Using Machine Learning to Enhance Activity Prediction on Sensory Data*

Haohui Zhang(2722930)[1], Yiming Xu(2696855)[1], and Yilin Li(2737659)[1]

1 Vrije Universiteit Amsterdam, Netherlands

Abstract. Recently, sensory data from subjects have been widely used in predicting human behavior through using multiple machine learning methods. Considering the quantified self data is still worth to analyze in-depth, we proposed to describe an exquisite pipeline of data pre-processing, feature engineering and selection. Then, machine learning methods were evaluated and compared based on the result of prediction.

Keywords: Machine Learning · Behavior Prediction · Clustering

1 Introduction and Related Work

The concept of *quantified self* was first introduced by G.Wolf and K.Kelly in 2007[8], which describe any kind of individual biological, physical and behavioral information from self-tracking. The reason why we emphasize it is that early quantified self and sensory data contains a significant value for many AI-related tasks, especially for behavior prediction and classification where analyzing behavioral pattern is not possible without machine learning methods. Alex and Andrew[14] modeled and classified human behaviors had demonstrated great promise in establishing feature-based mathematical models, such as Markov dynamic model, to assist human behavior prediction with sensory data. However, human behavior analysis still faced challenges due to the noise in self-tracking, lack of relational data points and unexpected patterns in normal behavior.

Considering that the difficulties of adopting non-purely empirical quantified self approach for real sensory data[12], Jingcheng[3] intended to promote the performance of prediction by data fusion and feature engineering. What's more, multiple machine learning methods (e.g., decision tree, cluster, and Naive Bayesian) were imported to promote behavior prediction in Jing's work[9]. In total, most of the previous works focused on feature optimization or machine learning model design to enhance the accuracy of behavior prediction. In our project, we intend to analyze a sensory data concerning multiple users which have posted in a Kaggle competition[1] by executing all major steps of ML4QS pipeline[5]. In detail, data aggregation, outlier detection, missing value imputation, feature engineering, and selection are implemented in our work. Then, not only stressing on the importance of pre-processing, clustering algorithm[16], which is suitable

* Supported by Machine Learning for the Quantified Self, Vrije Universiteit Amsterdam

for multi-subject dataset and expresses characteristics of the data objectively, is used to predict human behavior. To provide a detailed evaluation, ensemble machine learning-based algorithms[4] are also imported to produce a prediction and compared to the benchmark at last. The overview is shown in Fig.1.

2 Data Exploration & Preparation

2.1 Dataset Description

The **MotionSense** Dataset[2] contains time-series sensory data which is collected by monitoring `accelerometer`, `attitude`, `gravity` and `rotationRate sensor`. All valid data collected in 50Hz sample rate through bundling the iOS devices with 24 subjects in various `Age, Gender, Height` and `Weight`. Each subject performed 6 different activities (i.e., Downstairs, Upstairs, Walking, Jogging, Sitting, and Standing) in 15 continuous trials. For our initial exploration, we aim to capture how many occurrences there are per column and what the size of the data is. Details are shown on the top panel of Table.1.

Table 1: Overview of dataset after different phases. *: `accelerometer`, `attitude`, `gravity` and `rotationRate` data on different axes. **: `Age, Gender, Height` and `Weight` per subject. #: Window size is 30 or 40. 75Hz for collecting features.

Initial Data: 50Hz	Description	Type	Rows per subject
Basic sensory data	Collected from sensor	float [-5,5]	$51,006 - 71,949$
Subject-related data**	Collected from sensor	integer & bool	$51,006 - 71,949$
Aggregated Data: 200ms	Description	Type	Rows per subject
Basic sensory data	Aggregated and avg...	float [-5,5]	$5,099 - 7,194$
Subject-related data	Collected from sensor	integer & bool	$5,099 - 7,194$
Engineered data#	Description	Type	Rows per subject
Subject-related data	Aggregated & avg...	integer & bool	$365 - 514$
Time domain features	Aggregated & mean...	float [-2.5,2.5]	$365 - 514$
Frequency domain features	Fourier transform...	float [0,100] [-5,5]	$365 - 514$

2.2 Feature Aggregation and Outlier Detection

Now, a set of 20-minutes measurements for each subject's is obtained by us. Considering that raw dataset only expresses the independent state of each time point, we intend to optimize the dataset and convert it to temporal data by setting a suitable time step size (i.e., granularity Δt). In detail, numerical attributes, such as `accelerometer` and `attitude`, are aggregated by averaging the values in the interval. In terms of categorical attributes, the value **True** is assigned to each label of activity if the subject performs the corresponding activity during the interval (i.e. *Binary method* in Mark's work[5]).

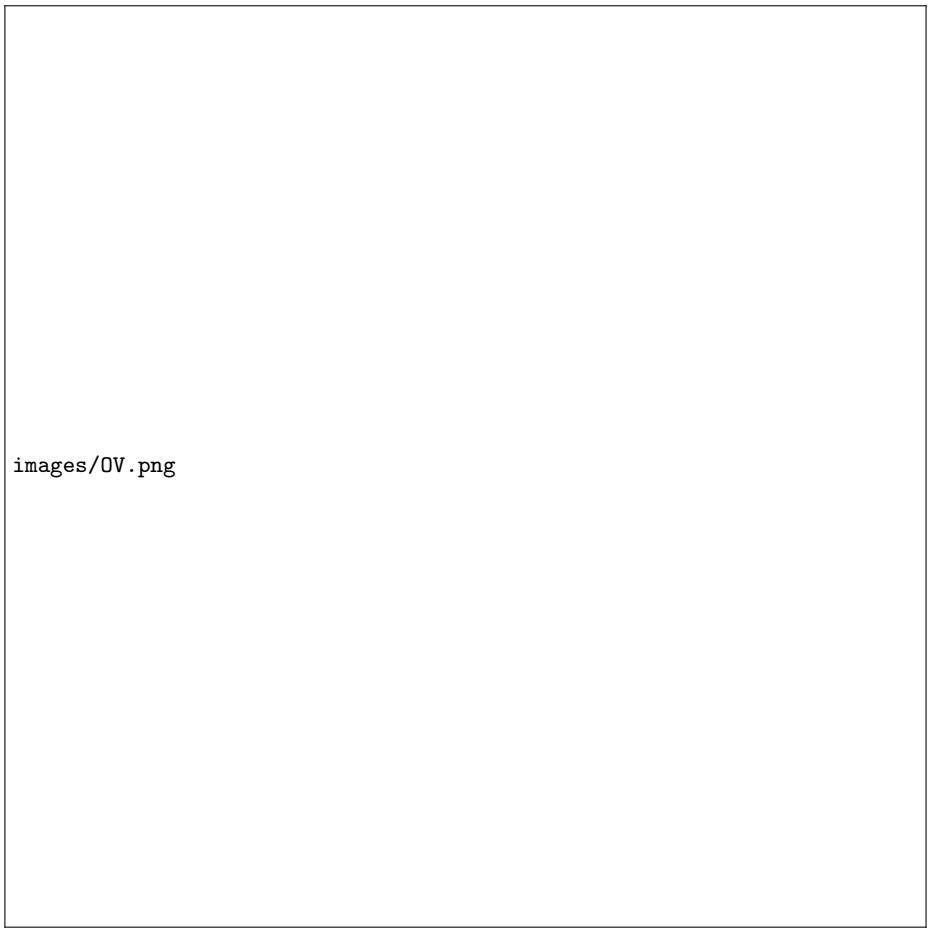

Fig. 1: The overview of pipeline in our work. Blue Panel: After feature engineering, we first cluster the dataset by `User` to generate a new attribute `User_state` to aid prediction.Then, Datapoint Cluster divided dataset into 18 groups and 18 models are trained with corresponding group of data for prediction. Red Panel: a LightGBM is trained with the whole dataset. When a row comes as an input, we first assign it in a specific group and integrate the result from group model and the whole model by using ensemble learning. Details shows in Section 3.4.

In our case, after a careful comparison between various granularity, we find that the patterns of most sensory data fit well with the change of behavioral activities when taking a Δt of 200ms. In detail, each type of sensory data has a different pattern when subjects perform different activities. During continuous motion (e.g., walking, upstairs, etc.), the changes in sensory data can be described as pulse waves or sinusoidal functions with different amplitudes and frequencies. At rest (i.e., standing and sitting), the different sensory data have different static values and are perturbed by noise. Details are shown on Fig.2.(a).

In the process of aggregation and temporal dataset generation, many outliers are found in the interval of various behavior activities. This is because the sensors recorded the measurement error of the device and the unexpected intensity of the subject's activities. Then, we propose to use Chauvenets criterion [6] for outliers removal with a relatively high $c = 9$. The reason for selecting Chauvenets criterion is that extracting sensory data from subjects is an extremely complex activity and highly susceptible to the experimental setting, so designing a distribution centered on the mean of a normal distribution can statistically reject suspicious data points. Besides, with the aim of handling subtle noise, we first impute missing value based on previous state and then implemented low-pass filter (i.e., remove high-frequency noise) and principal component analysis which works on single attribute and across the entire dataset. Specifically, `accelerometer`, `gravity` and `rotationRate` are selected to execute lowpass filter. Next, all attributes in dataset (except labels) are treated as input to generate the principal component which is evaluated by the explained variance. Seven components are created to assist the prediction in the following steps. Fig.2.(b) (bottom panel) summarized the changes of `use_Acceleration` after outlier removal.

2.3 Feature Engineering

Instead of only processing existed features, the implementation of feature generation also helps in promoting performance. In our work, both time and frequency domain features are generated by implementing relevant methods. The standard deviation (Std) and median are created by setting a suitable window size $W = 30$. Std describes the dispersion of the data over time , while the median shows the general observation over a set of time points. What's more, Fourier Transformation is used to calculate the amplitudes of a specific frequency which illustrates a representative pattern over the change of activities. In this work, we intend to select a different window size, which is $W = 40$. After experiments, the value of power spectral entropy (PSE) and frequency weighted fitted well in the change of behavioral activities when taking a frequency of 75Hz. Details are shown in Fig.3. Specifically, the curves for each feature in Fig.3 can be fitted to functions or specific static values when the subject performs different activities.

2.4 Feature Selection

Details of the procedure are presented in Table.2 which consists of two steps:

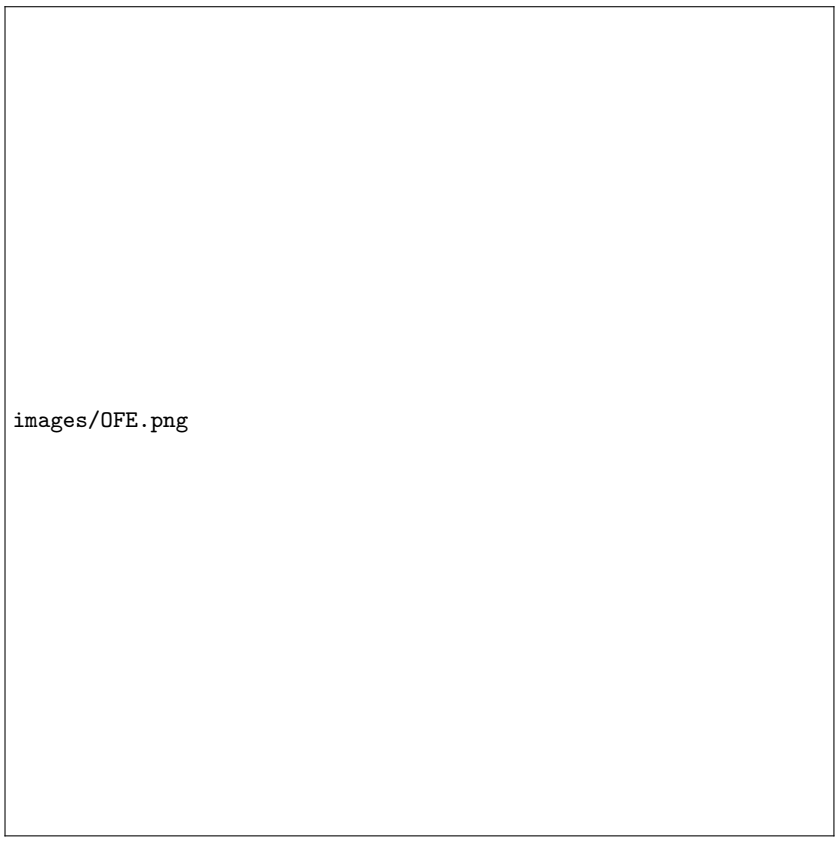

Fig. 2: (a): The overview of temporal dataset which generated by aggregating in Δt=200ms. Each attribute shows a clear pattern over the change of activities. (b): The processed curve of `user_Acceleration.x` after outlier removal. The process of removal eliminates the out-of-domain values and high frequency noise.

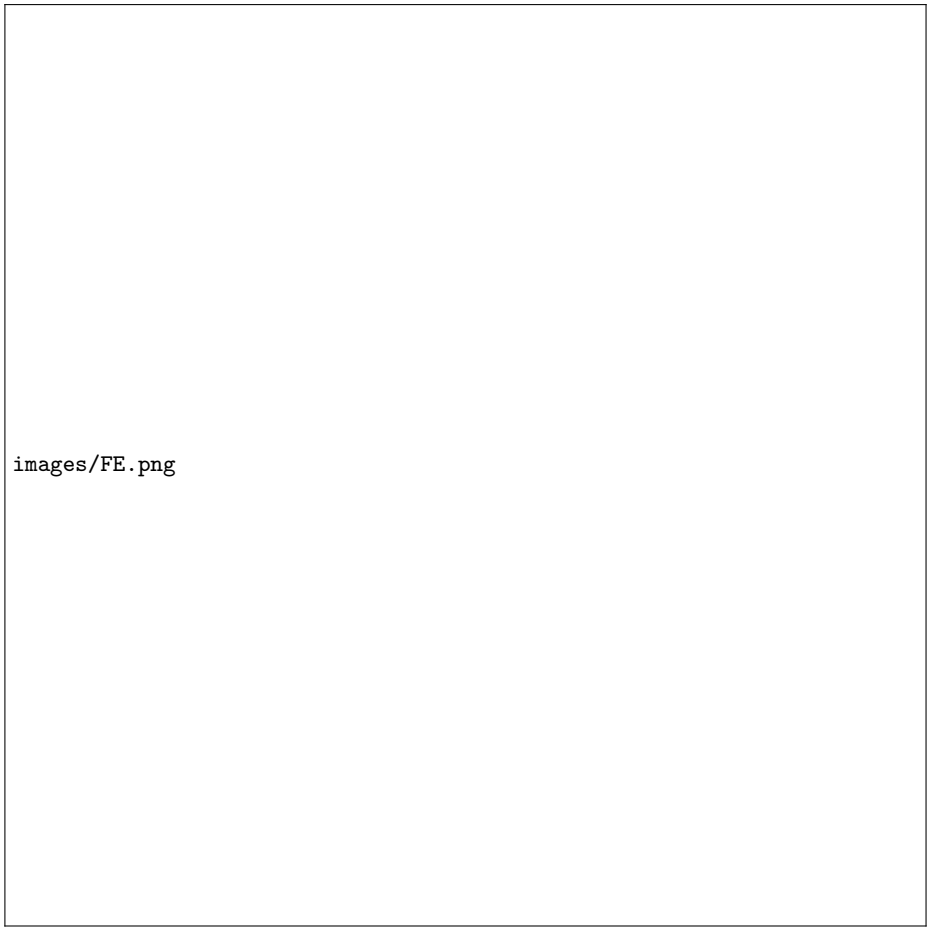

Fig. 3: (a): The overview of selected time-domain features. $W = 30$ (b): The overview of selected frequency-domain features. $W = 40$, $f = 75Hz$. For example, `attitude.yaw_pse` exhibit Boxcar functions of different frequencies as well as amplitudes under multiple activities.

1) After feature engineering, Pearson correlation coefficient is first calculated to evaluate the correlation between different attributes. Then, we manually select representative features and remove some existed attributes that are too relevant based on the patterns shows in Fig.3.

2) We apply **k-means** to add two more features and use mutual information, correlation and backward selection based on decision tree for further selection.

Table 2: The table presents input features, algorithm, feature number after each algorithm and is assigned *Code* for *feature_after*. The first set is the attributes provided by **MotionSense**; the second set represents the use of the provided features to go through the previous feature engineering steps; the third and forth sets represent the use of four selection algorithms; the sixth set represents the procedure of two cluster algorithm; the fifth and seventh sets are the three sets that used a comparison to train our model and perform an evaluation.

set	feature_input	(Selection) Algorithm	#feature_after	Code
1	basic_13	provided feature	13	basic_13
1	user_5	provided subject feature	5	user_5
2	basic_13	feature engineering	46	basic_46
3	basic_46	Pearson correlation	31	selected_31
4	selected_31	mutual information	10	mutual_10
4	selected_31	correlation	10	corr_10
4	selected_31	backward selection (decision tree)	10	back_10
5	selected_31	mutual_10 $\cup$ corr_10 $\cup$ back_10	21	selected_21
6	user_5	**kmeans**(mutual_10 $\cap$ corr_10 $\cup$ user_5)	1	cluster_1
6	selected_21	**kmeans**(selected_21 $\cup$ cluster_1)	1	cluster_2
7	selected_21	selected_21 $\cup$ cluster_1 $\cup$ cluster_2	23	final_23
7	selected_31	selected_31 $\cup$ cluster_1 $\cup$ cluster_2	33	final_33

3 Algorithm & Model

3.1 Train-test Split

Considering the prediction task may have two situations, separately predicting activity for a new subject or predicting activity for an existing subject, therefore, we split our dataset that contains 24 individual data into three sets. We initially randomly split the data of three individuals into a set *test1*. And we split the rest into the set *train* and the set *test2* based on the ratio of 2:8. Note that each of the individual data in three sets contain data for all 6 labels. Furthermore, we also employ K-fold cross-validation during the model training process. Each time, one of the five sections of the training set is randomly chosen for testing, while the other four are used for training. Finally, we have five sets of results. The final outcome used to assess the effectiveness of the model is the average value of each indicator.

3.2 Clustering Algorithm

After splitting the original dataset into *train*, *test1* and *test2*, we use the *train* set to perform two clusterings, both using the **k-means** algorithm. We first concatenate *train* with the provided subject information containing 5 features respectively `subject id`, `height`, `weight`, `age` and `gender`. We aggregate the concatenation set according to `subject id` grouped by mean and cluster the 21 subjects into 3 clusters. The clusters amount k is decided by the Sihouette score of k-means. This clustering result is used as a new feature. As for the second clustering, the purpose is to cluster all the data points into several clusters whose result can be used as a new feature and be used in the ensemble learning. To figure out the best setting for the clusters amount k, we run the algorithm with k ranging 2 from 25 shown in Fig.4a (using *selected_21* $\cup$ *cluster_1* result), and measure the Sihouette score to judge the quality. We finally select 18 for k which can be interpreted as strong, medium, and weak activity level for all 6 labels. The clustering performance is visualized using T-SNE shown in Fig.4b. Two test sets' labels are predicted based on the two **k-means** models we learnt.

(a) Sihouette score of k-means for different values of k

(b) Visualization of the clusters found with k-means with k=18(colors) using TSNE to reduce the number of dimensions to 2

Fig. 4: Visualization of second clustering for cluster all data points into 18 clusters

3.3 Benchmark

We first design a **Decision Tree** model[15] as a baseline and train the model on *selected_31*. Then, two other models called **Gradient Boosting Classifier(GBC)**[13] and **LightGBM(LGBM)**[7] are also built for prediction. To train these models, we use the features that are extracted using 3 different methods.

For **GBC**, we set learning_rate to 0.1, n_estimators to 20, mi_samples_split to 100. For **LGBM**, we set learning_rate to 0.01 and n_estimators to 5000. The results of our model will be detailed in the next section.

3.4 Ensemble Learning Models

The pipeline of the ensemble model is shown in Fig.1. The main idea behind this is to assign a weight *w1* to the probability predicted from a **LGBM** training on the entire training set and to assign another weight *w2* to the probability predicted from a **LGBM** training on its cluster set. Finally, *w1* and *w2* are trained using a **Multilayer perceptron(MLP)**. You can find the code in https://github.com/HarryZhangHH/ML4QS.

4 Results and Evaluation

Table.3 shows the result of our models, when we use the *final_33* features and **LGBM**, our model achieves the best performance. Details are shown in Fig.3

Table 3: The results of benchmark and our models in three kinds of feature. Both **GBC** and **LGBM** perform poorly when we use *final_23*, probably because the backward selection method removes important features. The two models perform well on *select_31* as well, but not better than the results in *final_33*.

	test1		test2	
	F1	ACC	F1	ACC
Benchmark	0.7057	0.7759	0.8604	0.9009
gbc [selected_31]	0.8667	0.8973	0.9176	0.9443
lgbm [selected_31]	0.9110	**0.9316**	0.9690	0.9780
lgbm [back_10]	0.5085	0.5737	0.7336	0.8032
gbc [final_23]	0.8578	0.8854	0.9138	0.9383
lgbm [final_23]	0.8958	0.9162	0.9628	0.9734
lgbm [final_33]	**0.9150**	0.9314	**0.9770**	**0.9836**
ensemble [final_33]	0.8130	0.8161	0.9350	0.9311

As can be observed, the features `acc_userAcceleration.y_temp_std_ws_30` and `mag_rot` in Fig.5(a) are crucial for classes 0 - 4 and class 5, respectively. Fig.5(b) and Fig.5(c) show the confusion matrix and feature importance of **LGBM** on *final_33* features separately, once again demonstrating the significance of `acc_userAcc- eleration.y_temp_std_ws_30` for the classification model and our model performance good on classifier class 0, class 1 and class 3.

5 Discussion

The task of this project is to use machine learning model to classify the activity in the sensor data of the quantified self. Except for implementing the decision

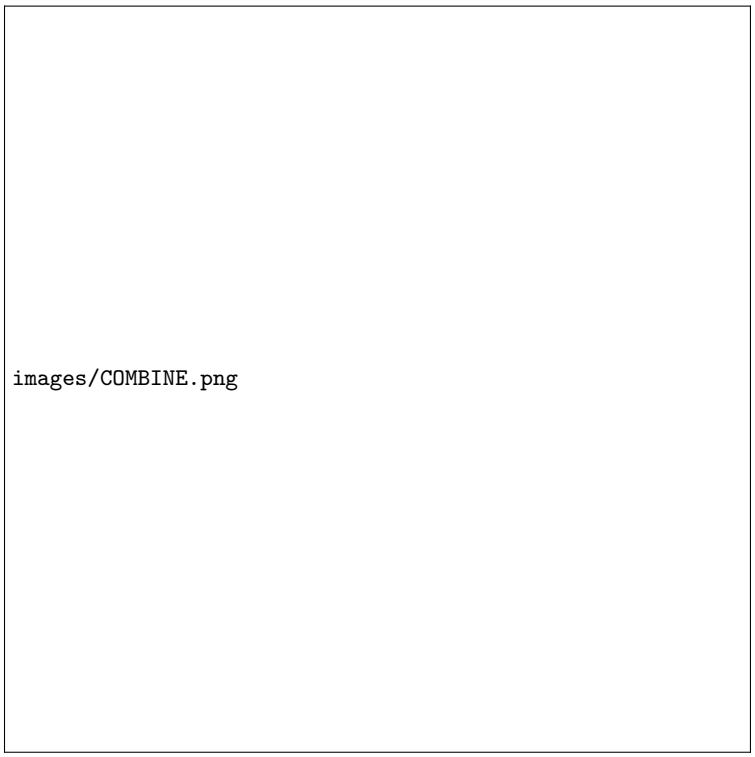

Fig. 5: (a) SHAP value[11] [10] of **LGBM** using *final_33* (b) Feature importance of **LGBM** using *final_33* (c) Confusion matrix of **LGBM** using *final_33*

tree model and two advanced tree models **LGBM** and **GBC**, we also tried to implement an ensemble model based on **LGBM** and **MLP**. We can observe from Table.3 that the results of our ensemble model are not ideal, this is most likely because the ensemble model overfits the training set, which is probability caused by the lack of enough training data or the excess of invalid clusters. Although the performance becomes worse, it is still worthy to try to predict a label or numerical value combining the information learnt from whole dataset with the information learnt from its specific cluster. For future study, we can adjust the ensemble model to predict the Q value for the **function approximated Q-learning network** to provide feedback and support to the user.

Bibliography

[1] Kaggle competition - motionsense dataset, https://www.kaggle.com/datasets/malekzadeh/motionsense-dataset Accessed June, 2018

[2] Motionsense dataset, https://github.com/mmalekzadeh/motion-sense Accessed October, 2021

[3] Chen, J., Sun, Y., Sun, S.: Improving human activity recognition performance by data fusion and feature engineering. Sensors **21**(3), 692 (2021)

[4] Chung, Y.W., Khaki, B., Li, T., Chu, C., Gadh, R.: Ensemble machine learning-based algorithm for electric vehicle user behavior prediction. Applied Energy **254**, 113732 (2019)

[5] Hoogendoorn, M., Funk, B.: Machine learning for the quantified self. On the art of learning from sensory data (2018)

[6] Irwin, J.: On a criterion for the rejection of outlying observations. Biometrika pp. 238–250 (1925)

[7] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems **30** (2017)

[8] Kelly, K., Wolf, G.: What is the quantified self. The Quantified Self **5**, 2007 (2007)

[9] Li, J., Pan, S., Huang, L., et al.: A machine learning based method for customer behavior prediction. Tehnički vjesnik **26**(6), 1670–1676 (2019)

[10] Lundberg, S.M., Erion, G.G., Lee, S.I.: Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888 (2018)

[11] Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Advances in neural information processing systems **30** (2017)

[12] Marcengo, A., Rapp, A.: Visualization of human behavior data: the quantified self. In: Innovative approaches of data visualization and visual analytics, pp. 236–265. IGI Global (2014)

[13] Natekin, A., Knoll, A.: Gradient boosting machines, a tutorial. Frontiers in neurorobotics **7**, 21 (2013)

[14] Pentland, A., Liu, A.: Modeling and prediction of human behavior. Neural computation **11**(1), 229–242 (1999)

[15] Song, Y.Y., Ying, L.: Decision tree methods: applications for classification and prediction. Shanghai archives of psychiatry **27**(2), 130 (2015)

[16] Xu, D., Tian, Y.: A comprehensive survey of clustering algorithms. Annals of Data Science **2**(2), 165–193 (2015)