Capstone 2: Final Report

Problem: Can we detect someone's sobriety/drunkenness by their movement?

Client: Public health organizations, sociologists, universities, younger people. Potential to create app that warns users of heavy alcohol use.

Data: UCI Detecting Heavy Drinking Dataset
https://archive.ics.uci.edu/ml/datasets/Bar+Crawl%3A+Detecting+Heavy+Drinking

Data Description: Accelerometer data for 13 participants involved in a "bar crawl" event. CSV file includes position in 3 axes and time of measurement in milliseconds. Additionally, separate CSV files for each participant of TAC (transdermal alcohol content) readings, taken roughly every half hour. "Clean" readings are shifted back in time by 45 minutes to account for time it takes to release alcohol through the skin.

Features: variance in 3 axes (continuous numerical), participant ID (categorical string), phone type (categorical string)

Target: "drunk" label (binary categorical)

Data Wrangling: The following steps were taken:

1) The first two rows of the accelerometer data were dropped, as they contained all zero values and therefore appear to be a data entry error.

2) Time measurements for accelerometer data are taken in milliseconds since UNIX Epoch (01/01/1970). Values converted to correct datetime objects using the pandas .to_datetime function.

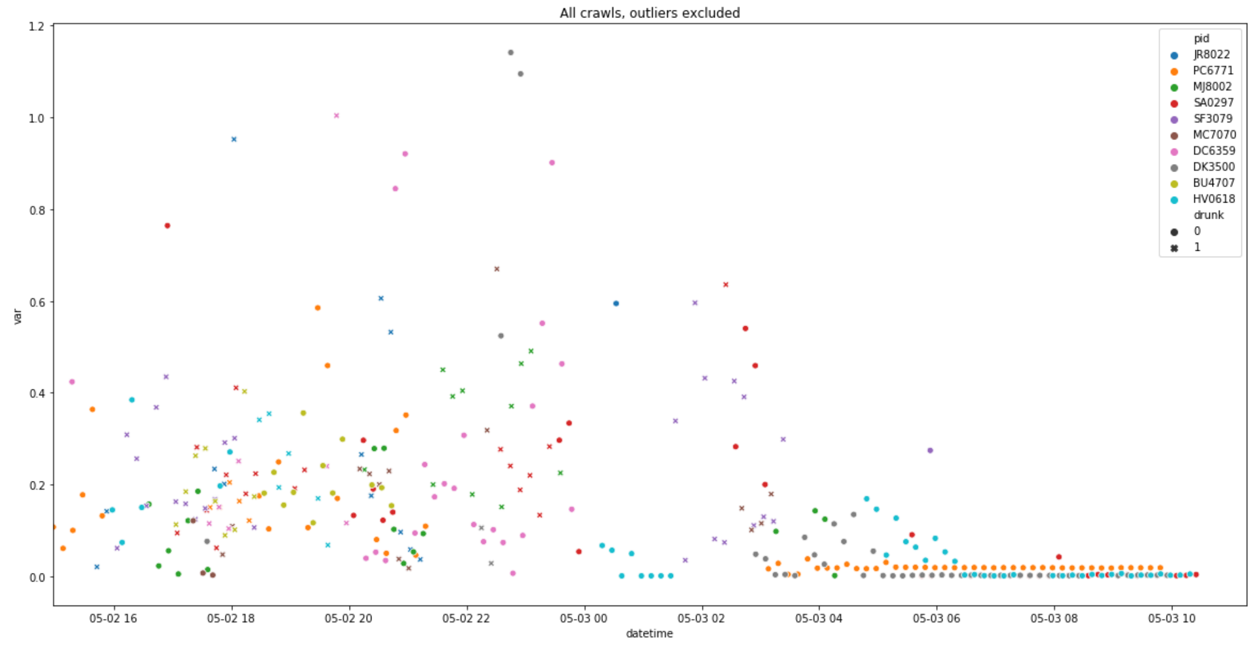
3) TAC readings were concatenated into a single dataframe by looping over the unique participant ID's. Here, time measurements are changed to datetime objects as well.

4) TAC readings for two participants are found to be identical, indicating a data entry error. One participant is dropped.

5) Variance is calculated from accelerometer data.
    a) Loop over readings for an individual participant.
    b) Take average of TAC readings within a thirty minute window. Because readings happen roughly every half hour, usually only one or two readings are aggregated.
    c) For every one TAC window, there are 3 sub-windows of 10 minutes each. Calculate standard deviation within sub-windows.
    d) If the TAC reading is greater than or equal to 0.08 (legal limit), label row as "drunk" (1 in "drunk" column).
    e) Loop over all participants.

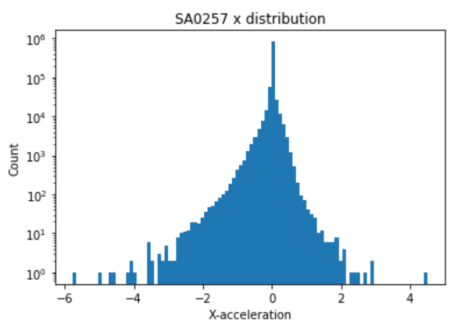
Exploratory Data Analysis

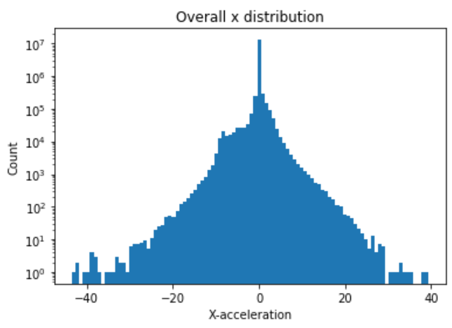
1) Two participants found to be outliers; discussed more below
2) Plotted variance over time:



At least visually, drunk variance looks higher. We see higher variance in general late at night, when participants would be drinking most heavily. Correspondingly, variance tapers off as the event ends and participants sober up.

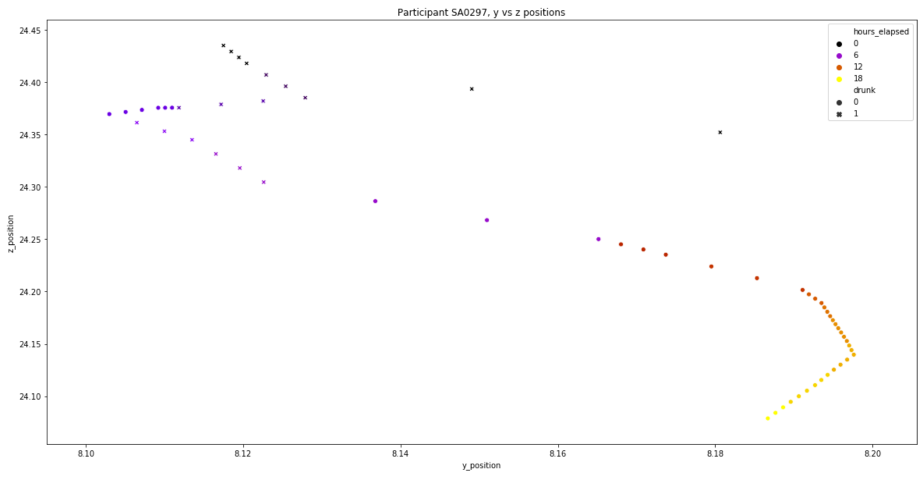
3) Graphed distributions of accelerometer data:

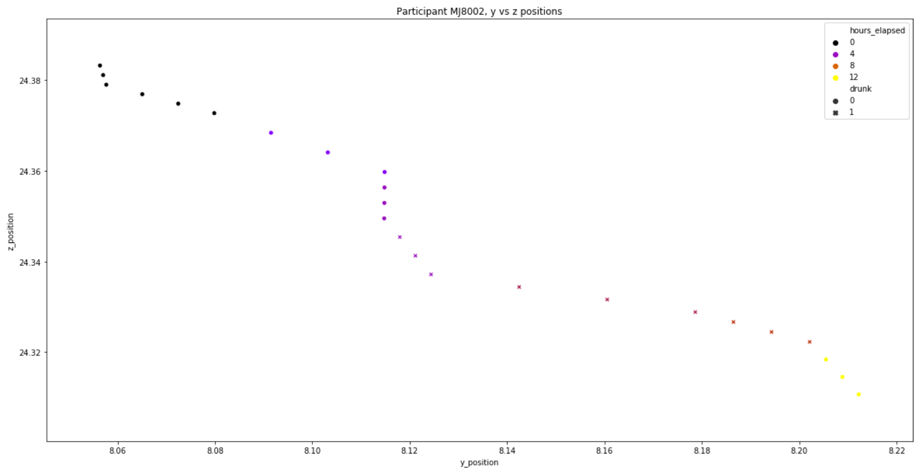




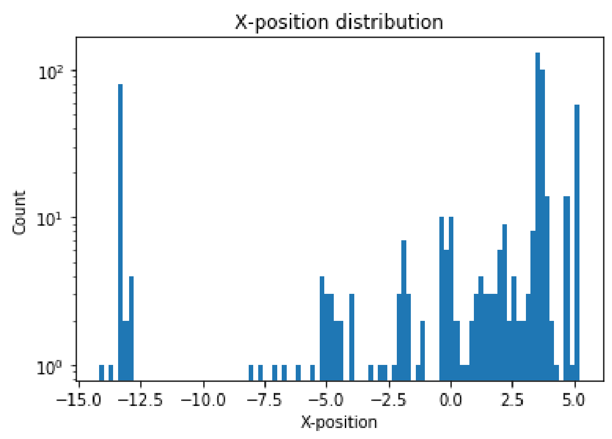
For all three axes, the data is normally distributed.

4) Calculated velocity and position from accelerometer data by integrating twice. This made it possible to graph participant movement in 2D space:





Positional data is randomly distributed, indicating there is no center of the bar crawl event:



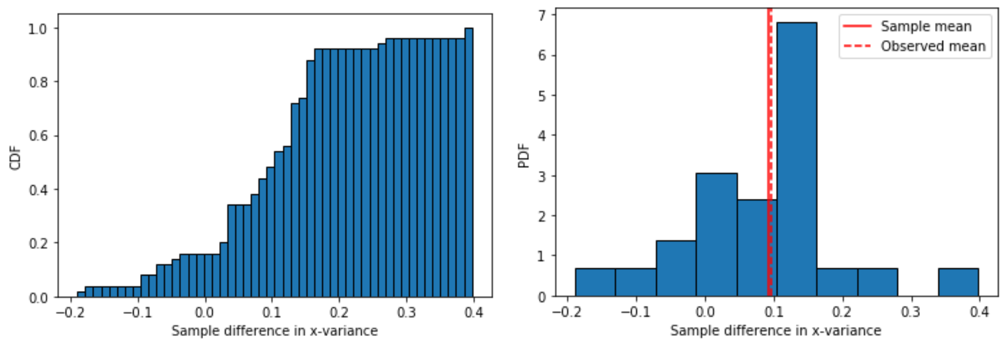
Inferential Statistics:

1) Observed accelerometer data compared to a bootstrap sample (n=50) to compute z-scores, which are in turn used to calculate p-values:

$$z = \frac{\bar{x}_1 - \bar{x}_2 - \Delta}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

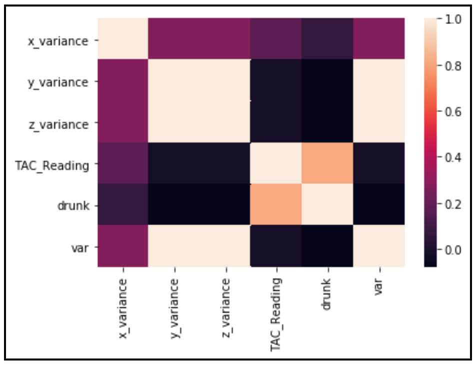
Standard deviations are computed directly from the observed population. P-values can be found in a z-table, or by using SciPy's norm.ppf function.

The null hypothesis is that there is no difference between drunk and sober movement, so the alternative hypothesis is that there is a statistically significant difference. The p-values were 1.7e-7, 0.001, and 3.3e-7 for x, y, and z respectively. Assuming the standard significance level of 0.05, the null hypothesis is rejected - there is a difference between drunk and sober movement. The CDF and PDF of variances were plotted:



Machine Learning Models:

1) Created heatmap to show collinearity:



Unsurprising that y- and z-variance are correlated, since most movement is done in 2D plane and therefore affects both axes.

2) The primary evaluation metric was decided to be F1 score, the harmonic mean of precision and recall:

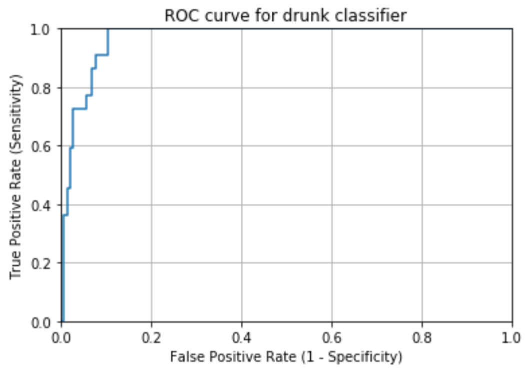
$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

The data has a class imbalance, as only about 30% of points correspond to drunk movement. The null accuracy comes out to about 87%, which means that a model that only predicts labels of 0 would be this accurate. Hence, accuracy is a bad metric.
In this project, one of the goals is to eliminate Type II errors, or false negatives. This is because it would be worse to tell a drunk user they're sober than vice versa - the sober person could just ignore the incorrect warning, while the drunk person would never get one. While an end user might find the erroneous message more irritating, the primary stakeholders of this project (medical or academic institutions) would be more concerned with correctly classifying drunk movement. Therefore, we have to measure the true positive rate, also known as recall.

3) Participant IDs were converted from categorical strings to binary numericals via one-hot encoding (Pandas' get_dummies function).
4) Two types of scaling were used: Standard and MinMax. Standard scaling subtracts the mean from each value and divides by the standard deviation. MinMax scales features to a given range, in this case 0 to 1. This scaling had some effect on model performance.
5) TAC readings were tested as a feature, which allowed some models to reach 100% accuracy. However, the high degree of collinearity between these readings and the target variable (as seen in the heatmap above) invalidates this model.

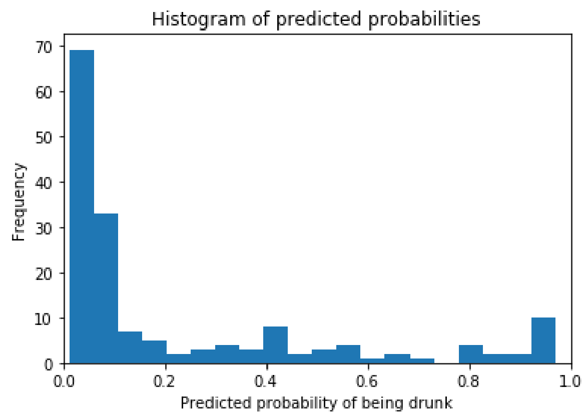
Additionally, using blood alcohol content as a feature is practically meaningless - in the real world, this is only rarely monitored.

6) 4 machine learning algorithms were used - Logistic Regression, Support Vector Classifier, Random Forest Classifier, and Gradient Boosting Classifier. The last two algorithms are ensemble methods, meaning they combine multiple different approaches; ensemble models tend to do better than single-approach models precisely because of the flexibility/adaptability of ensemble approaches. Random Forest uses bootstrap aggregating (sampling with replacement and then aggregating, also known as bagging) with different features for each decision tree within. Gradient boost also utilizes decision trees and bagging, but in a different way. Each bagged tree has its feature weights updated through gradient descent, which calculates the error, multiplies it by a learning rate, then subtracts it from the old weight to find the new weight. Random Forest aggregates its trees at the end of its run, but Gradient Boost uses gradient descent to incrementally improve weak learners. For this reason, Gradient Boost often outperforms Random Forest and most other classifiers, as we saw in this project.
7) One of the most challenging aspects of this project was splitting data into training and test sets. This data is a time series, so it requires stratified sampling; otherwise, collinearity with time means "future leakage" will contaminate model (that is, such a model "cheats" because later points give the model information about earlier patterns; this model would perform poorly on unseen data). Originally, my strategy was to naively split the data such that the first 70% was training data and the last 30% test data. Later I realized my mistake: this 70-30 split needed to happen within each participant's data, or else I was essentially randomly sampling. Fixing this error generally improved model performance.
8) I graphed the ROC (Receiver Operating Characteristic) curve for one of the Gradient Boosting models, which shows true positive rate vs. false positive rate:



The fact that area under the curve is so close to 1 indicates that the model is very high-performing.

9) Typically, the threshold for prediction is 0.5, meaning that all prediction values for a given observation below 0.5 are labeled 0, while all those above 0.5 are labeled 1. We look at the histogram of predicted probabilities:

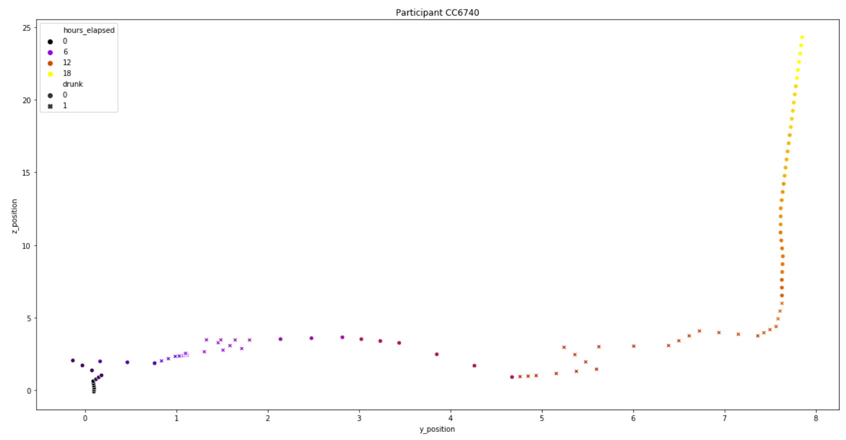


The heavy positive skew of this distribution close to 0 suggests that the threshold should actually be lowered in order to offset the class imbalance.

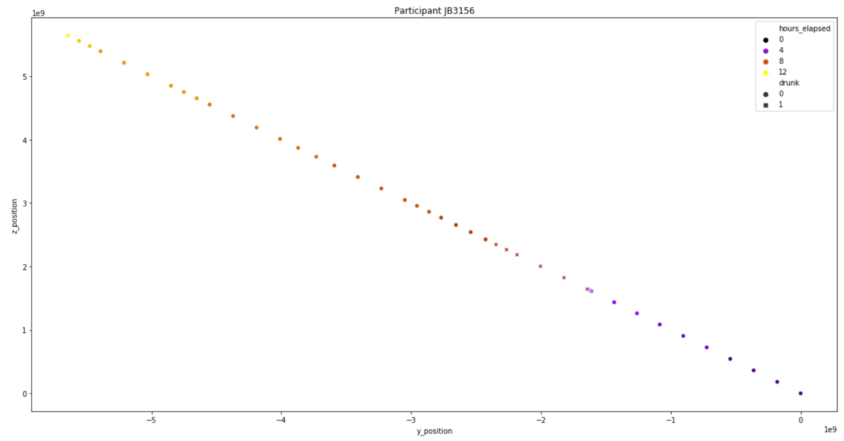
a) I used Sklearn's Binarizer function to manually change the threshold to 0.1, based on the above histogram. Classification reports show this actually hurt the model - the weighted F1-score average dropped from .92 to .79
b) I built a function which allowed me to evaluate model performance when changing the prediction threshold. These models are evaluated in terms of sensitivity and specificity, or true positive rate and (1 - false positive rate). Changing the threshold from 0.5 to 0.3 raised the sensitivity from .77 to 1.0, but the specificity lowered from .93 to .83. Because this improvement is somewhat dubious, I decided not to implement any threshold change.

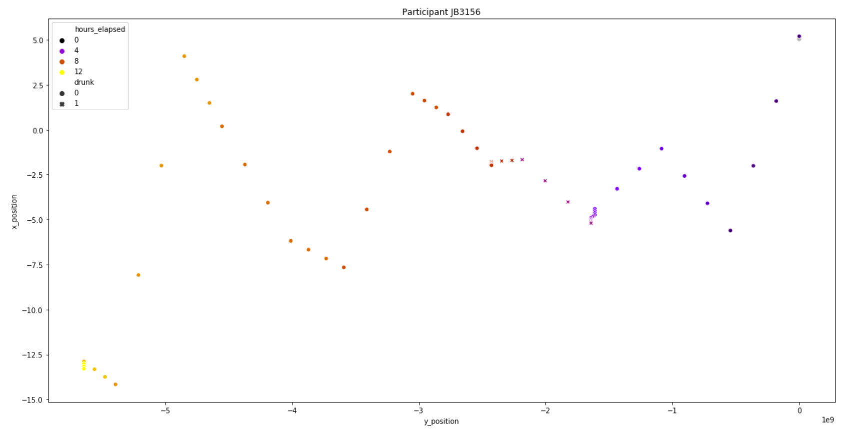
Re-examination:

1) I looked again at the average variance of sober movement and drunk movement and found a discrepancy - sober variance was actually about four times larger than drunk variance.
2) I decided to examine the data in terms of phone types. The dataset has 2 Android users and the rest are iPhone users.
    a) Mean variance for Android users was 1.4, compared to .08 for iPhone users. These distributions are not even on the same scale.
    b) A plot of positions clearly shows a data entry error. One Android user had relatively normal movement:



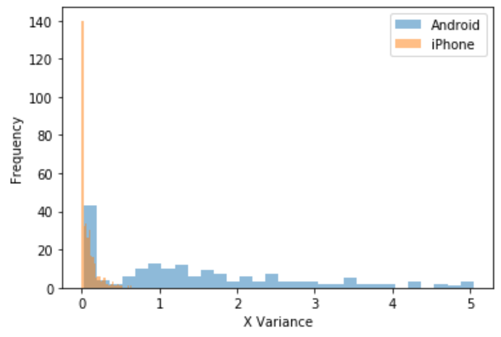
The other user clearly had erroneous data:



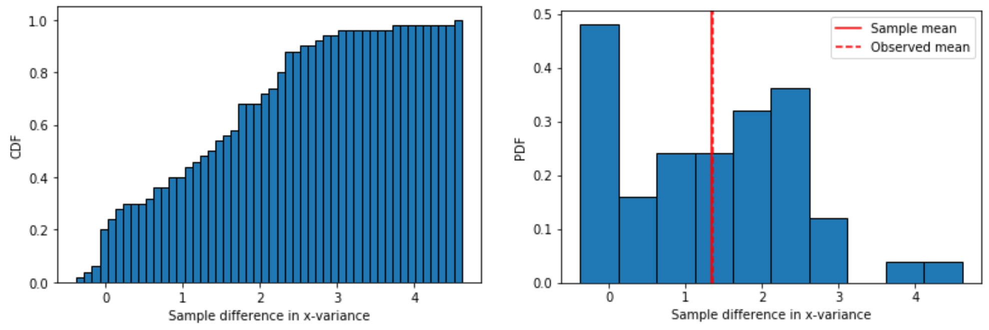


The first plot here shows the participant walking in an exact straight line for more than 12 hours while still managing to get drunk. Even more improbably, the second graph shows them walking exactly straight even as they go up and down steep hills.

3) Similar to before, I ran a z-test to check the statistical difference between android and iPhone users. The visual difference in histograms is stark:



Now the CDF and PDF for this z-test:

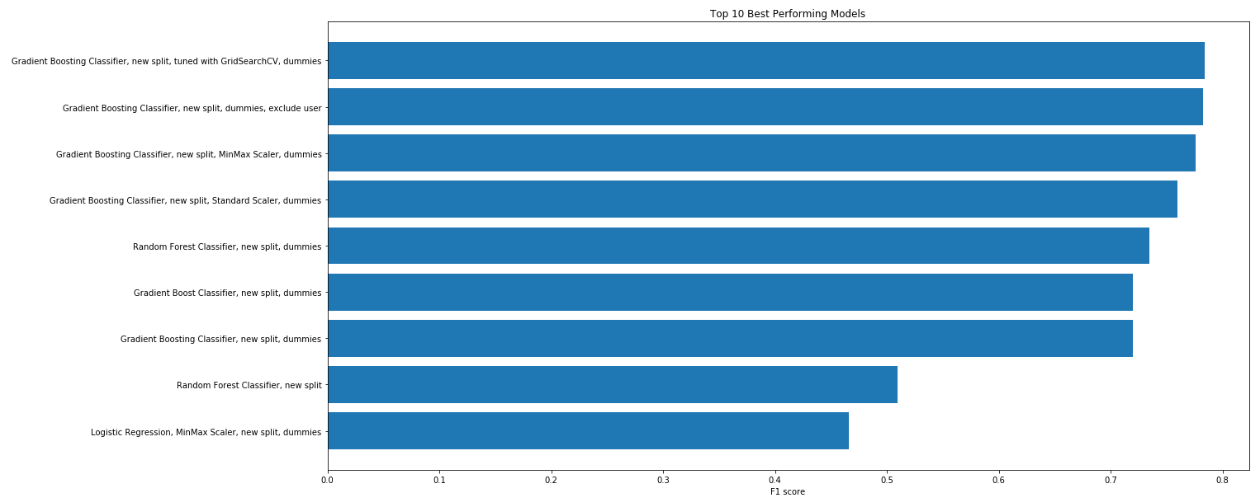


The z-test yielded a p-value of 3.3e-12, which allows us to reject the null hypothesis that there is no difference between these two groups of phone users.

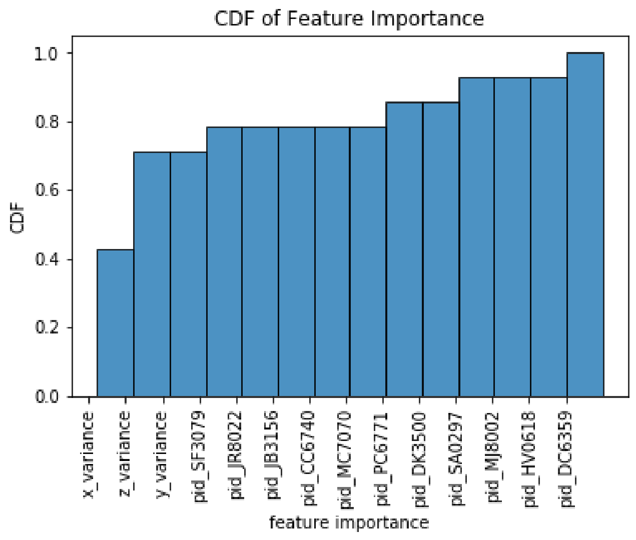
4) The extreme outlier user was excluded and mean variance was recalculated: .27 for sober movement, .84 for drunk. This matches our hypothesis that drunk variance should be higher.
5) Three new models were trained: one on the data with the outlier user excluded, another with phone type as a categorical feature, and a third with a ‘crazy_user’ label for the outlier. While excluding the outlier noticeably improved model performance, the other two models had negligible improvement.

Conclusion:

Our best-performing model was a Gradient Boosting classifier which had its hyperparameters tuned, with an F1-score of .784 and a test accuracy of 93.3%.
Here are the top 10 best models:



The feature importances were extracted from the model and plotted:



As expected, the variances were the most important features. However, the participant IDs accounted for roughly 25% of the model variance.

Further Work:

1) This model could be directly implemented into an application that would analyze phone accelerometer data and alert users when they might be drinking heavily. Again, this is why recall is so important - for stakeholders with a medical mindsight, the worst case scenario is failing to warn a drunk user.
2) The TAC readings used here were already processed such that they were easy to read/use. The original, raw readings were included in the project files. A potential expansion would be to process these raw readings ourselves, paying

particular attention to the participant who was dropped because they had duplicate data.

3) I decided not to change the threshold here, but future models could adjust it along with other hyperparameters to get higher accuracy / F1 scores.
4) I mentioned earlier that ensemble methods often outperform their more simple counterparts. We could take this further and combine our best models into one ensemble, stronger than any individual model.
5) The UCI machine learning repository has other projects with accelerometer data, such as this:

https://archive.ics.uci.edu/ml/datasets/WISDM+Smartphone+and+Smartwatch+Activity+and+Biometrics+Dataset+

The data here could be combined with the data from this project to expand the range of activities and physical states the model can predict.

Hyperparameter table:

model_name	**features**	**training_accuracy**	**test_accuracy**	**f1_score**	**hyperparameters**
Logistic Regression	variance	0.6594594595	0.5660377358	0.2580645161	
Logistic Regression with dummy variables	variance, pid dummies	0.772972973	0.5597484277	0.2553191489	
SVM classifier	variance	0.6837837838	0.5597484277	0.3	
SVM classifier with dummies	variance, pid dummies	0.8054054054	0.5597484277	0.2553191489	
Random Forest Classifier	variance	1	0.5094339623	0.3275862069	
Random Forest Classifier with dummies	variance, pid dummies	1	0.5157232704	0.3529411765	
Gradient Boost Classifier	variance	0.9621621622	0.5283018868	0.3801652893	
Gradient Boost Classifier with dummies	variance, pid dummies	0.9513513514	0.5220125786	0.3333333333	
Logistic Regression with Standard Scaler	variance	0.6621621622	0.5660377358	0.2580645161	
Logistic Regression with Standard	variance, pid	0.775675	0.566037	0.258064	

Scaler, dummies	dummies	6757	7358	5161	
Logistic Regression with MinMax Scaler	variance	0.6540540541	0.5911949686	0.2696629213	
Logistic Regression with MinMax Scaler, dummies	variance, pid dummies	0.7756756757	0.5660377358	0.2580645161	
Logistic Regression, new split	variance	0.6016483516	0.8666666667	0	
Logistic Regression, new split, dummies	variance, pid dummies	0.6016483516	0.8666666667	0	
SVM Classifier, new split	variance	0.6043956044	0.8666666667	0	
SVM Classifier, new split, dummies	variance, pid dummies	0.6043956044	0.8666666667	0	
Random Forest Classifier, new split	variance	1	0.8484848485	0.5098039216	
Random Forest Classifier, new split, dummies	variance, pid dummies	1	0.9212121212	0.7346938776	
Gradient Boost Classifier, new split	variance	0.9285714286	0.8363636364	0.4489795918	
Gradient Boost Classifier, new split, dummies	variance, pid dummies	0.9148351648	0.9151515152	0.72	
Logistic Regression, Standard Scaler, new split	variance	0.6016483516	0.8	0	
Logistic Regression, Standard Scaler, new split, dummies	variance, pid dummies	0.7527472527	0.7090909091	0.4146341463	
Logistic Regression, MinMax Scaler, new split	variance	0.6043956044	0.8121212121	0	
Logistic Regression, MinMax Scaler, new split, dummies	variance, pid dummies	0.739010989	0.7636363636	0.4657534247	
Gradient Boosting Classifier, new split, tuned with GridSearchCV, dummies	variance, pid dummies	1	0.9333333333	0.7843137255	{'max_features': None, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 100,

					'warm_start': True}
Gradient Boosting Classifier, new split, Standard Scaler, dummies	variance, pid dummies	0.8928571429	0.9272727273	0.76	
Gradient Boosting Classifier, new split, MinMax Scaler, dummies	variance, pid dummies	0.9010989011	0.9333333333	0.7755102041	
Gradient Boosting Classifier, new split, dummies, exclude user	variance, pid dummies	0.9463722397	0.9305555556	0.7826086957	
Gradient Boosting Classifier, new split, dummies	variance, pid dummies, phone type	0.9148351648	0.9151515152	0.72	
Gradient Boosting Classifier, new split, dummies	variance, pid dummies, phone type, crazy_user	0.9148351648	0.9151515152	0.72	