Interpretable detection of unstable smart TV usage from power state logs

Tamir Mazaev[1], Olivier Janssens[1], Dirk Van Gheel[2], Lieve Lanoye[2], Guillaume Crevecoeur[1], and Sofie Van Hoecke[1]

[1] Ghent University - imec, Department of Electronics and Information Systems, IDLab, Ghent, Belgium,
{tamir.mazaev@ugent.be,odjansse.janssens@ugent.be, guillaume.crevecoeur@ugent.be,sofie.vanhoecke@ugent.be}
[2] TP Vision Belgium NV, Ghent, Belgium,
{lieve.lanoye@tpv-tech.com,dirk.vangheel@tpv-tech.com}

Abstract. Power state logs from smart TVs are collected in order to construct a time-series representation of their usage. Time-series that belong to a TV exhibiting instability problems are classified accordingly. To do so, an automated feature extraction approach is used, together with linear classification methods in order to realize interpretable classification decisions. A normalized true positive rate of 0.84 ± 0.10 is obtained for the classification. The normalized true negative rate equals 0.80 ± 0.03. The final model returns a regularity statistic called the Approximate Entropy as its most important feature.

Keywords: user profiling · smart TV · time-series · feature extraction · TSFRESH · logistic regression · approximate entropy

1 Introduction

Gartner has predicted that over 25 billion "things" will be connected by 2020. In today's smart home, there are already a variety of connected devices that we interact with on a daily basis, including thermostats, home lighting, security systems, music speakers and smart TVs [4].

This rise of the Internet of Things (IoT) and Big Data enables companies to make better-informed business decisions by collecting and properly exploiting massive sets of data from every aspect of their organization.

By exchanging information over the network, collected by (virtual) sensors attached to these IoT devices, the devices become more context-aware as they aggregate knowledge on their surroundings [5]. Applying machine learning on the collected data may lead to the optimization of operational processes, or a better understanding of a company's customer base. In this paper, we focus on the analysis of smart TV usage. When a customer experiences problems with his TV, he may file a claim describing the problem and bring the device in for repair. We focus on a particular subclass of anomalous behavior called "instability"

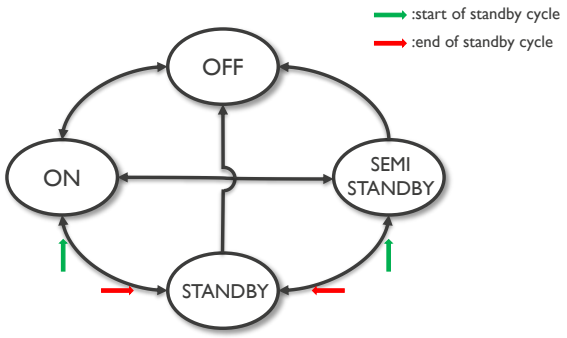


Fig. 1: Simplified power state transition diagram of a smart TV. Circles are possible states of the TV, and the black arrows indicate how the different states can be reached. Colored arrows define the concept of a "standby cycle". If a TV enters a state through a transition indicated by a green arrow, this marks the beginning of the cycle. The red arrows indicate its termination.

problems. In Section 3 we will explain in more detail what this umbrella term entails.

Overall, TPVision collects the smart TV data to detect stability issues in TVs at the customers' home. This way, we want to identify which features are linked to instability and hence need to be avoided in software or, when seen in test, which need to be fixed with highest priority. As part of this overall investigation, smart TV power cycle data are collected. These records specify when individual TVs are switching between different power collection states (see Figure 1). Obviously among devices and even among power cycles on each unique device, variations in timing of these parameters are observed, making it difficult to detect anomalies in this behavior.

We proceed in the rest of the paper as follows: Section 2 describes how we construct time-series representing the usage of a TV based on its power state logs. Section 3 describes the classification methodology applied to detect unstable devices. In Section 4, we describe the results and discuss them. Section 5 concludes this paper.

2 Constructing Time-series of Smart TV Usage

The power state logs were collected in the summer of 2017. In Figure 1, a simplified diagram is given of the logged power states of the smart TVs:

- *On* state: TV screen is on;
- *Off* state: the TV is completely inactive;
- *Standby* state: low power state where the screen is off and CPU is inactive;
- *Semi-standby* state: low power state where the screen is off but CPU is still active.

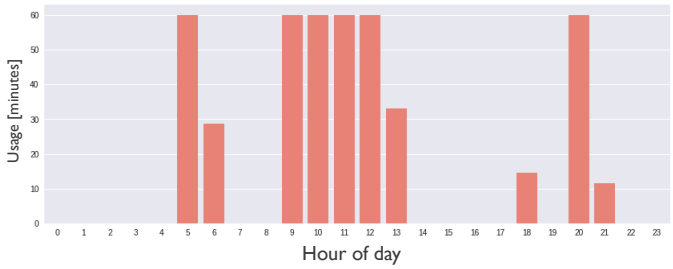
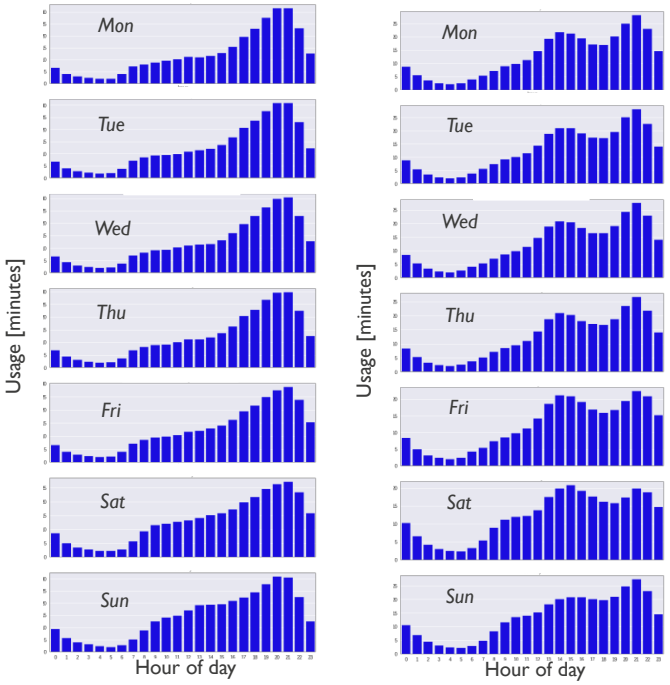


Fig. 2: Example of smart TV usage per hour on one day.



(a) The Netherlands (b) Spain

Fig. 3: Smart TV usage per hour for each day of the week. Averages for two countries are shown (N = 1000 for each country).

Different cycles can be defined on the smart TV power state transition diagram. For example, a "boot cycle" starts with a transition from the *Off* state to the *On* state, and ends with a transition to the *Off* state from any other state.

For this paper, the "standby cycle" is of interest to us because it leads to a representation of the usage of the TV. The corresponding transitions are indicated on the figure; green arrows indicate the start of the cycle, and red arrows indicate its termination. As an extra constraint, we may require that the *On* state must be reached during the standby cycle. This way, we limit the standby cycles to "user sessions" where the TV is actually being used by a human. Standby cycles that are not user sessions are, for example, power cycles where updates are being pushed to the operating system of the smart TV. For these cases, the device is never in the *On* state during the standby cycle.

The logging system of the TVs keeps a timestamp of every transition in the power cycle diagram. This way, the initiation and termination of every user session is known. The intermediate time interval then represents the device being used. Using the logs and timestamps, time-series showing the usage history of a device can be constructed. For example, in Figure 2 we visualize for how many minutes a smart TV has been used per hour throughout the day.

In Figure 3, the average usage profile per day of the week is shown for the Netherlands and Spain (N = 1000 each). As can be seen, these two countries' profiles differ clearly. One can see that there is much more activity in the early afternoon in Spain than in the Netherlands. This is just one example, more additional insights can be discovered from the data other than the identification of instability issues. Note that many other data representations could be considered starting from the smart TV log files as, for example, system states other than those describing TV power cycles are also logged. For this paper, the choice of constructing the TV usage history was made since this particular view on the data is very straightforward to interpret. We will discuss in the following sections how the time-series are used to detect unstable behavior.

3 Instability Detection

The following list describes a range of problems that may occur for a smart TV:

- (Re)boot issues;
- Operating system locks out / crashes / hangs;
- Picture disappearing.

All of these issues are called "instability" problems. Consequently, devices, brought in for repair under these claims, are labeled as being unstable.

We will now formulate the detection of instabilities in a TV as a predictive modeling problem. A schematic overview of our instability detection approach is given in Figure 4. We start by constructing a pair $(\mathbf{x}, y)$ per smart TV with $\mathbf{x} = (x_1, x_2, ..., x_n)$ the time-series associated with the usage of the TV as described above in Section 2 and y a binary target vector describing whether the device

was brought in for repair for instability problems during the workweek right after $\mathbf{x}$ (stable = 0, unstable = 1).

We always consider a period of two week starting on a Monday. Since we look at the TV usage binned per hour of the day, we always have $n = 24 \times 14 = 336$. We call $\mathbf{X}$ the set of all time-series $\mathbf{x}$.

We train a classifier to predict the target label y from its associated $\mathbf{x}$. Our total dataset consists of 91 unstable and 1000 stable devices. The set consists of German and Dutch devices combined in order to get enough unstable examples. We hold out 30% of this set as a test set, and train (and cross-validate) on the remaining samples.

After constructing the time-series $\mathbf{X}$, we apply an automated feature extraction algorithm called TSFRESH [3]. TSFRESH stands for "*Time-Series Feature extraction based on Scalable Hypothesis tests*". Its development was motivated by industrial big data applications as predictive maintenance or production line optimization, but its use also applies to our classification goal.

TSFRESH characterizes time-series by first calculating a large number of well-established feature mappings (e.g. mean, kurtosis, Fourier coefficients, ...). In total, 788 features are calculated. A feature selection step follows. Each feature vector is evaluated with respect to its dependence on the target label. If the feature is deemed significant for predicting the target, it is kept as input for the ensuing prediction algorithm. The significance of a feature is addressed by statistical hypothesis testing.

In our case the target label is binary, and the calculated features non-binary. The following hypothesis is tested by a Kolmogorov-Smirnov test:

$$H_0^\phi = \{f_{x_\phi|y=0} = f_{x_\phi|y=1}\}, \tag{1}$$

$$H_1^\phi = \{f_{x_\phi|y=0} \neq f_{x_\phi|y=1}\}. \tag{2}$$

Here x_ϕ stands for a calculated feature based on $\mathbf{x}$. $f_{x_\phi|y}$ denotes the conditional density function of x_ϕ given y. Further details on the independence tests may be found in [3].

Features that pass the previous feature selection step are Z-normalized before the next step.

Two different supervised learning methods are used for classifying the extracted features. Logistic regression is a binary linear classifier where a logistic sigmoid is applied to a linear function of the feature vector $\mathbf{x}_\phi$ confining the output between 0 and 1. Logistic regression is based on the following probability model:

$$p(y = 1 \mid \mathbf{x}_\phi) = \sigma(\mathbf{w}^T \mathbf{x}_\phi) = \frac{1}{1 + \exp(-\mathbf{w}^\mathbf{T} \mathbf{x}_\phi)} \tag{3}$$

where $\mathbf{w}$ is the weight vector that needs to be learned. Computing the classification model of linear support vector machine (SVM) classifier amounts to minimizing an expression of the form

$$\Big[\frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_{\phi,i}))\Big] + C||\mathbf{w}||^2, \tag{4}$$

where C indicates the importance assigned to maximizing the classification boundary margin between the two classes versus classifying the classes correctly, which makes it a regularization parameter [1]. We also use L1 and L2 regularization in order to improve the generalization performance of the logistic regression model. On all classifiers 3-fold stratified cross validation is performed on the training set as a hyperparameter optimization step for the regularization term. We are dealing with imbalanced classes, so the Area Under the Receiver Operating Characteristic (AUROC) curve is chosen as the validation metric. This metric is insensitive to disparities in the class proportions.

Next we apply another feature selection step. In order to further reduce the number of features, recursive feature elimination is used. First the linear estimators are trained on the original set of features and the weights of each feature are ranked according to their value. Then the least important weights are excluded from the current set of features. This process is recursively repeated on the pruned set until the optimal number of features to select is reached. Each recursive step is performed within a cross validation loop, again with the AUROC score as the validation metric. In short, we call this *recursive feature elimination with cross validation* (RFECV). For the evaluation of the final classification, we look at the normalized true positive and true negative rate.

We also apply a nonlinear classification method (SVM with Radial Basis function (RBF) kernel) on the full feature set. These SVMs are able to represent a wide range of nonlinear decision boundary classes by setting the appropriate hyperparameters [1]. We set these parameters by using the same validation procedure as mentioned above, and compare the result with the linear classifiers.

The feature selection method used in this work is only directly applicable to linear models. We note that different feature selection methods exist other than the one used in this work [2]. For example, note that for a SVM with a RBF kernel the weights $\mathbf{w}$ can not be explicitly computed, so they are not directly available for ranking. Hence another approach is required.

The class distribution of the dataset is unbalanced, so the class weights for all classifiers are balanced.

For the final classification model we focus on linear models only, since these offer the most straightforward interpretation of the relation between the feature values and their corresponding classification outputs.

4 Results

After the calculation of the 788 features by TSFRESH, only 217 features are deemed relevant by the algorithm. The normalized true positive rate (unstable class) and normalized true negative rate (stable class) of the trained classification models after applying RFECV on these remaining features are shown in Table 1. The averages and standard deviations for both classes are shown for ten runs with a different train-test set splitting seed. Before the application of RFECV, the linear classifiers perform better on the positive class and worse on the negative class than the SVM with a RBF kernel. After the application

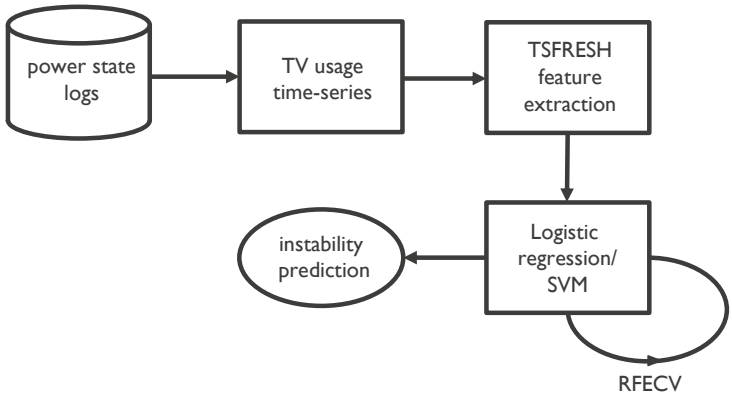


Fig. 4: Schematic representation of instability detection approach

Table 1: Test set result of the trained classification models. The normalized true positive rate (unstable class) and normalized true negative rate are shown (stable class). Ten runs with a different train-test split seed where performed.

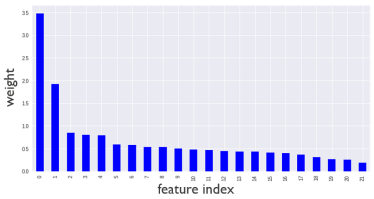
(a) Before RFECV

	L1 LR	L2 LR	SVM LIN	SVM RBF
unstable	0.72 ± 0.1	0.74 ± 0.06	0.66 ± 0.08	0.60 ± 0.09
stable	0.79 ± 0.02	0.73 ± 0.02	0.79 ± 0.02	0.83 ± 0.02

(b) After RFECV

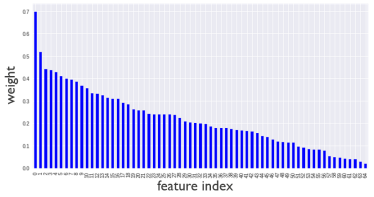
	L1 LR	L2 LR	SVM LIN
unstable	0.84 ± 0.10	0.83 ± 0.07	0.79 ± 0.08
stable	0.80 ± 0.03	0.80 ± 0.02	0.80 ± 0.03

weight	TSFRESH feature name
3.48	approximate_entropy__m_2__r_0.1
1.92	change_quantiles__f_agg_"var"__isabs_False
-0.84	fft_coefficient__coeff_70__attr_"real"
-0.79	fft_coefficient__coeff_28__attr_"real"
0.79	fft_coefficient__coeff_24__attr_"abs"
0.58	fft_coefficient__coeff_61__attr_"abs"
0.57	fft_coefficient__coeff_74__attr_"abs"
0.53	fft_coefficient__coeff_29__attr_"abs"
0.53	agg_linear_trend__f_agg_"mean"__chunk_len_5
0.49	sum_of_reoccuring_data_points



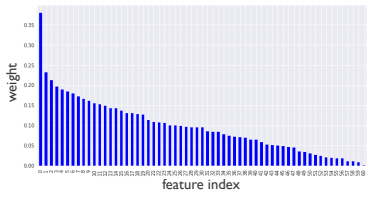
(a) For L1 logistic regression

weight	TSFRESH feature name
0.70	approximate_entropy__m_2__r_0.1
-0.51	value_count__value_0
0.44	fft_coefficient__coeff_24__attr_"abs"
-0.44	fft_coefficient__coeff_28__attr_"real"
0.43	fft_coefficient__coeff_54__attr_"abs"
0.40	fft_coefficient__coeff_38__attr_"abs"
-0.39	number_cwt_peaks__n_1
0.39	range_count__max_1__min_-1
0.38	sum_of_reoccuring_data_points
0.37	agg_linear_trend__f_agg_"max"__chunk_len_50



(b) For L2 logistic regression

weight	TSFRESH feature name
0.38	approximate_entropy__m_2__r_0.1
-0.23	fft_coefficient__coeff_28__attr_"real"
0.21	fft_coefficient__coeff_52__attr_"abs"
0.20	sum_of_reoccuring_data_points
0.19	fft_coefficient__coeff_54__attr_"abs"
-0.18	fft_coefficient__coeff_70__attr_"real"
-0.18	mean_second_derivative_central
0.17	fft_coefficient__coeff_24__attr_"abs"
0.16	fft_coefficient__coeff_38__attr_"abs"
-0.16	agg_autocorrelation__f_agg_"var"



(c) For SVM with linear kernel

Fig. 5: Features with top ten largest weights of the linear classification models. The ranked feature importances are also plotted for all features remaining after applying RFECV.

of RFECV, the linear classifiers perform very similar to each other. The feature selection procedure also improves the performance of each classifier considerably.

In Figure 5, we show the top ten most important features for both regularization schemes. Feature names from the TSFRESH package are used (a full description of all extracted features can be found in the documentation of the package). A positive weight here means that a higher value for the corresponding feature indicates that the device belongs to the stable class. A negative weight means the opposite. The value of the learned weights is also plotted for all features that remain in the classification model. For L1 logistic regression, only 22 features are retained in the final model. For L2 regularization, 71 features remain. For the SVM, 65 features remain. As can be seen from the logistic regression models, a L1 regularization term typically leads to a solution that is more sparse in the number of features [1]. As a result this enhances the interpretability of the final model. We remark that it also gives the best classification performance.

We now focus on two interesting observations concerning the shown features. For all models, the so called Approximate Entropy is the most important feature. This statistic has been developed in order to quantify the amount of regularity and unpredictability in time-series data. It has an important use in the analysis of physiological time-series. Its exact mathematical description can be found in [6]. In this classification task, we try to predict instability in smart TV usage. Hence, it is interesting to observe that a direct measure of unstable behavior emerges as the most discriminative aspect of the time-series with this goal in mind. Another interesting feature to note is the real part of the 28^{th} Fourier coefficient, which appears in all models with a negative weight. Since the length n of the time-series is equal to 336, this corresponds to periodicity on the scale of 12 hours. So if the TV usage pattern shows strong periodicity on the scale of 12 hours, this indicates a higher probability of the device being stable.

5 Conclusion

We have presented an approach to construct a time-series representation of the usage of a smart TV based on its power state logs, visualizing for how many minutes a smart TV has been used per hour throughout the day.

Instability issues in the TVs can be detected using an automated feature extraction procedure of the time-series called TSFRESH. Applying linear classification models using those features leads to predictive models that can be interpreted based on the weights assigned to their features. A feature called the Approximate Entropy is shown to be the most important feature, as it has the highest weight for all classification models. The best classification result is obtained by using L1 logistic regression. It enables the detection of TVs with instability issues with a normalized true positive rate of 0.84 ± 0.10. The normalized true negative rate equals 0.80 ± 0.03.

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Secaucus, NJ, USA (2006)
2. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Computers & Electrical Engineering **40**(1), 16 – 28 (2014). https://doi.org/10.1016/j.compeleceng.2013.11.024
3. Christ, M., Kempa-Liehr, A.W., Feindt, M.: Distributed and parallel time series feature extraction for industrial big data applications. arXiv preprint **1610.07717** (May 2017)
4. Günther, W.A., Mehrizi, M.H.R., Huysman, M., Feldberg, F.: Debating big data: A literature review on realizing value from big data. The Journal of Strategic Information Systems **26**(3), 191–209 (September 2017). https://doi.org/10.1016/j.jsis.2017.07.003
5. Ng, I.C., Wakenshaw, S.Y.: The internet-of-things: Review and research directions. International Journal of Research in Marketing **34**(1), 3–21 (March 2017). https://doi.org/10.1016/j.ijresmar.2016.11.003
6. Pincus, S.M., Gladstone, I.M., Ehrenkranz, R.A.: A regularity statistic for medical data analysis. Journal of Clinical Monitoring **7**(4), 335–345 (October 1991). https://doi.org/10.1007/BF01619355