July 2018

Localized Indoor Temperature Estimation Using Smartphone and Laptop Internal Sensors

Mostafa Rafaie
University of Nebraska–Lincoln, United States of America, mostafa.rafaie@gmail.com

Mehari Tesfay
University of Nebraska–Lincoln, United States of America, mehari58@gmail.com

Fadi Alsaleem
University of Nebraska–Lincoln, United States of America, falsaleem2@unl.edu

Localized Indoor Temperature Estimation Using Smartphone and Laptop Internal Sensors

Mostafa Rafaie[1], Mehari Tesfay[2], Fadi Alsaleem[3*]

[1]University of Nebraska - Lincoln, Durham School of Architectural Engineering,
Omaha, Nebraska, USA
Contact Information (mostafa@huskers.unl.edu)

[2]University of Nebraska - Lincoln, Durham School of Architectural Engineering,
Omaha, Nebraska, USA
Contact Information (mtesfay2@unl.edu)

[3]University of Nebraska - Lincoln, Durham School of Architectural Engineering,
Omaha, Nebraska, USA
Contact Information (316-719-0450, falsaleem2@unl.edu)

* Corresponding Author

ABSTRACT

This paper investigates a mechanism in which indoor air temperature can be predicted using the temperature sensor on the battery and the CPU of smartphones or laptops, where data is easily accessible and ubiquitous. As a case study, several machine-learning methods were used to build models from a laptop data and the measured surrounding air temperature. The effects of the machine learning type and input feature size (by including other parameters such as CPU processing usage and battery charge percentage) on the model accuracy were investigated. The goal is to determine a set of feature combinations that can be used to build models which can accurately predict the indoor temperature. The accuracy of these models was measured by comparing their prediction to the actual indoor temperature.

1. INTRODUCTION

The increasing advancement in miniaturization technology and the recent surge in electronic devices equipped with built-in sensors that are capable of capturing processes in real-time has become a valuable input for research. Temperature control and monitoring have been the center of most industrial automation processes in industries and buildings. This paper investigates a mechanism in which indoor air temperature can be predicted using the battery temperature sensor of laptops or Android smartphones, except for iOS based iPhone, where data is easily accessible. This idea is different from the idea of using crowdsourcing to predict the average outdoor city temperature for meteorological purposes, using smartphones battery sensor.(Overeem et al. 2013)

In our recent work (Hasan et al., 2016) and (Rafaie et al., 2017) we have proposed an idea of blending traditional environmental sensing data streams (such as temperature) with newly available wearable sensing information to develop new models to predict thermal comfort of building occupants and to optimize building operations. However, a major challenge in that approach is the requirement for each occupant to carry a wireless temperature sensor. The idea in this paper is to replace that requirement by employing the temperature sensors in smartphones or laptops' batteries (along with other parameters) without directly measuring the temperature in the room. Moreover, this new way of sensing indoor temperature can also be used in optimizing energy efficiency for applications of high-energy demand.

It is obvious that the estimation of indoor temperature from the battery temperature would be influenced by the insulation, intensity of load, quality, and location (pocket or indoor) of the smartphone. This work intends to tackle this bias and noises by working in a higher dimensional feature space. Apart from the temperature of the battery in the cellphone, there are dozens of more features of the laptop/smartphone where online real-time data can be collected and made useful in the model construction process. The indoor air temperature estimation could be more consolidated if features such as CPU internal temperature and other processor activities are found to have a meaningful correlation with varying indoor air temperature values.

Data of more than 300 features from a running laptop along with the actual ambient air temperature had been collected during the learning phase. The data collected will be used to build non-linear models that best represents the relationship between the smartphone feature data and the indoor air temperature. In doing so, different machine learning algorithms of high-performance success will be employed to develop a dependable generalized model that works fine not only in the same environment and cellphone but also for significantly different data and new user cell- phones. It is quite intuitive to assume that different features will have different correlation degrees with the indoor air temperature. Therefore, it is imperative to give more relevance to features that show best correlations with the air temperature.

The organization of this paper is as follows. In section 2, we introduce the experiment that we have conducted for this investigation. In section 3, the experimental results and discussion are presented. In section 4, we summarize the paper and provide some conclusions.

2. EXPERIMENT AND METHODOLOGY

In this investigation, a new MacBook Pro with a built-in quad-core i7 processor and 16GB of RAM was used. Using tools such as Glances(Hennion n.d.) and iStats(Naud-Dulude n.d.), a dataset with about 300 features was collected online from this laptop with high sampling rate. Simultaneously, a Hobo data logger logs and stores the indoor temperature. Based on common sense and prior knowledge, preliminary filtering was made to reduce the number of features to eleven with a belief that these features would capture the characteristics of the indoor air temperature. Battery temperature, CPU temperature, the percentage of processor usage of three CPUs, load intensity of the last few minutes and the total number of processes running in the system are among the features chosen for the analysis.

To investigate the combination of features that most capture the indoor air temperature, the features were grouped into 255 different possible combinations, Figure-1. To avoid the omission of the features presumably considered as a relevant such as battery temperature, and to minimize the number of possible combinations features that exhibit similar behaviors, we employed the following unification rule:

1. The battery temperature and CPU temperature are mandatory
2. The three processors information is considered as a variable meaning if they include in the feature, the data of all CPUs will be incorporated.

Utilizing the Python programming language, an application called trainer is implemented to create the models and evaluate the performance. The Scikit-learn package (Pedregosa et al. 2012) is the main library to facilitate the process of creating the models. The Scikit-Learn is a Python-based program, built on top of SciPy and distributed under the 3-Clause BSD license and includes a vast collection of machine learning methods. Running the trainer application for every dataset was managed by the Holland Supercomputing Center (HCC) at the University of Nebraska. The trainer application utilizes different regression machine learning methods including Epsilon-Support Vector Regression (SVR) with kernel RBF, PLS 2 blocks regression (PLS-Regression), Nu-Support Vector Regression (Nu-SVR) with kernel RBF, K-Nearest Neighbor Regression, and the Randomized Decision Trees regressor (Extra-Trees Regressor) (Ligeza 1995). The combination of the different machine learning methods and applying different initialization variable created 33 unique models. Figure 2 shows the details of the models used in this experiment. To validate the accuracy of each model and to avoid the overfitting problem, we used the cross-validation with the four-folds. This means that the data was partitioned randomly into four different groups. Three data groups are randomly selected as the training dataset and the last part use as the test dataset for evaluating the model accuracy and run this process for four times.

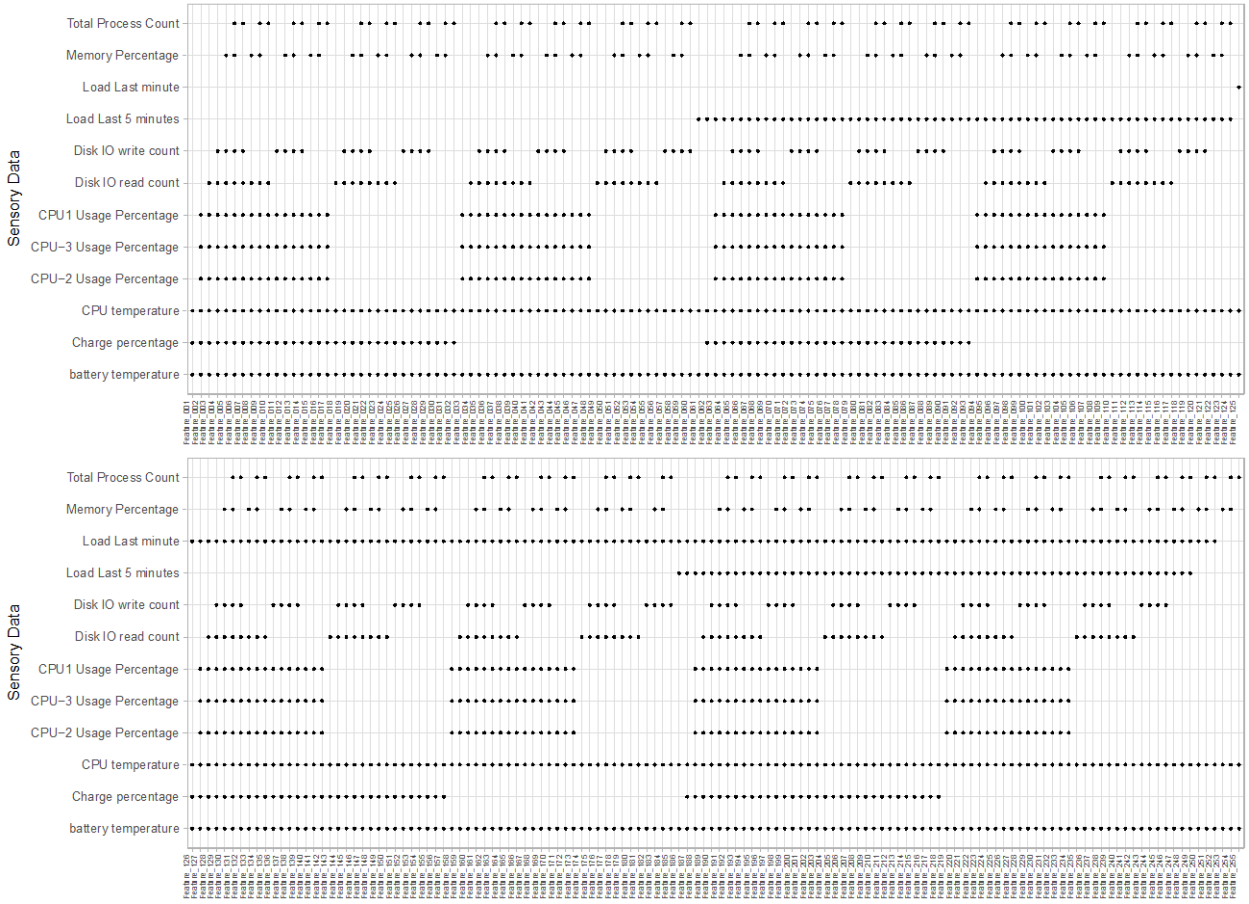

Figure 1: The different combinations of the feature list.

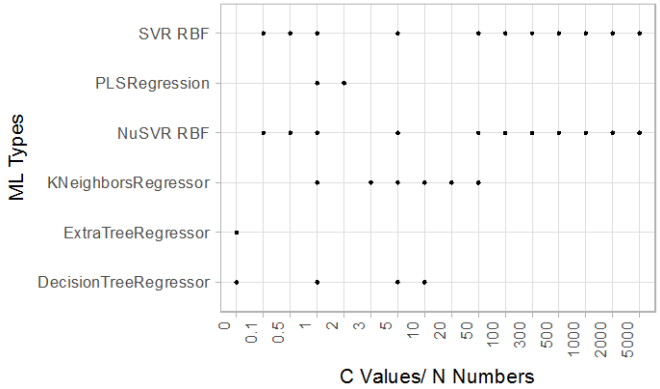

Figure 2: The different machine learning methods with their parameters used in this investigation.

Figure 3 shows the median accuracy of the ambient temperature prediction for each machine-learning algorithm. The far left of the black line represents the most accurate feature list for a given model, while the far right represents the least accurate feature list. The red dots toward the middle represent the median accuracy of a machine learning given all feature lists. In the figure, the y-axis is the machine learning type identified with its kernel C value or N numbers, and the x-axis is average prediction error for the indoor temperature in C° (the lower the mean value, the better model accuracy).

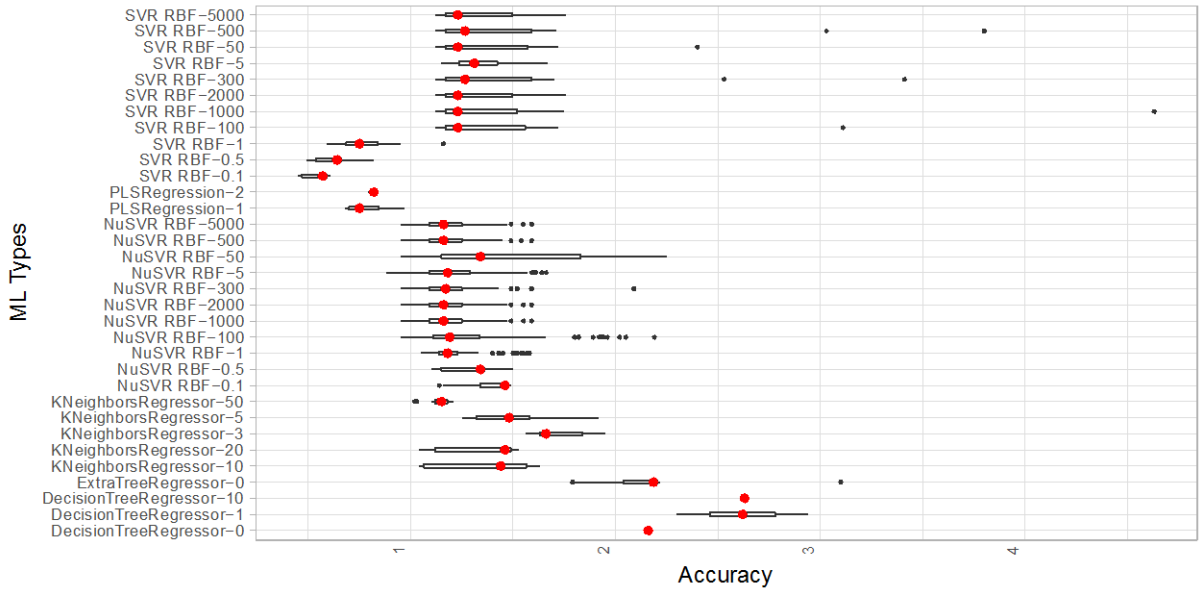

Figure 3: The median of accuracy for predicting indoor temperature using different machine learning methods.

Based on the results of Figure 3, the best ten machine learning methods have been identified and are shown in Table 1. It can be seen that PLS regressor and Super Vector Regressor (SVR) with RBF kernel are the ones with superior performances.

Table 1: Median of ML accuracy for the best features list considering all other variations

	ML Types	Median ML Accuracy
1	SVR RBF-0.1	0.57
2	SVR RBF-0.5	0.64
3	PLSRegression-1	0.75
4	SVR RBF-1	0.75
5	PLSRegression-2	0.82
6	KNeigborsRegressor-50	1.15
7	NuSVR RBF-1000	1.16
8	NuSVR RBF-2000	1.16
9	NuSVR RBF-500	1.16
10	NuSVR RBF-5000	1.16

Figure 4, shows the best 40 feature lists that were obtained by averaging the median accuracy from the best machine learning methods listed in Table 1. Figure 5 lists the feature (sensor) name for these top feature lists. Finally, Table 2 summarizes the details of the top 15 feature lists. This includes listing the name of the best machine learning model that provides the best accuracy for that feature, which surprisingly happened to be the same model for all these top feature lists. Figure 6, visually confirms the good accuracy of the best three feature lists using the best machine

learning model (the first three rows in Table 2). The figure shows a great agreement between the actual indoor temperature and the model estimated temperature.

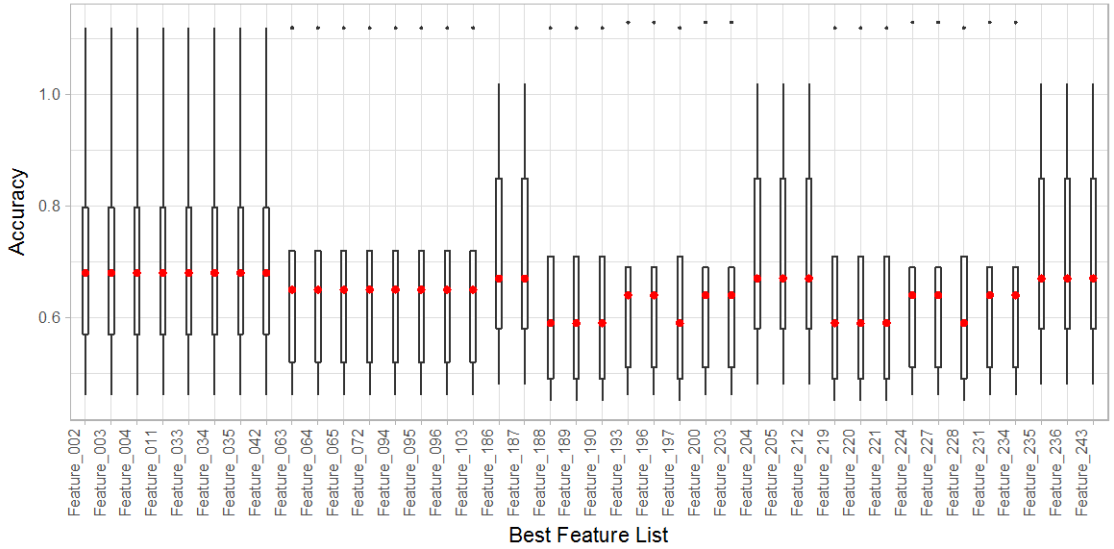

Figure 4: The top 40 feature lists obtained by averaging the median accuracy of the top machine learning methods in Table 1.

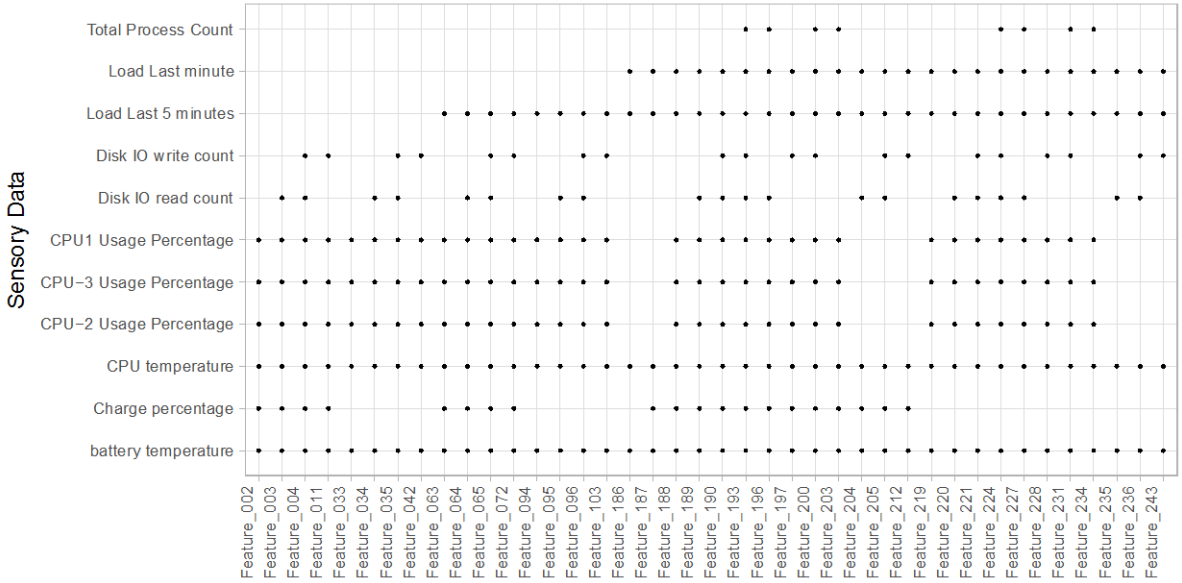

Figure 5: The actual sensor name for the top 40 feature lists.

The data in Figure 5 was used for counting the number of times a feature occurs in the top 40 feature lists. This count, shown in Table 3, helps to identify features with the highest likelihood of correlation with the actual indoor temperature. For example, one can conclude from Table 3 that the laptop load intensity of the last 5 minute and the three CPU processing usage, even though they are not temperature measurements, are among the most reliable features that can be used to predict indoor temperature.

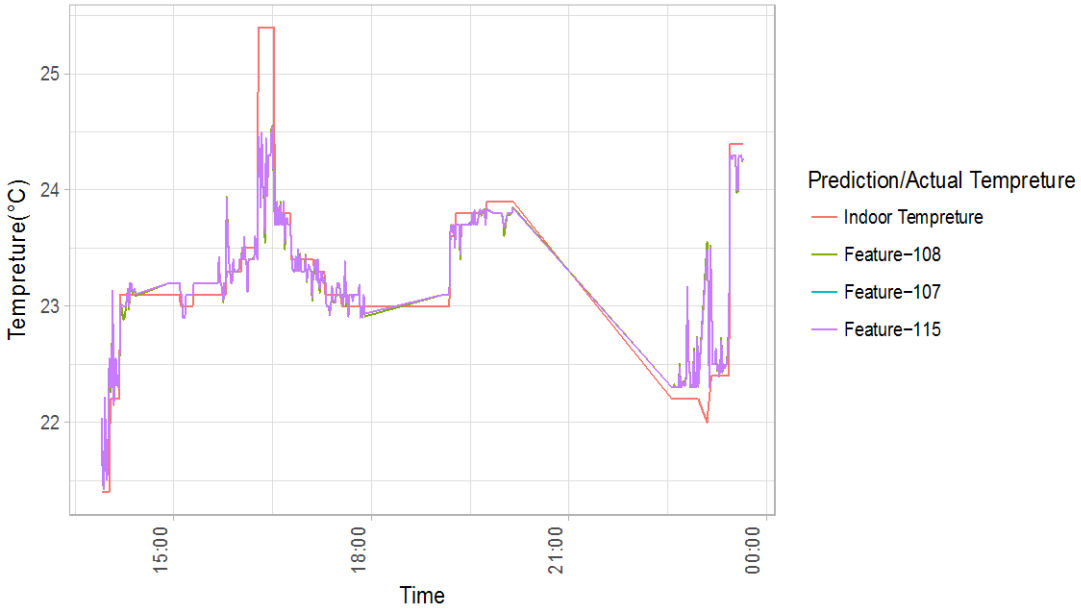

Figure 6: Model predicted Vs Actual indoor temperature

Table 2: The best machine learning method with the best feature list.

	ML Type	Features	Median ML Accuracy
1	SVR RBF-0.1	Feature_108	0.45
2	SVR RBF-0.1	Feature_107	0.45
3	SVR RBF-0.1	Feature_115	0.45
4	SVR RBF-0.1	Feature_106	0.45
5	SVR RBF-0.1	Feature_139	0.45
6	SVR RBF-0.1	Feature_138	0.45
7	SVR RBF-0.1	Feature_146	0.45
8	SVR RBF-0.1	Feature_137	0.45
9	SVR RBF-0.1	Feature_111	0.46
10	SVR RBF-0.1	Feature_114	0.46
11	SVR RBF-0.1	Feature_118	0.46
12	SVR RBF-0.1	Feature_121	0.46
13	SVR RBF-0.1	Feature_142	0.46
14	SVR RBF-0.1	Feature_145	0.46
15	SVR RBF-0.1	Feature_149	0.46

Table 3: Feature appearance frequency rank

	Feature	Appearance in Feature List
1	Bat_temp	40
2	Cpu_temp	40
3	Load_min5	32
4	Percpu_1_total	32
5	Percpu_2_total	32
6	Percpu_3_total	32
7	Load_min1	28
8	Charge_per	24
9	Diskio_disk0_read_count	20
10	Diskio_disk0_write_count	20
11	Processcount_total	8

3. CONCLUSIONS

This work has identified the best machine learning algorithms and features of MacBook Pro that could effectively model the indoor air temperature. As a result, features that are not temperature based such as CPU processing usage and the laptop load have been found as salient features in capturing and characterizing the surrounding temperature. This work can be extended for smartphones with the intention of controlling and individual's comfort in a room as opposed to the traditional holistic temperature control approach in buildings. We believe that a more reliable and accurate generalized model that works universally, across most modern electronic computing platforms, could be produced with better noise filtering approaches on raw data and internal sensors of high sensitivity.

REFERENCES

Hasan, Mohammad H., Fadi Alsaleem, and Mostafa Rafaie. 2016. "Sensitivity Study for the PMV Thermal Comfort Model and the Use of Wearable Devices Biometric Data for Metabolic Rate Estimation." *Building and Environment* 110:173–83.

Hennion, Nicolas. n.d. "Glances an Eye on Your System." https://github.com/nicolargo/glances.

Ligeza, Antoni. 1995. "Artificial Intelligence: A Modern Approach." *Neurocomputing* 9(2):215–18. Retrieved April 30, 2018 (http://thuvien.thanglong.edu.vn:8081/dspace/handle/DHTL_123456789/4010).

Naud-Dulude, Christophe. n.d. "Ruby Gem for Your Mac Stats." https://github.com/Chris911/iStats.

Overeem, A. et al. 2013. "Crowdsourcing Urban Air Temperatures from Smartphone Battery Temperatures." *Geophysical Research Letters* 40(15):4081–85. Retrieved April 30, 2018 (http://doi.wiley.com/10.1002/grl.50786).

Pedregosa, Fabian et al. 2012. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12(Oct):2825–30. Retrieved April 30, 2018 (http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html).