participants was highly dependent on the self-reports which are susceptible to be biased, inaccurate, or have missing data. Therefore, it is important to make accurate predictions with less sensitive information and have more reliable way of collecting data. For allowing the gathering of more accurate data and avoid impractical ways, one possible way is through the recording of motion and orientation of wearable devices.

Wearable devices have gained a lot of popularity in the recent times because of the variety of features that they offer. In our project, an appropriate wearable device will be the one which is application-oriented, has built-in motion sensors, is programmable, is not dependable on other devices, and is versatile.

In the paper we are referring to for our project [1],they've addressed these shortcomings by developing a reliable mobile classifier that uses only non-sensitive accelerometer data to detect periods of heavy drinking. This is a reliable, bias free and convenient method to obtain user data as it doesn't require users to enter data manually. We have trained machine learning models like Random Forest, Decision Tree, and Gradient Boost that work on 10-second windows of accelerometer data to make classifications of sober transdermal alcohol content (TAC < 0.08) and intoxicated (TAC $>= 0.08$). Unlike previous works which use self-reports, we will be using TAC values as the ground truth while training the model to make classifications. This will help us in building a more accurate system. The classification results by the model will be used to send just in time messages to the users during heavy drinking episodes. For training the model we use the data set comprising of the smartphone accelerometer readings and transdermal alcohol content (TAC) for 13 subjects.

Method of Processing

Approach

To detect heavy drinking episodes using smartphone accelerometer data, the first step is data collection. After collecting the data, data is preprocessed using several data preprocessing procedures to eliminate errors in the results which may be caused by irrelevant observations. Finally, with the extracted data, several classifiers are used to train the data with the available datasets and the data is tested to compare the performance of different classifiers and best classifier is used to solve the problem.

Motion sensors are there in all mobility devices, for example cell phone or smart watch, which stick with the user all the time. Accelerometer data, once calibrated, is very precise in its measurements, leaving very little room for error, making it even more suitable. To predict, we will be training using classifiers – Random Forests, Decision Tree and Gradient Boost on this data, infer which does a better job. These models will be working on data collected in every 10 second sliding window to infer if someone had a heavy drinking episode or not.

Data Selection

The dataset that we use for this project is from the UCI Machine Learning Repository. They had recruited 20 undergraduate students comprising of 50% men and 50% women, each in their senior year, aged 21-23. Only the students who are active consumers of alcohol were chosen as participants. A sensor for measuring TAC was attached to the participants which they had to carry throughout the drinking event. Simple application for iphone and android were developed to sample triaxial accelerometer readings at 40Hz, which were periodically sent to an InfluxDB server. The TAC data was sampled every 30 minutes using a SCRAM ankle bracelet sensor. Over 30M accelerometer samples were collected in this dataset.

Available at -`https://archive.ics.uci.edu/ml/datasets/Bar+Crawl%3A+Detecting+Heavy+Drinking` To determine the importance of each feature, data is trained by Random Forest Classifier, then evaluated the importance of each feature calculated as the normalized reduction of the Gini impurity brought by that feature. Three parameters needed tuning regardless of the underlying machine, namely the TAC cutoff between classes, the fraction of important features to keep and the length of the window of accelerometer data.

Data Preprocessing

The raw data we extracted for this project is preprocessed to make it understandable. Data is imported using libraries like pandas, numpy and datetime. The dataset has various issues with formatting, construction, arrangement, duplication, extra spaces, and so on. In order to come to an accurate conclusion, data is cleaned by following several approaches like getting rid of extra spaces, selecting and treating all blank cells, converting numbers stored as text into numbers, removing duplicates, downcasing all the strings and deleting all the formatting. Our data set is large and we had some records with missing values. As the dataset is large, we deleted those records. All the time series data is converted to frequency data. Apart from that raw-data is collected for a span of 1 minute which was divided to a window span of 10 seconds. Each attribute is split with respect to X axis, Y axis and Z axis. Irrelevant observations are deleted and the data is made uniform.

The entire data has above 14M records and 12 attributes for each record. For each record, mean, zero crossing rate, max/min, standard deviation, spectral entropy, centroid, spread, median, max frequency, skewness, curtosis, rms are collected.

Each attribute is described as follows -

- Mean: Mean represents the average of raw signal.
- Max/min: It represents the Max/Min of raw/absolute signal. It has a total of four different metrics like Max for raw signal, Min for raw signal, Max for absolute signal, Min for absolute signal.
- Standard Deviation: It represents the standard deviation of raw signal.
- Spectral entropy: It represents the Entropy of energy. It has total of two metrics, spectral entropy in frequency domain and spectral entropy in time domain.
- Centroid: It represents the weighted mean of frequencies.
- Spread: It represents the measure of variance about the centroid.
- Median: It represents the median of raw signal.

- Max frequency: It represents the max frequency from FFT.
- Skewness: It represents the Measure of asymmetry of time-series signal.
- Kurtosis: It represents the Heaviness of tail/Fourth moment of time-series signal.
- RMS: It represents the Root-mean-square of accelerations for each axis.

We generated our feature set corresponding to one 'pid' namely, 'BK7610' . We had initially generated one dataframe per metric and finally joined these 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 70:30 for the purpose of train and test.Next, we generated the labels for the data entries in our feature set. We repeated the same process for all the pid's and merged all the data into a single processed csv file and employed it for model training.

We have processed 44 features and one output label which corresponds to the TAC reading. All the instances with a TAC value of greater than or equal to the threshold value of 0.08 are labelled as '1' and all the instances with TAC value less than 0.08 are labelled as '0'.

x_mean	x_variance	x_median	x_min	x_max	x_rms	x_energy_en	x_energy	x_skew	x_Kurtiosis	x_FFT_variar	x_FFT_spect	y_mean	y_variance	y_median	y_min	y_max	y_rms	y_energy_en	y_energy	y_skew	y_Kurtiosi
-1.0935069	1.0710961	-0.511803	-2.5568151	-0.0176519	1.5056073	-162823.83	2.26685335	-0.3366328	-1.7280535	104.538511	(0.30656656	4.02065866	14.7090342	1.8232354	0.20933159	9.458371	5.55650342	-434983.66	30.8747302	0.36039177	-1.70317
-0.0288783	0.0009802	-0.0322443	-0.1363995	0.06583004	0.04259286	-230.70894	0.00181415	-0.0613739	2.3730285	0.08605267	(0.30836639	0.10137458	0.0060129	0.11948805	-0.0663656	0.19808426	0.12763111	-485.95174	0.0162897	-0.5465876	-0.87990
-0.0117758	0.00013715	-0.0154592	-0.0398304	0.02239119	0.01660766	-16.346928	0.00027581	0.94128799	1.31508074	0.01350884	(2.91115754	0.02904421	0.00160954	0.02521132	-0.0621779	0.11987356	0.04952889	1.78678929	0.00245311	0.250679	0.037353
-0.0177324	0.00010357	-0.0191424	-0.0408783	0.01338884	0.02044532	-503.87007	0.00041801	0.31076577	0.50459598	0.02113931	(1.18181709	0.00401581	0.00087534	0.00624482	-0.0773714	0.08705098	0.02985747	-85.076414	0.00089147	-0.0571746	0.488276
-0.037622	0.00016708	-0.0369512	-0.0759521	-0.0079046	0.0397806	-98.004197	0.0015825	-0.3690106	1.66672922	0.07584687	(0.06355880	0.00077317	0.0006239	-0.0002588	-0.0501034	0.05335829	0.02498988	-1442.0032	0.00062449	-0.0423663	-0.71654
-0.0451915	0.00100823	-0.0437982	-0.1202987	0.05682158	0.05523133	-253.84341	0.0030505	0.5453215	2.24746368	0.15067287	(-0.0050407	0.00046952	0.00054325	-0.0009283	-0.0482998	0.08821773	0.02331239	-6242.5486	0.00054347	0.93771416	2.431745
-0.0484469	0.00011637	-0.0488297	-0.0787247	-0.0182474	0.04963334	-102.93679	0.00246347	-0.0533269	1.68367328	0.11721939	(0.11040426	-0.0011845	0.00036586	4.33E-05	-0.0466168	0.03878205	0.01916401	-778.19047	0.00036726	-0.0079894	-0.4105
-0.0453657	0.00013004	-0.045743	-0.0838487	-0.0157276	0.04677695	-179.93902	0.00218808	-0.4062181	2.02810804	0.10695312	(0.06183491	-0.0014721	0.00033693	-0.0039669	-0.0343238	0.04410482	0.01841448	-186875.18	0.00033909	0.30958854	-0.43402
-0.0277993	0.00019609	-0.0274037	-0.0566834	0.00079801	0.03112707	-1663.4478	0.00096889	-0.0345279	-0.762867	0.04623457	(0.22433920	0.00014467	0.00058619	-0.0004062	-0.0587976	0.05116125	0.02421177	-514.08817	0.00058621	-0.0933553	-0.47539
-0.0087859	0.00025488	-0.0120086	-0.0428165	0.04951425	0.01822277	-3767.2967	0.00033207	1.62019852	4.28556209	0.01623648	(0.29068271	0.00113093	0.00032137	-0.0005281	-0.046439	0.03466526	0.01796236	-92.068437	0.00032265	-0.2779107	-0.08336
-0.0046036	0.00013639	-0.0086499	-0.0242021	0.03092756	0.01255309	-48.743928	0.00015758	1.39408007	1.75892457	0.00761984	(0.29202105	0.00022484	0.00029953	0.00351371	-0.0323975	0.04820079	0.01730836	-1508.9829	0.00029958	0.55208336	0.328803
0.00039463	0.00014309	-0.0036086	-0.0169152	0.02977074	0.01196845	-3894.3692	0.00014324	1.03578354	0.01064694	0.00653814	(0.51210136	-0.0005269	0.00070517	-0.0023021	-0.0613695	0.05925994	0.02656024	-76.785684	0.00070545	0.11755212	-0.04045
0.00125875	0.0001471	-0.0050483	-0.016415	0.03132913	0.0121937	-24.967357	0.00014869	1.0032212	-0.159099	0.00725207	(0.62905551	0.00213566	0.00046731	0.00371324	-0.0472323	0.05669609	0.02172249	-73.10351	0.00047187	0.13484221	-0.39109
0.00086431	0.00020961	-0.0017425	-0.0305151	0.0342314	0.01450372	-160.71014	0.00021036	0.10726846	-0.176966	0.01044113	(0.56932214	0.00058416	0.00037255	0.00099662	-0.0592952	0.06707844	0.0193105	-18.472228	0.0003729	0.186026	2.731117

Figure 1: Processed File

Principle

We will use the following methods to determine the level of alcohol consumed by a student. Each classifier is used because of its pro's and a comparison is made between the below three classifiers on this problem.

Random Forest: Random Forest operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random Forest is selected over Decision trees as they are more accurate as they don't overfit in their training set. The training set is fed into the Random Forest and the data is tested. Accuracy, precision and Recall are calculated. For our project we initialized the forest with 10 decision trees and the set the maximum depth as 20.

Decision Tree : A decision tree (or a reduction tree) is a predictive model which is a mapping from observations about an item to conclusions about its target value. Nonlinear relationships between parameters do not affect tree performance. Decision trees implicitly perform variable screening or feature selection

Supposition

In this project we are trying to measure how accurately we can distinguish between a sober and an intoxicated person using the time series data from the accelerometer data from participant's mobile phones. This will help in answering the questions like

- can a machine learning model identify if a given user has been intoxicated more than the safe limit?
- Can we make a system where the input data is not dependent on the user to make it bias free?
- Can we make a system that does not need to take longer time to get results?

The accelerometer data must provide a more convenient, bias free and secure to use data that can be helpful in promoting healthier drinking habits by delivering adaptive interventions on mobile platforms just in time during the heavy drinking episodes.

Architecture

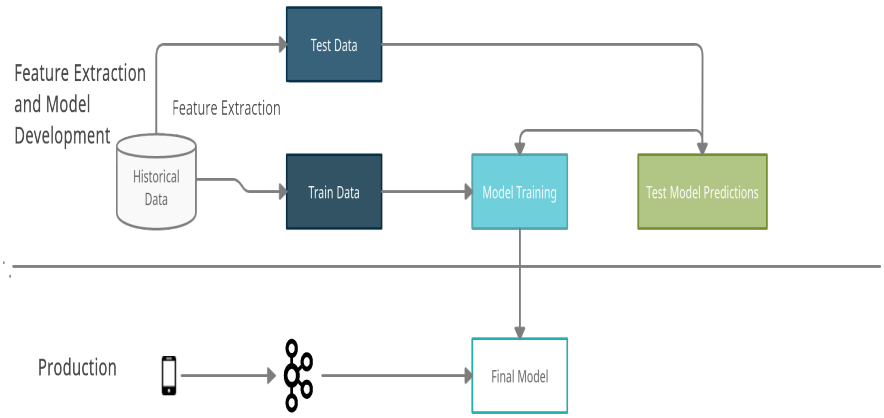

Figure 2: Architecture Overview

The Fig 2 gives us an overview of our project, we have divided our project into two separate processes 1) Feature Extraction and Model Development 2) Production deployment

Feature Extraction and Model Development

In this phase of the architecture we use the historical data to build a model that can be used to predict newer data. We use Spark and Spark ML as the platform to build our model. The data to be trained is at rest, hence we wouldn't be using Kafka in this process

We've explained about Feature Extraction in the Data Preprocessing section earlier

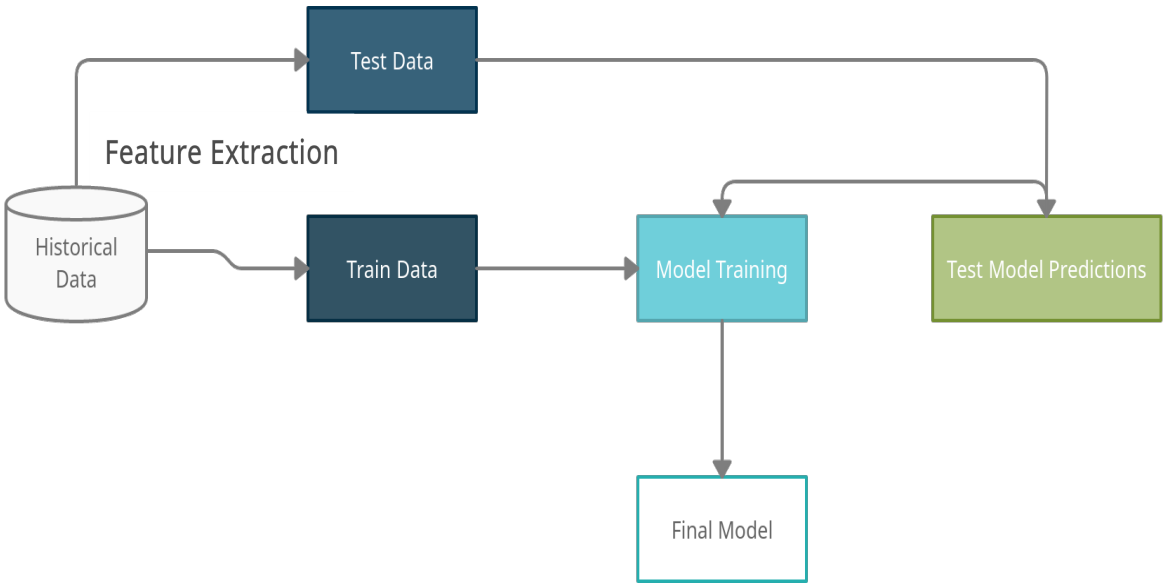

Figure 3: Model Development

Once the features of the data are extracted, we then proceed with the model development. First we divide the model into Training and Test datasets to evaluate the models once trained. We then load the Training data into Spark and use Spark ML libraries classification algorithms to build a model. We then used the Test dataset to evaluate the best model of the multiple we've trained and choose it has our final model to be deployed on the production environment. This phase of our architecture is the longest as we had to do multiple hyperparameters tuning to come up with the best model

Production Deployment

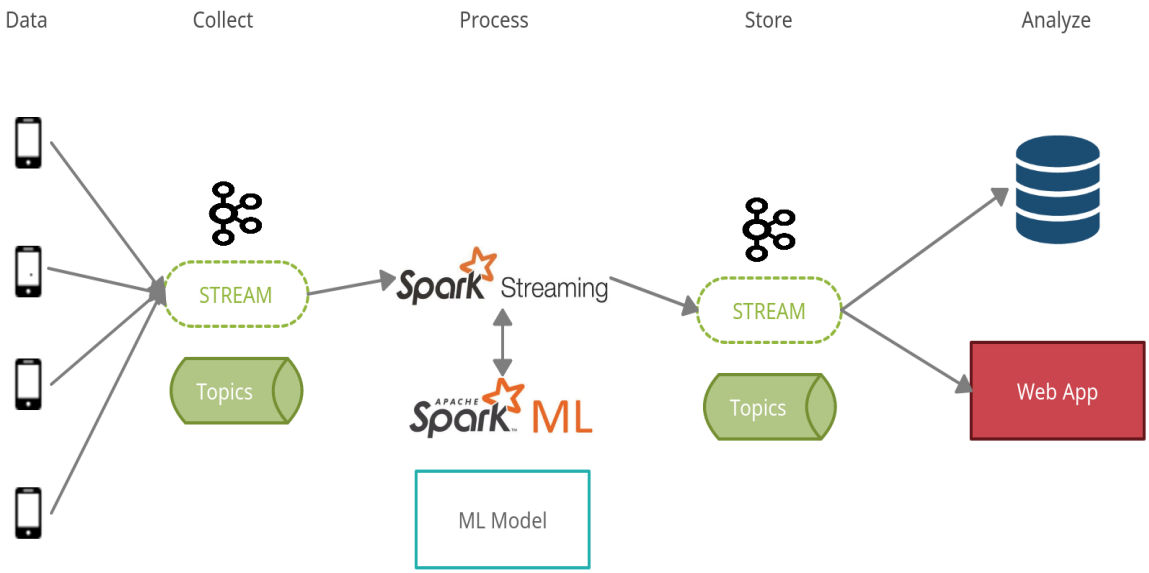

Figure 4: Production Deployment Overview

Once the model has been trained we would be deploying it to be used in a streaming setting as shown above. Here multiple mobile phones publish streams of accelerometer data to a Kafka topic which collects data until it can be processed further. Spark Structured Streaming reads the collected stream of the data as a Spark Dataframe which is then used by Spark ML to predict based on the previous model trained. The predicted stream of results is once again published into a Kafka topic. Finally our Web-App server reads the data from Kafka topic to display the result on to the dashboard.

As can be seen from the image above, we've used Kafka twice to indicate that we've used two separate topics.

- Topic 1: features - receives data from the mobile data
  - Producer: Mobile Phones
  - Consumer: Spark Structured Streaming
- Topic 2: preds - receives prediction from the Spark ML
  - Producer: Spark ML
  - Consumer: Web App

To demonstrate the above process for our project, we've simulated the data being sent by mobile phones using a python script which publishes already preprocessed data to the Kafka topic.

Our web app can be accessed at http://3.94.7.116/

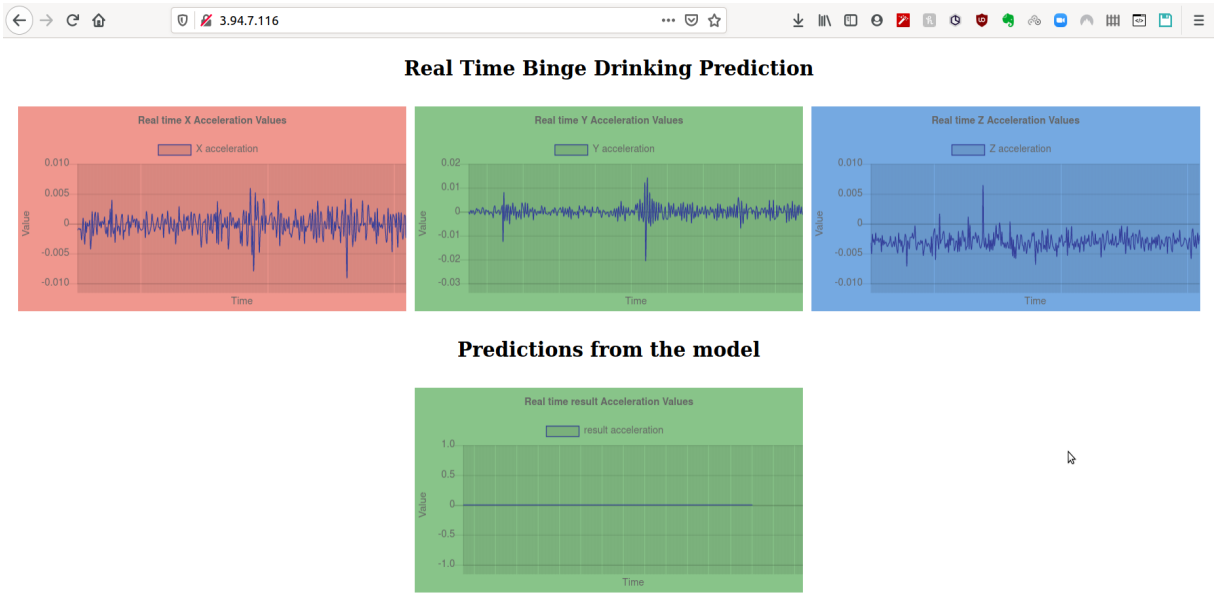

Figure 5: Screenshot of Webapp

Result

Following are the metrics obtained for the three ML models generated, Since the performance of Random Forest was high, we choose it as our final model for predictions.

```
precision = metrics.precision(1.0)
recall = metrics.recall(1.0)
f1Score = metrics.fMeasure(1.0)
print("Summary Stats")
print("Precision = %s" % precision)
print("Recall = %s" % recall)
print("F1 Score = %s" % f1Score)
```

```
Summary Stats
Precision = 0.9484389963608504
Recall = 0.9562413100571605
F1 Score = 0.9523241725484163
```

Figure 6: Random Forest

```
precision = metrics.precision(1.0)
recall = metrics.recall(1.0)
f1Score = metrics.fMeasure(1.0)
print("Summary Stats")
print("Precision = %s" % precision)
print("Recall = %s" % recall)
print("F1 Score = %s" % f1Score)
```

```
Summary Stats
Precision = 0.6171128518550401
Recall = 0.5524602165452934
F1 Score = 0.5829995730579218
```

Figure 7: Gradient Boost

```
precision = metrics.precision(1.0)
recall = metrics.recall(1.0)
f1Score = metrics.fMeasure(1.0)
print("Summary Stats")
print("Precision = %s" % precision)
print("Recall = %s" % recall)
print("F1 Score = %s" % f1Score)
```

```
Summary Stats
Precision = 0.5770110269830784
Recall = 0.4369996921182266
F1 Score = 0.49733920241771234
```

Figure 8: Decision Tree

Future Work and Improvements

We currently are running Kafka cluster with a single Amazon EC2 server which even though would work on small distributed clients wouldn't scale well when multiple clients connect. So we would want to have a better infrastructure in a future version

We want to implement the intervention alarm system, which sent a notification to the user once we are able to detect if the user has crossed the threshold for intoxication

References

[1] Jackson A Killian Kevin M Passino Arnab Nandi Danielle R Madden1 John Clapp1 (2019) Learning to Detect Heavy Drinking Episodes Using Smartphone Accelerometer Data KHD@IJCAI .

[2] [Adapa et al., 2018] Apurva Adapa, Fiona Fui-Hoon Nah, Richard H Hall, Keng Siau, and Samuel N Smith. Factors influencing the adoption of smart wearable devices. International Journal of Human–Computer Interaction, 34(5):399–409, 2018.

[3] [Abadi et al., 2015] Martın Abadi, Ashish Agarwal, Paul Barham, and et al. TensorFlow: Largescale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] [Arnold et al., 2015] Zachary Arnold, Danielle LaRose, and Emmanuel Agu. Smartphone inference of alcohol consumption levels from gait. In Healthcare Informatics (ICHI), 2015 International Conference on, pages 417–426. IEEE, 2015.

[5][Adapa et al., 2018] Apurva Adapa, Fiona Fui-Hoon Nah, Richard H Hall, Keng Siau, and Samuel N Smith. Factors influencing the adoption of smart wearable devices. International Journal of Human–Computer Interaction, 34(5):399–409, 2018.

[6] https://medium.com/@stressed83/live-streaming-data-using-kafka-node-js-websocket-and-chart-js-8750acad549c

[7] https://blog.clairvoyantsoft.com/machine-learning-with-spark-streaming-281b2d1e4fd5

[8] https://gist.github.com/MOAMIndustries/340442db992b2e6bfa77a85d752f2d52

[9] https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html

[10] https://towardsdatascience.com/streaming-scikit-learn-with-pyspark-c4806116a453