Determination of Alcohol Consumption Level Using Smartphone Data

twsl
45483159+twsl@users.noreply.github.com
University of Bremen
Bremen, Germany

holysh1t
30555166+holysh1t@users.noreply.github.com
University of Bremen
Bremen, Germany

hiraad
35299559+hiraad@users.noreply.github.com
University of Bremen
Bremen, Germany

dast96
66383908+dast96@users.noreply.github.com
University of Bremen
Bremen, Germany

ABSTRACT

It's the unfortunate reality that people drive under the influence of alcohol, leading to accidents, which may result in injuries and even fatalities. With modern technology such as smartphones becoming a norm in the current day and age, it is a responsible decision to exploit the power of technology to avoid human disasters and prevent fatalities. In our paper, we aim to do just that by following an approach that seeks to determine the alcohol consumption level using smartphone data and identify how our ensemble methods compare to the performance of the methods discussed in *Learning to Detect Heavy Drinking Episodes Using Smartphone Accelerometer Data* [14]. To achieve this, we used the available accelerometer data from the smartphone to create a regression model that predicts the transdermal alcohol concentration. For the preparation of the data set, we used various methods such as interpolation, and splitting the data into sets containing only valid data. To optimize our machine learning model, we extracted features with sliding windows. Our best regression model presented a result with 0.0058061 mean squared error but it did not match our expectations.

KEYWORDS

drinking, accelerometer data, smartphone, alcohol detection, smartphone data, sliding windows, regressor, machine learning, optuna optimize

1 INTRODUCTION

It's the unfortunate reality that people drive under the influence of alcohol, which leads to accidents that cause injuries and even fatalities. In the age of digitalization, smartphones are becoming more and more the norm. Therefore, it is a responsible decision to take advantage of this technology to prevent people from driving under the influence of alcohol. In 85% of car accidents in Germany causing personal injury, at least one person was under the influence of alcohol [10]. In addition, 4.5% of all car accidents in Germany are accidents due to alcohol [5]. To reduce the amount of accidents with alcohol involved, we propose the usage of accelerometer data as an additional indicator of alcohol consumption level.

We used a prerecorded dataset, which consists of smartphone accelerometer and transdermal alcohol concentration (TAC) data. In order to decrease the amount of drunk driving accidents, we aim to train a regressor model that predicts when someone is about to exceed the legally allowed alcohol level. On the other hand, such a solution comes with a great responsibility of protecting people's personal information, as some companies may target vulnerable users with more specific ads, insurance companies may raise the monthly bill for reckless drinking and driving under the influence. Drunk shopping is an enormous industry and further highlights the significance of responsible acting regarding our topic at hand.

> "Retailers go to great lengths to capitalize on your drunken stupor and capture a chunk of this $45B market" [8]

Even tho Killian et al. in [14] decided to use a binary prediction formulation rather than a regression formulation, because of the high variance of the accuracy of the SCRAM sensors, we wanted a solution that is not trained on a specific legal limit, and therefore can easily be adopted all over the world regardless of local the laws concerning the legal limit of alcohol consumption. Nonetheless, using the same dataset as Killian et al., we expect to achieve better performance by using a regression formulation for the problem at hand.

2 BACKGROUND

The dataset used was downloaded from the UCI Machine Learning Repository [12] and is owned by Killian et al. [14]. It contains three-axis time series accelerometer data from mobile phones and time series TAC data, which is a real-time measure of intoxication collected through the skin and was collected by a SCRAM ankle bracelet. All data is fully anonymized. A total of 20 undergrad students, 10 men and 10 women, each in their senior year, were recruited. The recordings were taking during a bar crawl. One participant couldn't install the application, such that no accelerometer data was recorded and the TAC readings for an additional six participants were deemed unusable by SCRAM. The accelerometer readings were collected at 40Hz by a self made application and periodically send to an InfluxDB server. The data was collected from a mix of 11 iOS and 2 Android phones. Meanwhile, the TAC readings were collected at 30min intervals.

Using the IR Voltage and the temperature, cleaned time-series TAC data was calculated from these raw readings. The cleaned TAC readings were processed with a zero-phase low-pass filter to smooth noise without shifting phase, then were shifted backwards by 45 minutes such that the labels more closely match the true

intoxication of the participant, since alcohol takes about 45 minutes to exit through the skin. Cleaned TAC readings which are more readily usable for processing have two columns: a timestamp and TAC reading. This results in 14057567 accelerometer readings and 715 TAC readings for a total of 13 participants. The TAC is measured in grams per deci liter where $0.08\frac{g}{dl}$ is the legal limit for intoxication while driving in California and $0.05\frac{g}{dl}$ in Germany.

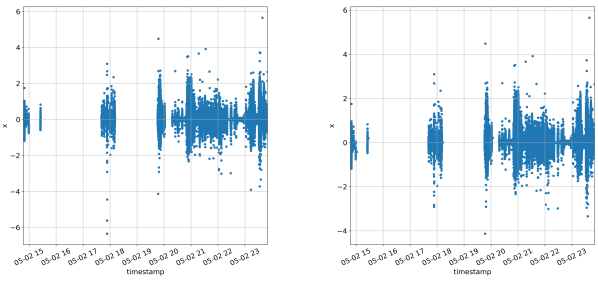
In the end there are three sets of data: three-axis time-series accelerometer, the cleaned time-series TAC and the uncleaned time-series TAC data. For further details on how the data was processed see the original paper [14].

For this particular use-case the usage of SCRAM ankle bracelets was unproblematic, but generally speaking, especially with regards to law enforcement and the current Covid pandemic, the evaluation is prone to faults by disinfectants containing alcohol like hand sanitizers and even tiny alcohol splashes. These events can falsify the readings of the aforementioned ankle bracelet.

The National Highway Traffic Safety Administration deemed the technology of transdermal alcohol measuring valid, but "the actual equipment with false-negatives rates [...] are too high" [1].

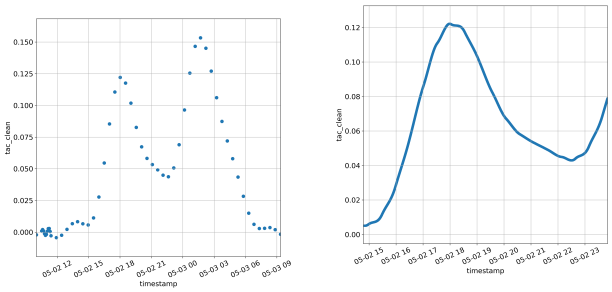
3 EXPLORATORY DATA ANALYSIS

The three sets of data explained in the last section were composed of one large, accelerometer dataset that contained data points for all the participants. And the remaining two were a raw and clean version of the TAC data per participant. In order to extract more features we had to merge the three sets into one complete dataset. The data needed some preprocessing, before feature extraction, which is the purpose of this section. The accelerometer sets (Figure 1) for one, sampled at 40Hz contained many more points as opposed to the data from the bracelets (Figure 2), captured once every 30 minutes. Another problem was that the accelerometer data often contained gaps for various reasons including phone power outages. We splitted accelerometer data to contain points for each user independently and then merged raw and clean TAC datasets together to have both attributes into one set before a final combination of all three. In order to prevent target leakage and the raw readings were later dropped again. Even though the dataset is apparently recorded at 40Hz, the merged set did not contain equally spaced timestamps,



(a) Original **(b)** Resampled and interpolated

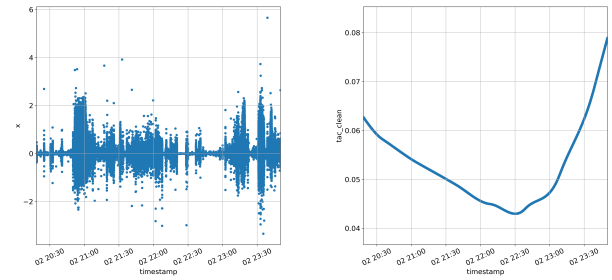
Figure 1: Comparison of accelerometer data for sample DC6359



(a) Original **(b)** Cropped, resampled and interpolated

Figure 2: Comparison of TAC data for sample DC6359

so we resampled the data at 25ms to get an equally distanced time series. We then interpolated the data linearly to makeup for the missing TAC points lying in between the frequently sampled accelerometer points. There was a threshold however in order to not interpolate over the large areas, missing accelerometer data points, so we could cut them out later. The results are visible in a few extra data points on the end of each gap in Figure 1b showing merged accelerometer data before hitting the threshold. Apart from these points the data is very similar. Another interpolation was applied to the TAC data exclusively employing the 'akima spline' method. This is visualized in the two plots shown in Figure 2. Notice that both are scatter plots, but the second one consists of much denser data points. Also, the second plot consists only of the first and the last valid accelerometer data timestamps. Finally we removed the missing gaps in between the data points by splitting the time series into chunks where only a valid combination of both, accelerometer and TAC data were present. A chunk of the same dataset is visible in Figure 3a depicting the fifth chunk of the accelerometer data and in Figure 3b the TAC clean data points of the same gap between 20:00 and 12:00. The process was repeated for each chunk of valid data until valid sets of clean, continuous data was achieved.



(a) Accelerometer **(b)** TAC

Figure 3: Chunks of final splitted data for sample DC6359

4 METHOD

In order to process the accelerometer and TAC value timeseries data, an
required. Since almost all machine learning approaches require an input of constant length, a technique such as the sliding window time series analysis, combined with sequence padding, can be used. This method enables the analysis on fractions of the time series data with a constant length. The sliding window therefore partitions the data into a number of finite-length segments and tries to relate ws past data points to a prediction y by grouping [15]. The sliding window is defined by the parameters windows size (ws) and the stride size (ss). So at any given point t_n in time t, the subset tuple

$$s_{t_n} = (\{x_{t_{n-i}}, x \in X, ws > i \geq 0\}, y_{t_n}) \quad (1)$$

is calculated. As all data points are evenly spaced out due to the resampling, we do not have to take the elapsed time between observations into account. The stride indicates the increment of n for t_n. For this regression, a window size of 10s and a stride of 2s was used. The window size was chosen as in [14] and the stride values were determined experimentally in order to create a feasible amount of data to work with that could still represent motion and does lack specific scientific justification.

4.1 Feature Extraction

By applying the sliding window with the parameters as stated above on the data resampled at 40Hz, this results in a feature matrix of 400×3 features. In order to reduce the amount of data and increase the information at the same time, a manual feature extraction phase was implemented. The extracted features are shown in table 1. Principal component analysis (PCA) is used for dimensionality reduction to avoid issues like overfitting. The three-dimensional accelerometer data is reduced into one dimension by using the PCA class from `scikit-learn` [16].

Features	Generated data	Definition
First	3	The first elements per axis
Last	3	The last elements per axis
Mean	3	The mean per axis
Median	3	The median per axis
Var	3	The variation per axis
Std	3	The standard deviation per axis
Min	3	The minimum per axis
Max	3	The maximum per axis
Argmin	3	The index of the minimum per axis
Argmax	3	The index of the maximum per axis
Sum	3	The sum per axis
Q50	3	The 50% quantile per axis
Q75	3	The 75% quantile per axis
Q25	3	The 25% quantile per axis
PCA	400	The principal component value of all 3 axis
MinPCA	1	The minimum of PCA
MaxPCA	1	The maximum of PCA
MeanPCA	1	The mean of PCA

Table 1: Features extracted per sliding window dataframe.

Additional parameters have been extracted using `tsfresh` [7], but not all algorithms have been trained with the additional features due to runtime constraints.

Features	Generated data	Description
abs_energy	3	absenergy per axis
absolute sum of changes	3	absolute sum of changes per axis
autocorrelation	3	autocorrelation per axis
binned_entropy	3	binned entropy per axis
c3	3	c3 per axis
cid_ce	3	cid ce per axis
count_above	3	count above per axis
count_above_mean	3	count above mean per axis
count_below	3	count below per axis
count_below_mean	3	count below mean per axis
first_location_of_maximum	3	first location of maximum per axis
first_location_of_minimum	3	first location of minimum per axis
kurtosis	3	kurtosis per axis
last_location_of_maximum	3	last location of maximum per axis
last_location_of_minimum	3	last location of minimum per axis
longest_strike_above_mean	3	longest strike above mean per axis
longest_strike_below_mean	3	longest strike below mean per axis
mean_abs_change	3	mean abs change per axis
mean_change	3	mean change per axis
mean_second_derivative_central	3	mean second derivative central per axis
number_crossing m	3	number crossing m per axis
number_peaks	3	number peaks per axis
skewness	3	skewness per axis
time_reversal_asymmetry_statistic	3	time reversal asymmetry statistic per axis
variation_coefficient	3	variation coefficient per axis

Table 2: Additional features extracted using `tsfresh` [7].

In comparison to [14], a few things are handled differently already: first, all of the above features are calculated on the whole window, while Killian et al. used a two-tiered window approach, a short-term and a long-term window, and some additional features, that were calculated separately [14]. Secondly, Killian et al. extracted features more specific to audio analysis. Their features sum up to 1215 data points, while we currently extract 445 with 72 additional `tsfresh` features, creating a total of 517. This already shows a significant difference between the methods used and introduces additional obstacles for comparison.

4.2 Hyperparameter Optimization

In order to create the best possible regressor, another important element besides feature extraction is the tuning of said model. While parameters are learned during training – for example the slope of linear regression or weights of neural network, hyperparameters can be arbitrarily set by a data scientist beforehand.

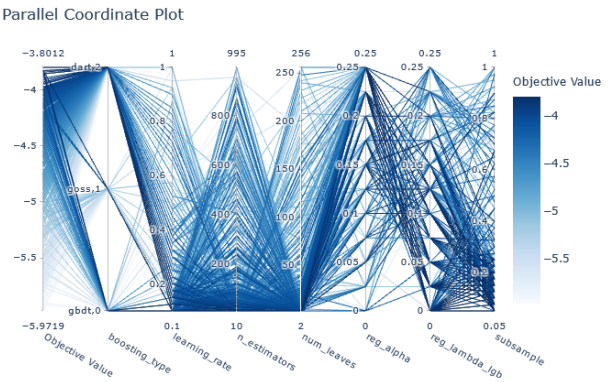


Figure 4: Exemplary HPO value plot for `LightGBM` with sample `JB3156` chunk 0

Figure 5: Exemplary HPO slice plot for `LightGBM` with sample `JB3156` chunk 0

Therefore, the selection of correct values for these hyperparameters is crucial and can significantly improve the performance of a model. This tuning process can be viewed as a type of optimization problem itself cause we aim to find the right combination of a set of hyperparameters to e.g. minimize the loss or maximize the performance metric. Apart from a manual search, random search and grid search are viable options. However, such an approach is more or less equal to a brute force attempt. Hence, so called automated hyperparameter optimization (HPO) algorithms have evolved, using techniques such as bayesian optimization, gradient descent and evolutionary algorithms. This enables efficiently searching large spaces and pruning of unpromising trials for faster results.

Multiple libraries can be used; most famous contenders `Hyperopt` [3] and `Optuna` [2] are subject of countless articles comparing them (e.g. [9]). `Optuna` was chosen based on the better documentation and build-in visualization.

A hyperparameter optimization study was performed separately for each of the algorithms presented in 4.3. Figure 4 shows a value plot for different hyperparameters for the `LightGBM` [13] algorithm, validated the chunk number 0 of sample `JB3156`. This dataset was chosen because it is the largest file after splitting.

Depending on the runtime of the algorithm and the respective features, each study was optimized for $100 - 1000$ trials.

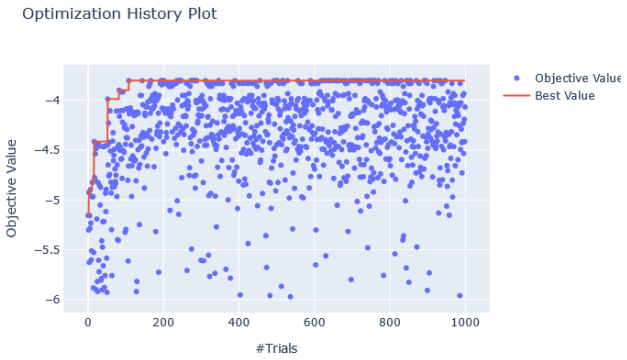


Figure 6: Exemplary HPO history plot for `LightGBM` with sample `JB3156` chunk 0

Figure 5 shows the variation of each hyperparameter over time. Whereas the final optimization history is shown in figure 6. This graph already clearly indicates, that the optimization reached a plateau quite fast and was unable to significantly improve the performance further on. In case that this model can not satisfy the desired performance requirements, further work in 4.1 is needed.

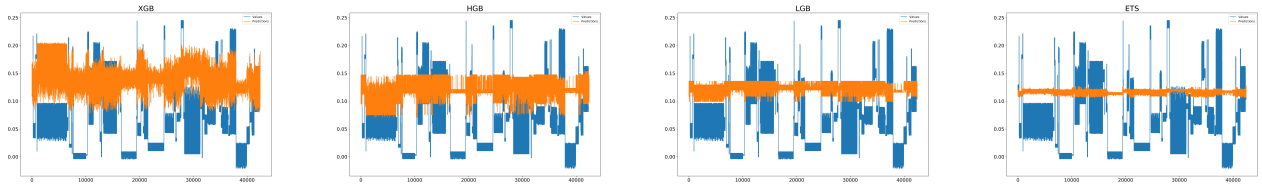
4.3 Algorithms

In ensemble learning theory, we call weak learners (or base models) models that can be used as building blocks for designing more complex models by combining several of them. Most of the time, these basics models perform not so well by themselves either because they have a high bias often due to low degree of freedom models or because they have too much variance to be robust in case of high degree of freedom models. But a low bias and a low variance, although they most often vary in opposite directions, are the two most fundamental features expected for a model. This is the well known bias-variance tradeoff.

So the idea of ensemble methods is to try reducing bias and/or variance of such weak learners by combining several of them together in order to create a strong learner (or ensemble model) that achieves better performances than any algorithm alone. One can combine these models in different ways; three major kinds are known in literature: bagging to decrease the model's variance, boosting decreasing the model's bias, and stacking to increasing the predictive force of the classifier.

Thus, in consideration of our research question, the algorithms used all implement the concept of ensemble learning. We evaluated `ExtraTrees`, `RandomForest`, `AdaBoost`, `GradientBoosting`, and `HistGradientBoosting` from the popular `scikit-learn` library [16] to establish a baseline for better comparison.

Nonetheless, one of the most popular shallow learning techniques in recent years has been gradient boosting, dominating many Kaggle (subsidiary of Google LLC, online community of data scientists and machine learning practitioners) competitions with heterogeneous tabular data. Similar to random forest, gradient boosting works by ensembling many decision trees in order to perform regression or classification. However, unlike random forest, gradient boosting grows trees sequentially, iteratively growing trees based on the residuals of the previous tree. Doing so allows gradient boosting to focus on particularly tricky observations and yields an extraordinarily powerful ensemble of trees. Among the most common algorithms in recent years are `XGBoost` [6], `LightGBM` [13] and `CatBoost` [11].



(a) XGBoost **(b)** HistoryGradientBoosting **(c)** LightGBM **(d)** ExtraTrees

Figure 7: Comparison of concatenated predictions (orange) to ground truth (blue).

All used algorithms share an implementation of the `scikit-learn` API [4], which made it easy to use a shared infrastructure.

5 RESULTS

In table 3 the mean squared error (MSE) of all trained models is shown. Clearly these errors are quite low, but in the context of our regression model the error is naturally quite low, because the mean squared error is not a relative value but an absolute value and depends on the value range which, in our case, is small.

Model	MSE
ETS	0.005806121997238007
HGB	0.006537057262386852
LGB	0.006659725937704261
XGB	0.009520274940766137

Table 3: Mean squared error of trained models

In the figure 7 the orange lines represent our prediction and the blue lines represent the ground truth. With these figures it is quite clear why the mean squared error value needs to be even lower.

Our results show that the regression is not able to differentiate between below and above the legal limit with the features extracted.

5.1 Discussion

We had a deadline to submit our results and in the mean time, the project continued to scale up as we progressed through different stages. Thus, we had to compromise a bit of quality to gain speed and finish our project on time. These difficulties and compromises include the feature extraction where we could use more promising signal processing approaches to obtain better results. More detailed description of this matter is further explained in the section 6.

In the case of an application like this to become publicly accessible to users, there comes great responsibility with the data collection of the owners. For one, this is extremely sensible data that could be trained to recognize even further information about an individual. Another point is that in case the TAC level data gets leaked or sold to a third party, it could be abused to target the user with relevant ads or give drunk users higher transportation fares. And finally, the app itself could produce defective detections, like a false negative resulting in irresponsible behavior while depending on a feedback from the application as an excuse to drive, for example.

A direct comparison of the results of our trained regression model to the model proposed in [14] is not possible, because not only did they not specify, on which chunks of the dataset they tested their trained models, they also performed a classification. And our regression has a more precise output and that results in worse metrics for our model, but does not necessarily represent the performance, because a threshold could be undershot by a tiny amount and would not necessarily mean a worse result, just a more precise one.

In addition to the aforementioned points, because this paper was created during a university course, many of our expectations couldn't be met because of strict time constraints. If more time was available we could have increased our model performance by using details we discussed in the beginning of this section and chapter 6.

5.2 Conclusion

Overall this project was a huge learning experience for all of us, and we learned a lot about data analysis. Additionally, we learned how fast a project can grow in dimension by adding more complexity and data. Furthermore we can conclude that in shallow machine learning algorithms the feature extraction has a huge impact on the model performance.

Our results were not meeting our expectations, but were of course limited by the time constraints of a university project and lacking the computational power to train more models with the proposed changes. Otherwise, we see our whole pipeline of data analysis, cleaning and model tuning as a huge success.

6 FUTURE WORK

For future work we formulate some specific ideas to further improve our machine learning model and the usability for end users. On one hand we've created an android application prototype, which can read the accelerometer data readings which are needed when using our model.

Improving the machine learning model presented in this paper could be done in several ways. One way could be to extract even more features, specifically using the accelerometer data as signal data and extract more audio features such as: Continuous wavelet transform, Mel-frequency cepstrum and many more. Together with extracting different statistical features the result could be improved and features that are deemed more usable for our regression can be used. Similarly, analyzing the accelerometer data could also be used to extract features that for example indicate if the participant

is walking, sitting or standing. And finally, features with a low correlation could be removed.

Additionally, a new dataset could be collected by recreating the original application from [14] and eliminating some shortfalls they had in collecting the data, e.g. loss of phone power, missing or completely unusable data points. Furthermore, using a more solid method of collecting either the transdermal or blood alcohol content could be used and more features could be collected, such as if the participant is taking a drink or if he is present in a bar.

Varying the sliding window length and step size could improve the regression. Using a bigger value range for the hyperparameter optimization and more trials could also help to improve the results as well. And last but not least, evaluating other types of machine learning algorithms such as deep learning models with LSTM cells is feasible.

REFERENCES

[1] National Highway Traffic Safety Administration. 2007. *Evaluating Transdermal Alcohol Measuring Devices.*

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 2623–2631. https://doi.org/10.1145/3292500.3330701

[3] James Bergstra, Daniel Yamins, and David D. Cox. 2013. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013 (JMLR Workshop and Conference Proceedings, Vol. 28)*. JMLR.org, 115–123. http://proceedings.mlr.press/v28/bergstra13.html

[4] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 108–122.

[5] Statistisches Bundesamt. 2018. *Alkoholbedingte Verkehrsunfälle nach Bundesländern 2018.* https://de.statista.com/statistik/daten/studie/180/umfrage/anteil-der-alkoholbedingten-verkehrsunfaelle-nach-bundeslaendern/

[6] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 785–794. https://doi.org/10.1145/2939672.2939785

[7] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. 2018. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh - A Python package). *Neurocomputing* 307 (2018), 72–77. https://doi.org/10.1016/j.neucom.2018.03.067

[8] Zachary Crockett. [n.d.]. *The 2019 Drunk Shopping Census.* https://thehustle.co/drunk-shopping-survey

[9] Jakub Czakon. 2019. *Optuna vs Hyperopt: Which Hyperparameter Optimization Library Should You Choose?* https://neptune.ai/blog/optuna-vs-hyperopt

[10] Statistisches Bundesamt (Destatis). 2019. *Verkehrsunfälle Unfälle unter dem Einfluss von Alkohol oder anderen berauschenden Mitteln im Straßenverkehr 2018.* https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Verkehrsunfaelle/Publikationen/Downloads-Verkehrsunfaelle/unfaelle-alkohol-5462404187004.pdf?__blob=publicationFile

[11] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. 2018. CatBoost: gradient boosting with categorical features support. *CoRR* abs/1810.11363 (2018). arXiv:1810.11363 http://arxiv.org/abs/1810.11363

[12] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[13] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 3146–3154. http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree

[14] Jackson A. Killian, Kevin M. Passino, Arnab Nandi, Danielle R. Madden, and John D. Clapp. 2019. Learning to Detect Heavy Drinking Episodes Using Smartphone Accelerometer Data. In *Proceedings of the 4th International Workshop on Knowledge Discovery in Healthcare Data co-located with the 28th International Joint Conference on Artificial Intelligence, KDH@IJCAI 2019, Macao, China, August 10th, 2019 (CEUR Workshop Proceedings, Vol. 2429)*, Nirmalie Wiratunga, Frans Coenen, and Sadiq Sani (Eds.). CEUR-WS.org, 35–42. http://ceur-ws.org/Vol-2429/paper6.pdf

[15] Ladan Mozaffari, Ahmad Mozaffari, and Nasser L. Azad. 2015. Vehicle speed prediction via a sliding-window time series analysis and an evolutionary least learning machine: A case study on San Francisco urban roads. *Engineering Science and Technology, an International Journal* 18, 2 (2015), 150 – 162. https://doi.org/10.1016/j.jestch.2014.11.002

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.