ManDown Machine Learning Component Report

Max Poynton (map213)

***Abstract*—To provide remedial feedback for young drinkers, it is important for ManDown to flexibly determine the level of intoxication of a user. In this report I outline the requirements of the submodule, describe suitable machine learning algorithms and relevant background, and explain how machine learning will fit into the system as a whole in its final implementation.**

I. Introduction

A key function of the ManDown app is to transform the level of intoxication it perceives into the provision of useful output. This output may range from less serious recommendations such as how many more units of alcohol it would be safe to drink or explaining the dangers of alcohol abuse, to alarms warning other users in the vicinity that a user is at risk. The user will also be able to engage in social games with the aim of increasing engagement with the app.

In order to test the first of our four hypotheses, namely that data collected through a standard smartphone and wearables will provide good indicators for the level of intoxication of the user, it is obviously necessary for the app to have some method of producing a prediction for the level of intoxication based on the data. This is the task of machine learning. The data gathered from the smartphone and wearable will be used to create a classification model. Various machine learning algorithms are available in order to generate this. Furthermore, machine learning techniques can be used to apply this model to subsequent ”live” data, and provide a classification to the data. In the context of ManDown the classification will be a level of intoxication, and this information will be used to prompt the app to decide when to provide the aforementioned warnings.

II. Specification and Integration with Other Subsystems

The smartphone and wearable will collect accelerometer and gyroscope data from the sensors. Furthermore, users will interact with the app through the use of games, some of which will be used to capture data, specifically reaction time and balance. Once a sufficient set of data has been collected from testing on participants, these data will be organised to reduce complexity and machine learning will be used to generate a classification model. This model will subsequently be applied to data from participants in further experiments with the aim of distinguishing between those who are safe and those who are intoxicated.

As the app is used in a live scenario, the data captured will be sent to a database having been pre-processed to remove noise and extract features. The machine learning subsystem will request this data and classify it, writing the result to the database. The app will then read this classification and offer some feedback to the user.

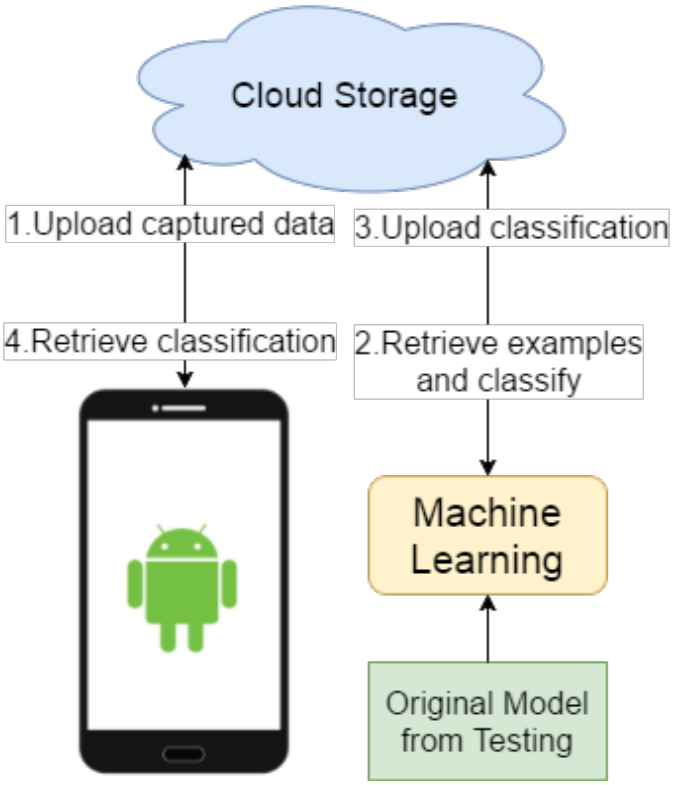

Figure 1. The Role of Machine Learning in ManDown

Ideally, the system would be capable of determining whether the user was sober, mildly intoxicated, or heavily intoxicated. This would allow the system to offer progressively severe responses. For example, if the user is only mildly intoxicated such that is does not impair their reaction time or motor abilities to the point where it may cause them harm, the app would react with helpful guidance. On the other hand, if the level of intoxication was perceived to be more severe, the app may react more strongly, alerting users in the vicinity, or suggesting the user call a taxi and stop drinking. However, this may not be possible due to the restraints of the likely small dataset that will be obtained. It may be the case that the sample size is simply too small to be able to accurately provide more than a binary classification. In that case, the specification will be reduced to requiring a binary classification space, i.e. ”not intoxicated”, and ”intoxicated”.

By the time the sensor data reaches this subsystem, it is specified that unwanted noise will be removed. As described in the design report for ManDown, there will be an initial data gathering experiment which will result in labelled examples of phone, wearable, and game data.

The machine learning subsystem should output either a

level of intoxication or an indication of whether the user is intoxicated or not, as mentioned above. This result will be passed to the database subsystem, where it will be accessible by the rest of the system. In particular, the UI subsystem will use this information as a trigger to inform the user of the fact that they nearing/past intoxication in whatever way is deemed appropriate or engaging.

The user will also be able to provide their own data on how much alcohol they have consumed etc. through the use of prompts in the app. They will also provide description of their own perception of intoxication. This allows the original model to be refined to each particular user as it increased the available data upon which the model can be trained. The machine learning subsystem will therefore be required to provide a model which can be refined after the initial training on the data from the data gathering experiment. Individual users should have a model unique to them, therefore the models will likely be stored locally on the smartphone.

Something to consider is that the user may not want to or be able to complete all the tests (games). There are a few ways to get around this. Either the assumption can be made that the user will perform all the tests. Alternatively, separate models can be learned, one for just the accelerometer data, and one for the accelerometer data combined with the results of the tests. Finally, there have been algorithms devised for use with incomplete data[4], specifically a logistic regression classifier, which could be used.

III. Other Research in the Field

The general problem can be reduced to obtaining a set of labelled training data, applying a machine learning algorithm to these data to produce a classification model, and finally using the model to predict the classification of new unseen data. Labelled training data in this context refers to pairs of sets of data describing all the measured parameters of a user at a specific time, and a known classification of that user at that time, i.e. intoxicated or not. There have been a few previous studies dealing with this problem as it applies to intoxication. Nassi et al [12] had much success in using several wearables to capture data relating to the motion of a user, and generating a machine learning model. Their findings indicate that one wearable alone is not sufficient to describe the gait of a person in order to predict intoxication through their gait. They also specifically mention that a smartphone and smartwatch combination is sufficient for accurate classification.

A. Feature Extraction

A problem which arises from continuous data such as those from an accelerometer is that of how to group the data before machine learning is performed. It is completely unfeasible to generate a model for every instant of data returned; it would also be surprising if this model would give any good indication of a particular gait, and subsequently a good indication of intoxication - the data at a particular instance only captures the concept of a single movement rather than the general degradation of the motor control system caused by intoxication. Nassi et al pioneer an interesting technique to solve this problem. The vector resulting from the individual x, y and z components of the accelerometer at one point in time is stored. The same vector from a point in the future is also stored. The difference is calculated, and this is the statistic on which the model is trained.

Arnold et al. provide a different solution [11]. The time and frequency spectrum of the accelerometer data is examined. Information such as the number of steps taken per time window can be calculated from identifying statistically high maxima in the vertical acceleration, for example.

According to Kao et al's findings[8], step time variance and the length of gait stretch are positive indicators of a gait of one who is intoxicated. Lin et al also employ this concept of feature extraction of accelerometer data to determine what movement the user is making and then use these features for training rather than the raw data itself [16]. In a similar study [3] it was also found that sway amplitude and velocity explained 92% of postural sway. Arnold et al included gait velocity, step time and step length in their list of extracted features [11]. This information will be useful when deciding exactly what features to extract from the accelerometer and gyroscope data. Finally, Nassi et al[12] set out to determine which combination of features leads to the best classification accuracy. This is a relevant point; the algorithm itself may not be the only parameter - the data plays a large part, too.

This concept of extracting meaningful features from high-dimensional data is known as dimensionality reduction and can aid in the reduction of noise and lead to an improvement in classification accuracy[10].

When it comes to the data collected from the app's in-built games, features will also be extracted rather than using the reaction times etc. themselves similarly to the discussion of the continuous sensor data above. Taking balance as an example, the variance of distance of the ball from the centre of the screen could be calculated and used as a feature. This will require the entire game to be completed, so this dataset would not be as continuous as others.

B. Machine Learning Techniques

Simple machine learning techniques are often used for such feature combination. In a study involving determining intoxication levels based on speaker state, techniques such as bag of words and bag of features were used[9].

Overfitting is an unfortunate result of using insufficient or incorrect data. If it is not possible to completely classify data based upon the examples given, a machine learning training model may be susceptible[7] to overfitting. The result is that the accuracy of the model will converge for the given set of training data, but will not generalise to new test data, and will actually begin to decrease in classification accuracy as it gets closer and closer to optimal training data classification.

Cross validation is a technique used to estimate how well a model will generalise to test data. There is often a problem of overfitting a model to a particular set of training data, particularly if that dataset is small, as it will likely be in this case. The k-fold cross-validation method involves splitting the training data into smaller sections of training data, and using

the remainder as test data [6], then repeating with a different subset of the data being chosen to be the training data until all data has been used.

C. Basic Relevant Algorithms

Since the results of the games and the data received from the accelerometer will be a set of discrete information, e.g. reaction time, score, velocity, etc., the machine learning problem is essentially one with categorical dependent variable requiring binary/three-class classification.

There are many machine learning algorithms available, however based on the fact that we are using supervised learning, that we have a binary classification problem, and that we have a small dataset, the following algorithms were found to be the most suitable, attending to the findings of Caruana et al[5]:

- Decision trees/random forest
- Bayesian network
- Support vector machine
- Neural network
- Logistic regression

Neural networks would provide a good model, but require very large amounts of data, which we will not have. Support vector machines will also provide a good model, and will work on smaller data sets, however, they are memory intensive, which may be a problem since our app is intended to run on a smartphone. However, if the machine learning processing is done in the Cloud, SVMs appear to be a good choice. On the other hand, Bayesian networks are simple to implement and have quick convergence with small datasets. Since it is intended that the algorithm will need to take new data into account to tailor the model for each user, logistic regression may be beneficial. Decision trees require the tree to be recalculated whenever new data is available for training, so this may lead to interruptions in the app due to intensive processing or delays as messages are sent to the Cloud. However, they are parametric, so outliers are not an issue. Furthermore, it is desired that suitable amounts of data are aggregated before being retraining. This batch method may reduce the aforementioned drawbacks of decision trees.

D. State-of-the-Art

Many state-of-the-art advances in machine learning involve algorithms which can be applied to existing machine learning algorithms. The concept is referred to as ”boosting”, describing the notion of improving upon a pre-existing algorithm arbitrarily[2]. In their study, Nassi et al[12] use such algorithms, namely GBR (gradient boosting regression) and ABR (AdaBoost regression).

Gradient boosting regression can abstractly be described as being concerned with formulating an optimisation problem on a devised cost function. The process will produce an ensemble of weak classifiers which as a whole form a new classifier for the data. Ensemble learning is the practice of using multiple learning algorithms in order to produce a result that demonstrates more accuracy together than each algorithm separately.

AdaBoost is short for adaptive boosting and is similar to GBR in that it produces a weighted sum of multiple different algorithms. It is considered the best approach to data whose reaction to particular machine learning techniques is unknown; this is likely why it was used in the study. The study’s analysis of the mean absolute error of each algorithm used showed that performance is different in different contexts and there may not be a ”best” algorithm in every case.

In a study involving the classification of intoxication in response to speaker state, additive logistic regression was used[9] once features of the speaker’s voice had been extracted using techniques such as bag of words and bag of features. Additive logistic regression (also known as LogitBoost) is a further extension of AdaBoost[1]. It applies the output of an AdaBoost interpretation of the classification model to attempt to minimise the probability that a particular label will be produced. It then uses this to minimise the logistic loss by minimising the probability of an incorrect label.

IV. Proposed Implementation

Data recorded from an initial bout of testing will be used to train a general model. Data captured for an individual (during the data gathering experiment) and the fact of whether they were actually intoxicated or not (as determined by breathalyzer) will be used to define a set of rules (the model) to determine whether a subsequent set of data is likely to describe an intoxicated person or not. This model will be the one by default on the app whenever it is downloaded by a user. As mentioned above, however; the user will be able to provide their own examples - the app will capture the relevant data, and the user will specify whether they deem themselves intoxicated or not. This is due to possible individual differences in how people psychologically and physiologically react to intoxication.

Since it has been specified that the machine learning model will be continuously refined for each particular user as they provide further data, it is necessary for the system to be able to regenerate a model at any time. At this point in the progression of the project, it can not be confirmed whether this, or indeed the process of classification, will be performed locally on the user’s device, or in the Cloud. There are advantages to performing this operation on the Cloud. Battery life is a major concern for this app. Users will not take kindly to it draining their battery, and this may lead to discontinuation of use. Therefore, it is preferable to perform intensive calculations such as those involved in the training of a model and classification off-device. Furthermore, it may be the case that these calculations are too computationally intensive to even run on many smartphones without causing delays.

It is also the case that only the individual user’s own data may be stored on the smartphone. Due to privacy and the general structure of the database infrastructure, it is necessary that each individual should not have access to any others’ data. Therefore, if data from more than one individual were to be aggregated and combined in order to learn some more complex model, it would make sense to do so in the Cloud, where all the data is stored.

There are two distinct uses of the Cloud in this context. Either dedicated Cloud machine learning services can be used, or regular machine learning libraries can be implemented using Cloud-hosted machines. Several Cloud APIs for machine learning exist, including Amazon Machine Learning [13] and Google Cloud Machine Learning [14]. Commonly used Cloud computing hosting is available from Amazon Web Services and Microsoft Azure. Finally, there are several options for from-scratch offline coding for machine learning. Most notably, OpenCV contains offerings of many machine learning library functions, and TensorFlow was created originally solely with machine learning in mind.

Since the preferred choice of database is Firebase, Google Cloud may be the best option from the Cloud API family. Firebase is very easily integrated with Google Cloud Platform projects. It also offers an intuitive UI and support for a number of programming languages, including Java, which will be the main language for the whole system, as the app will be running on Android.

Both OpenCV and TensorFlow support Java and Android. However, there is a GitHub project available which allows Firebase integration for TensorFlow [15].

Due to the small amount of data we are likely to be able to collect, a decision tree would make a conservative classification algorithm, being relatively simple to implement and having the possibility to become a random forest algorithm to reduce the possibility of overfitting. It is possible that data will not be collected from all the sources at each point, for example the accelerometer data may be noisy and removed for a certain datapoint. Decision trees are robust to this lack of consistency in input allowing for "don't care" inputs both in training and classification.

If the algorithm implemented turns out to be able to provide three or more classifications as mentioned above, in order to identify whether it will be appropriate to use, a confusion matrix may be used. The matrix compares results of classification by the model with the actual label assigned. This provides an indicator of which labels are commonly being confused by the model and how often. This information may be used to sway the decision as to whether not to include this functionality.

For the same reason of a likely small dataset, cross-validation techniques described above will be employed in order to predict the accuracy of the model when applied to new data. Finally, boosting techniques may be used to further improve classification accuracy.

V. Conclusion

Machine learning techniques can classify levels of user intoxication using data collected from the ManDown app. Algorithms can be narrowed down to one or two most suitable for this project and can be performed using a variety of different methods, the advantages of which have been discussed. The next steps are to collect data to generate a classification model, and to integrate this subsystem such that it uses data from the sensors, and provides output to send to the database.

References

[1] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)". In: *The annals of statistics* 28.2 (2000), pp. 337–407.

[2] Robert E Schapire. "The boosting approach to machine learning: An overview". In: *Nonlinear estimation and classification*. Springer, 2003, pp. 149–171.

[3] Michael J Pavol. "Detecting and understanding differences in postural sway. Focus on "A new interpretation of spontaneous sway measures based on a simple model of human postural control"". In: *Journal of neurophysiology* 93.1 (2005), pp. 20–21.

[4] David Williams et al. "Incomplete-data classification using logistic regression". In: *Proceedings of the 22nd International Conference on Machine learning*. ACM. 2005, pp. 972–979.

[5] Rich Caruana and Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning algorithms". In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 161–168.

[6] Payam Refaeilzadeh, Lei Tang, and Huan Liu. "Cross-validation". In: *Encyclopedia of database systems*. Springer, 2009, pp. 532–538.

[7] Pedro Domingos. "A few useful things to know about machine learning". In: *Communications of the ACM* 55.10 (2012), pp. 78–87.

[8] Hsin-Liu Cindy Kao et al. "Phone-based gait analysis to detect alcohol usage". In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM. 2012, pp. 661–662.

[9] William Yang Wang et al. "Automatic detection of speaker state: Lexical, prosodic, and phonetic approaches to level-of-interest and intoxication classification". In: *Computer Speech & Language* 27.1 (2013), pp. 168–189.

[10] Weiran Wang and Miguel A Carreira-Perpinán. "The Role of Dimensionality Reduction in Classification." In: *AAAI*. 2014, pp. 2128–2134.

[11] Zachary Arnold, Danielle Larose, and Emmanuel Agu. "Smartphone inference of alcohol consumption levels from gait". In: *Healthcare Informatics (ICHI), 2015 International Conference on*. IEEE. 2015, pp. 417–426.

[12] Ben Nassi, Lior Rokach, and Yuval Elovici. "Virtual Breathalyzer". In: *arXiv preprint arXiv:1612.05083* (2016).

[13] Amazon. *Amazon Machine Learning*. URL: https://aws.amazon.com/machine-learning/.

[14] Google. *Cloud Machine Learning Platform*. URL: https://cloud.google.com/products/machine-learning/.

[15] Google. *TensorFlow*. URL: https : / / github . com / tensorflow/tensorflow/tree/master/tensorflow/examples/android.

[16] Juanying Lin, Leanne Chan, and Hong Yan. "ANDROID PLATFORM". In: *Computer Science & Information Technology* (), p. 73.