Extracting Discriminative Shapelets from Heterogeneous Sensor Data

Om P. Patri[1], Abhishek B. Sharma[2], Haifeng Chen[2], Guofei Jiang[2], Anand V. Panangadan[1], Viktor K. Prasanna[1]
[1]University of Southern California, Los Angeles, CA 90089
{patri, anandvp, prasanna}@usc.edu
[2]NEC Laboratories America, Princeton, NJ 08540
{absharma, haifeng, gfj}@nec-labs.com

***Abstract*—We study the problem of identifying discriminative features in Big Data arising from heterogeneous sensors. We highlight the heterogeneity in sensor data from engineering applications and the challenges involved in automatically extracting only the most interesting features from large datasets. We formulate this problem as that of classification of multivariate time series and design shapelet-based algorithms for this task. We design a novel approach, called *Shapelet Forests* (SF), which combines shapelet extraction with feature selection. We evaluate our proposed method with other approaches for mining shapelets from multivariate time series using data from real-world engineering applications. Quantitative analysis of the experiments shows that SF performs better than the baseline approaches and achieves high classification accuracy. In addition, the method enables identification of noisy sensors from multivariate data and discounts their use for classification.**

***Index Terms*—Multivariate Data, Time Series Shapelets, Shapelet Forests, Feature Selection, mRMR**

I. Introduction

Big Data arising from real-world engineering applications are often collected from a large network of process monitoring sensors. Analysis of such large-scale sensor data can enable identification of patterns that are indicative of abnormal processes and suggest appropriate corrective actions. Analysis for these purposes thus focuses on identifying *discriminative features*, i.e., those patterns that are most relevant for distinguishing between normal and abnormal processes.

In this work, we study the case where the Big Data is from heterogeneous sensor sources. We formulate this problem as that of multivariate time series classification. While multivariate time series classification is a well-studied problem [5], [12], we focus on this problem in the context of complex real-world engineering applications such as manufacturing plants and automobiles. For example, in the case of a manufacturing plant, a common scenario involves a process being continuously repeated to produce batches of goods. During the production of each batch, monitoring data in the form of time series is collected by multiple sensors. Quality control tests can be performed after a batch of goods is produced to determine whether they are normal, i.e., meet a set of pre-determined standards, or abnormal. We describe an application from a manufacturing plant in Section II that fits this description. Identifying discriminative features in this application could help operators determine the *root cause(s)* of abnormal operation by identifying only the most relevant portions of the Big Data for this purpose.

Many forms of multimodal sensor data recorded in engineering and data science applications can be processed into a time series format. Thus, multimodal data received from heterogeneous sensors can be processed to multivariate time series data, with data from each sensor corresponding to one dimension of the multivariate time series. For instance, shapes of objects [25], electricity consumption data [18], hand gestures and signals [11], [16], medical ECG signals [11], [21], gyroscope and accelerometer data from physical activity and motion capture [11], [21], [25], as well as images, handwritten characters and audio samples (e.g., bird sounds) [25] can be easily converted into time series data. In this work, we specifically focus on such *multivariate time series* data. Our developed algorithm for multivariate time series classification thus has the potential to be used in a wide array of multimodal data.

Challenges. Solving the multivariate time series classification problem for complex physical systems involves two important challenges. Firstly, labels are associated with instances not the time series. For a normal instance, it is safe to assume that the multivariate time series in $\mathbf{T_i}$ is normal, but for an abnormal instance, often only a subset of the time series in $\mathbf{T_i}$ is abnormal. Similar scenarios arise in other domains [6], [7], [11], e.g., medical informatics where only a subset of clinical time series data (such as heart rate) collected from a patient might be capturing the disease symptoms.

The second challenge arises from the volume and variety of time series data. We present examples from real-world case studies in Section II. Physical systems such as manufacturing plants are operated 24x7 and monitored by a large number of sensors that can collect data at a high rate, e.g., one sample every tens of milliseconds to a second.Classification needs to be computationally fast even on large datasets. Training data from real-world use cases are imbalanced, i.e., the numbers of normal and abnormal instances are not similar. The time series data collected by these sensors exhibit different patterns–e.g. some exhibit variation with values over a large range whereas others are stable with values concentrated within a short range. Moreover, the labels for training may be available only at the end of the multiple processes making assignment of labels

978-1-4799-5666-1/14/$31.00 ©2014 IEEE

to specific data instances impossible. The training algorithm also has to accommodate for the possibility of redundant sensors and correlated data streams from such sensors. Another source of data variety is due to the fact that the n time series belonging to $\mathbf{T_i}$ may not have the same number of samples due to different sampling rates across sensors. Hence, when comparing time series from the same or different instances, we cannot assume that they have the same length.

Multivariate time series classification. The data consists of multiple labeled instances $\mathcal{I} = \{(\mathbf{T_i}, \mathbf{y_i})\}$ where y_i is a label for instance i and $\mathbf{T_i} = [\mathbf{T_i^1}, \ldots, \mathbf{T_i^n}]$ consists of n time series. Thus, $\mathbf{T_i}$ is n-dimensional and typically each of its time series represents the monitoring data collected periodically by a sensor. Each time series T_i^j is a sequence of real numbers $t_1, t_2, \ldots, t_m$. Given K labeled instances from $\mathcal{I}$, we want to extract *features* from multivariate time series $\mathbf{T_i}$ that are *discriminative*. Discriminative features can then accurately distinguish between different labels or classes of instances and can focus the attention of domain experts on those situations likely to provide the most insight.

Shapelets for multivariate time series classification. In this work, we explore the efficacy of *shapelets* [25] for solving our multivariate time series classification problem for complex physical systems. A shapelet is a subsequence or local temporal pattern in a time series that is a representative feature of the class to which this time series belongs [25]. Shapelets have been shown to be effective for a variety of time series data mining tasks including classification, clustering, summarization, and visualization [6], [7], [16], [18], [21], [25]. Though several other approaches exist for time series classification, the popularity of shapelets is due to four factors [6], [7], [21], [25]: (1) shapelet methods impose no assumptions or restrictions on nature of data unlike autoregressive or ARIMA time series models [5], (2) they are easy to *interpret* for domain experts [6], [7], [18], (3) they have been shown to be more accurate than other methods for some problems [6], [21], and once shapelets have been extracted from training data, we can classify a new time series quickly. These factors influenced our choice of using shapelets for the multivariate time series classification problem. Most of the work on shapelet mining is aimed at *univariate* time series data. We propose a new approach called *Shapelet Forests* which is able to find shapelets from *multivariate* time series.

Our contributions. Our work makes three significant contributions. First, we use two real-world case studies to motivate the problem of multivariate time series classification for complex physical systems. In doing this, we highlight the main characteristics of time series data collected from these systems and the challenges involved in training a classifier using them. Second, we propose a novel classification approach, called *Shapelet Forests*, and two other baseline approaches for solving our classification problem. Finally, we evaluate Shapelet Forests using real-world datasets from physical systems and compare its performance against several other approaches. Additionally, we use a publicly available dataset from a physical system to generate "synthetic" scenarios to further "stress test" these algorithms. This includes handling imbalanced and noisy/redundant data.

The rest of this paper is organized as follows. Section II illustrates two real-world case studies that motivate our work. Section III provides a brief background on algorithms for extracting shapelets from univariate time series, and Section IV describes different approaches for extracting multivariate shapelets. We present evaluation results in Section V. Section VI discusses related work and we conclude in Section VII.

II. Motivation: Engineering applications

In this section, we describe two real-world engineering applications that motivate our work. We point out the key characteristics of multivariate time series data from these cases and discuss the challenges involved in training a classifier using this data.

A. Monitoring automobile performance

This application involves monitoring data collected from an automobile under different driving conditions. 15 different sensors are used to periodically collect measurements on different aspects of the automobile's engine during each test run. We have access to data from 25 test runs. Thus, we have 25 instances with multivariate time series data from 15 sensors. Each instance is labelled as "normal" or "abnormal" by a domain expert. Our task is to design a classifier that can accurately predict whether an unseen instance is normal or abnormal based on data from 15 sensors.

Data properties. The number of samples varies across time series even for time series from the same instance. The 375 time series in the dataset (25 instances with 15 time series each, 11 normal instances and 14 abnormal instances) contain between 800 to 3000 samples each.

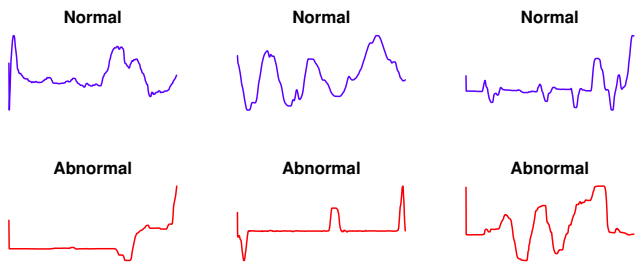

Fig. 1. Variety in time series instances within the same sensor - normal instances (top) and abnormal instances (bottom)

Figure 1 shows examples of time series from one of the 15 sensors for normal and abnormal instances. We can clearly see the challenge arising from data "variety" – even for instances from the same class and the same sensor, there are no clear global patterns, and the time series do not follow any predefined distribution or structural assumptions.

B. Identifying abnormal manufacturing processes

Our second application is a manufacturing process that consists of a pipeline of 5 processes shown in Figure 2. In

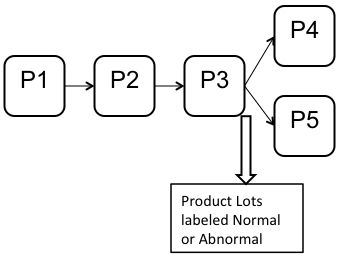

Fig. 2. The pipeline of manufacturing processes in our case study

this case, a product is manufactured in batches with each batch processed sequentially by $P1$, $P2$, and $P3$, and after $P3$ finishes, it is processed by either $P4$ or $P5$. When a process P_i is processing a batch, multiple sensors collect monitoring data. The 5 processes are monitored by 5 to 8 sensors each. In this work, we focus on processes $P1$, $P2$, and $P3$ because the domain expert labeled a batch as normal or abnormal at the end of $P3$ (before it is processed by $P4$ or $P5$).

We define a multivariate time series instance as follows. When a batch j is processed by P_i, we associate an instance $I_{ij} = (\mathbf{T_{ij}}, \mathbf{y_{ij}})$ with it that consists of all the sensor measurements collected during batch j's processing by P_i. Thus, for each batch j, we will have three instances I_{1j}, I_{2j}, and I_{3j}, one for each process $P1$, $P2$, and $P3$. Assigning labels y_{ij} to these instances raises an interesting challenge. If a batch j is marked as normal at the end of $P3$, then naturally we set $y_{ij} = 0$, i.e. normal, for $i = 1, 2, 3$. However, if it is marked abnormal, then there are several possibilities: all three processes have a problem or only one or two of them have a problem. It is not possible to disambiguate between these because we have the label only at the end of $P3$. In this work, we choose to set $y_{1j} = y_{2j} = y_{3j}$, i.e. if a batch is marked abnormal (normal) at the end of $P3$, then all the three instances associated with it are marked abnormal (normal). Despite this uncertainty with abnormal labels, we show that our shapelet-based classifier is able to achieve high classification in Section V. Next, we provide examples of the shapelets extracted for this dataset that demonstrate their discriminative ability.

Data and shapelet properties. Each of the three processes $P1$, $P2$, and $P3$ are monitored by a different number of sensors - 8 for $P1$, 5 for $P2$, and 6 for $P3$. Each time series contains between 100 and 1500 data points and we have 10 instances for each process. Figure 3 (top) shows a shapelet extracted from the time series data for sensor 2 of $P3$. The entire time series has 180 samples and the shapelet's length is 26 samples. The middle and bottom curves in Figure 3 show two time series – middle curve is from the same class as the shapelet and the bottom curve is from a different class – along with their distances from the shapelet found. We can see that the middle curve has a subsequence of length 26 that is similar to the shapelet and hence, its distance from the shapelet is much smaller compared to the bottom curve which does not have any subsequence of length 26 that matches the shapelet closely. This example demonstrates how shapelets can be discriminative. Figure 4 shows another such example. The shapelet is extracted from time series from sensor 1 of $P3$ labeled as class 1. Two other instances of Class 1 contain a subsequence from sensor 1 that closely matches the shapelet but even the closest matching subsequences for instances from Class 2 have a very different shape. We describe our approach for finding the shapelet shown in Figure 3 in Section IV-C and in Section V we show that our approach assigns high importance to it for differentiating between the normal and abnormal instances from $P3$.

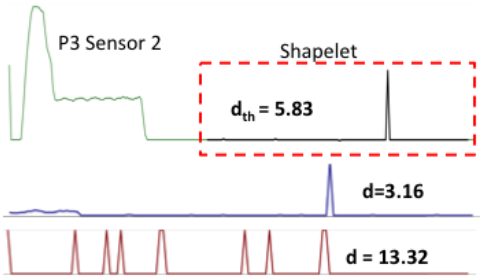

Fig. 3. Instances from P3 sensor 2, showing distances between the closest similar and dissimilar time series

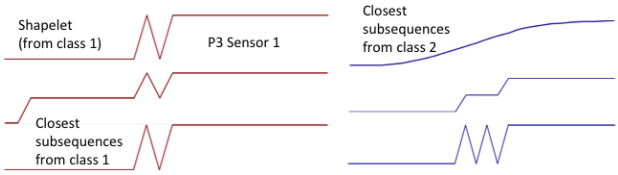

Fig. 4. Data from P3 sensor 1

Another advantage of using shapelets for our applications is their ability to pick out subtle changes in shape or pattern that might not be distinguishable to a human. Figure 5 shows a shapelet at the top and the closest matching subsequences to it from different instances. While all the subsequences appear to have similar pattern, we can see from the distance numbers that the shapelet mining algorithm can pick a distance threshold $d_{th} = 1.13$ (we present the details on how this is done in the next section) that can be used to distinguish between instances from Class 1 and Class 2.

III. Background on Shapelets

In this section we provide a brief overview of the shapelet extraction algorithm [25] for univariate time series and a computationally efficient version of it, called *Fast Shapelets* [21]. We use the Fast Shapelet algorithm as a sub-routine for multivariate shapelet mining (see the Shapelet Forests algorithm described in Section IV-C). An informed reader can skip to the next section.

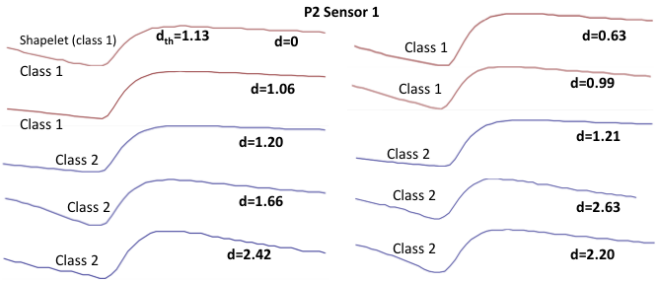

Fig. 5. Data from P2 sensor 1. The distance threshold separates instances of the different classes even though the overall pattern is similar for all instances

The shapelet extraction algorithm for univariate time series data proceeds as follows. After reading the input time series and labels, candidate subsequences of all possible lengths are generated. The distance between the candidate subsequence (tentative shapelet) and all the time series are calculated and this distance value is used to place the time series on an *orderline* [25], denoting the proximity of the candidate to the time series. A *discriminative subsequence* separates the time series instances from different classes by projecting them on the orderline in such a way that instances for the same class are close to each other and far away from instances belonging to other classes. Hence, given an orderline, a threshold or separation gap is defined that captures how far apart instances from different classes are based on their distance from the candidate subsequence, and used to compute the information gain for the candidate subsequence. The user is required to specify the *minL* and *maxL* parameters that denote the minimum and maximum length of shapelets.

A decision tree classifier is built during the shapelet extraction phase with shapelets as nodes. The simplest case is a decision tree with a single node corresponding to a shapelet that can cleanly split instances from two different classes. However, the decision tree may contain multiple nodes or shapelets, particularly in case of real-world datasets described in Section II. When a new time series instance is encountered, this decision tree is used to classify the instance and predict a label. At each node of the tree, the distance to the corresponding shapelet is compared with the threshold for that shapelet, and either the left or right subtree is chosen for traversal, until a leaf node is reached. Each leaf node contains the class label to which it belongs, and this is the predicted class label for this instance.

The brute-force shapelet discovery algorithm is slow because it computes all possible subsequence-information gain combinations. Several optimizations and pruning techniques to improve upon the brute-force method have been suggested [16], [21], [25]. The state-of-the-art shapelet-based classification algorithm is Fast Shapelets [21] which uses random projections to get approximate, nearly optimal solutions. We build upon the Fast Shapelets algorithm, referred to as `FAST-SHAPELETS(`$\mathcal{I}$`)`, in this work as a subroutine to extract shapelets from multivariate time series.

IV. Multivariate shapelets

There are two broad strategies for generalizing univariate time series shapelets to a multivariate setting: (1) convert the multivariate data to univariate data, and then apply the univariate shapelet mining algorithm, and (2) modify the shapelet mining algorithm discussed in the previous section to work directly with multivariate data. In this section, we propose three methods, including our *Shapelet Forests* algorithm to deal with multivariate shapelets. We also review existing related approaches.

A. Naive Shapelets (NS)

We can assign the instance label y_i to each of the n time series in $\mathbf{T_i}$. This gives us n univariate time series with label y_i. Suppose there are two classes, i.e. $y_i \in \{0, 1\}$ with k_0 instances of Class 0 and k_1 instances of Class 1 where each instance has n time series. Using this approach, we end up with $k_0 \times n$ univariate instances of class 0 and $k_1 \times n$ univariate instances of class 1. We can then use algorithms such as Fast Shapelets [21] to extract shapelets.

To classify a new multivariate instance $\mathbf{T_i}$, NS uses the decision tree learned by the Fast Shapelets algorithm (as in case of classifying univariate time series). For a multivariate instance with n time series, it first uses the decision tree to learn the label for each univariate time series. It then computes the final label, e.g. normal or abnormal, by majority voting, i.e., the final label is the same as the label for the majority of the univariate time series.

This approach ignores any "correlations" between time series from the same instance, and we refer to it as *Naive Shapelets (NS)* (analogous to the Naive Bayes approach). This approach implicitly assumes that **all** the n time series are useful for classification. Our evaluation results in Section V show that it does not perform well when only a subset of the n time series are best suited for distinguishing between different classes. Hu et al. make the same observation [11].

B. Concatenated Shapelets (CS)

Another simple approach for converting the multivariate time series from an instance to a single time series is to concatenate them. A few works have explored this approach including Logical-Shapelets [16]. We refer to this approach as *Concatenated Shapelets (CS)*.

A major drawback of this approach is high computational complexity. If the concatenated time series has a large number of samples (as is the case for our automobile and manufacturing datasets), the training phase can take a long time due to the fact that the running time of shapelet extraction is quadratic in the number of samples in a time series. Additionally, as in Naive Shapelets, this approach may not work well when only a subset of the n time series capture the differences between the normal and abnormal classes.

C. Shapelet Forests (SF)

The Naive and Concatenated Shapelet algorithms convert multivariate time series data to an equivalent univariate representation. The alternative is to design new shapelet mining

algorithms that can work directly with multivariate data. In this paper, we propose such a new approach called *Shapelets Forest (SF)* that works directly with multivariate data and is equivalent to NS for univariate instances.

SF training phase. For extracting shapelets, i.e. in its training phase, SF takes the three steps shown in Figure 6. (We state the SF algorithm more formally later in this section.) In the first step, it uses a univariate shapelet mining algorithm separately for each of the n time series. E.g., given k_0 instances of Class 0 and k_1 instances of Class 1 where each instance has n time series, SF first performs n different univariate shapelet extractions, where for time series i, $i = 1, \ldots, n$, we have k_0 univariate time series labeled as Class 0 and k_1 labeled as Class 1. This step can give more than n shapelets. The number of shapelets extracted from each of the n dimensions depends on the data. Intuitively, time series that capture the differences between classes are more likely to provide one or more shapelets compared to time series that are same or similar across classes.

The shapelets extracted in the first step by SF represent the *features* of multivariate time series data, and typically a subset of these are *discriminative*, i.e. help us classify a new instance as normal or abnormal. The second and third steps in SF, labeled as *Compute Data/Distance Matrix* and *Learn Feature Weights* in Figure 6, accomplish the goal of identifying the best subset of shapelets for differentiating between normal and abnormal instances. The *Compute Data/Distance Matrix* step is needed to convert the multivariate time series data into the format accepted by typical feature selection algorithms that SF uses to learn feature weights. Our key contribution here is to do this using the shapelets extracted in the first step of SF.

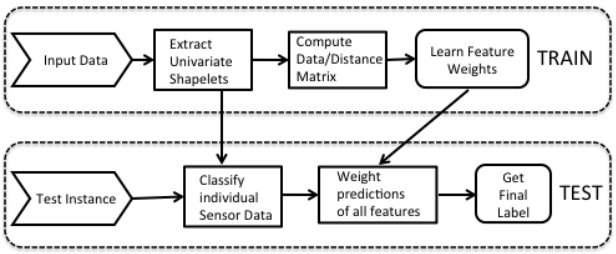

Fig. 6. Shapelet Forests Approach

Distance Matrix. In typical feature selection approaches [2], [19], [27], a data matrix ($\mathcal{D} = \{d_{ij}\}$) is provided as input where instances are rows and features are columns of this matrix. However, in our data, each column or feature is a time series dimension or sensor, and hence, d_{ij} is not a scalar but a sequence of numerical values. We propose an intuitive transformation of multivariate time series data into the data matrix format that is required by state-of-the-art feature selection algorithms. Instead of using a sensor or time series dimension, we use the extracted shapelets as *features* for the columns of our data matrix and the instances make up the rows. Each entry d_{ij} is filled with the distance of time series instance to the corresponding shapelet(s) originating from the same sensor. E.g., suppose we want to compute the entry $d_{3,2}$. This corresponds to the third instance and the second shapelet extracted from the data. We find out which sensor the second shapelet was from. Say it was from sensor 1; then the distance is computed between the second shapelet and the univariate time series corresponding to sensor 1 in the third instance, i.e. distance($\mathbf{T_3^1}, s_2$) where s_2 is the second shapelet. Since the shapelets represent the columns, it is possible to have more than n columns in the data matrix. The size of the data matrix is equal to the number of instances times the total number of shapelets.

Learning weights for shapelets. After transforming time series data into a matrix using shapelets, we can use state-of-the-art feature selection algorithms to learn the weights for each shapelet. These weights are then used for *weighted voting* to classify a new instance as shown in Figure 6. In this paper, we explore three feature selection methods. Two of these methods, MCFS [2] and SPEC [27], are based on spectral clustering and require a similarity matrix along with the data matrix. We used common similarity matrices such as the dot product matrix and Gaussian kernels.

mRMR [19]: uses the *minimum Redundancy Maximum Relevance* criterion that ranks features based on how closely related they are to the class labels. At the same time, the set of high ranked features should not be redundant; loosely, not correlated to each other. We use the mRMR implementation provided by the authors of [19] and use the Mutual Information Quotient (MIQ) indicator as the weight of a feature.

MCFS [2]: stands for Multi-Cluster Feature Selection, and it can be used for both supervised and unsupervised feature selection. In the supervised MCFS approach, the dot product kernel or Gaussian kernel or 0-1 kernel are commonly used for the similarity matrix [2]. In the unsupervised approach, k-nearest neighbors are used to construct the similarity matrix. We use $k = 5$. It is a two-step process based on spectral clustering [23]. First, it uses the k smallest eigenvalues (and their eigenvectors) of the graph Laplacian to embed the high dimensional data into a low dimensional space. Then, L1-regularized least squares optimization is used to find the best features.

SPEC [27]: stands for SPECtral Feature Selection and this algorithm provides a unified approach for both supervised and unsupervised feature selection using spectral graph theory. For the unsupervised case, the linear dot product kernel and RBF kernel are popular choices for the similarity matrix.

The SF algorithm. Algorithms 1 and 2 present the pseudo-code for the the training and classification phases of our SF algorithm. `FEATURE-WEIGHT()` represents any of the three feature selection algorithms used to assign weights to shapelet features. Since there are $z \geq n$ shapelets in total, we have w_z weights.

A new instance is classified by first obtaining the predictions from the n univariate classifiers. Then, we compute a weighted combination of these decisions (e.g., weighted majority voting)

to compute the final label. The weights for n classifiers are computed from the weights of the shapelets provided by the feature selection algorithms. The `REDUCE()` method represents any procedure for computing the weights for n classifiers using the weights of $z \geq n$ shapelets. A simple method is to sum up or average the weights for all shapelets from a certain sensor when a specific sensor provides multiple shapelets. We use a more sophisticated approach that adjusts the weights of the classifiers based on the test instance being classified. As described in Section III, we use shapelet-based decision trees to classify univariate time series. When classifying time series j from a multivariate instance using its corresponding decision tree classifier, we track the "path" through the decision tree to remember which shapelets were responsible for the label. We use only the weights of these shapelets instead of all the shapelets in the decision tree to set the weight of the classifier (as average of the weights of shapelets). This gives us an effective data reduction approach as it is specific to each test instance and can capture its classification path to provide better results.

Algorithm 1 Shapelet Forests Training

procedure SHAPELET-FORESTS-TRAIN($\mathcal{I}$)
  $\mathcal{I} = \{(\mathbf{T_i}, \mathbf{y_i})\}, \mathbf{T_i} = [\mathbf{T_i^1}, \ldots, \mathbf{T_i^n}]$
  for `f = 1 to n` **do** ▷ n features
    $\mathcal{I}_f = [(\mathbf{T_1^f}, y_1), \ldots, (\mathbf{T_p^f}, y_p)]$ ▷ p instances
    $[S_f, DT_f] \leftarrow$ `FAST-SHAPELETS(`$\mathcal{I}^f$`)`
  end for
  $z \leftarrow \Sigma|S_f|$
  $\mathcal{D}$ `is a matrix (p x z) such that` ▷ z shapelets
  $d_{i,j} \leftarrow$ `distance(`$\mathbf{T_i^g}, \mathbf{s_j}$`)`
  `where g is the dimension which produced` s_j
  $[w_1, \ldots, w_z] \leftarrow$ `FEATURE-WEIGHT(`$\mathcal{D}, y_i$`)`
end procedure

Algorithm 2 Shapelet Forests Testing

procedure SHAPELET-FORESTS-TEST($\mathcal{J}$)
  $\mathcal{J} = \{(\mathbf{T_j})\}, \mathbf{T_j} = [\mathbf{T_j^1}, \ldots, \mathbf{T_j^n}]$
  for `f = 1 to n` **do** ▷ n features
    for `j = 1 to q` **do**
      $c_j^f \leftarrow$ `CLASSIFY(`$\mathbf{T_j^f}, DT_f$`)`
    end for
  end for
  for `f = 1 to z` **do** ▷ z shapelets
    for `j = 1 to q` **do** ▷ q instances
      $w_f^j \leftarrow$ `REDUCE(`w_f, S_f`)` ▷ Classifier weights
    end for
  end for
  `pred(j)` $= \Sigma_f w_f^j c_j^f$ ▷ Predicted Label
end procedure

D. Other approaches

We discuss related existing approaches in detail to highlight their similarities and differences with SF.

Dictionary-based nearest neighbor classification. Hu et al. [11] present an approach for classifying multivariate streaming time series data that shares certain similarities with SF. We refer to this approach as adC named after the key metric, called *adjusted Confidence score*, used for classification in [11]. As in SF, adC first builds a separate classifier for each of the n univariate time series. It uses the *Nearest Neighbor (NN)* classifier for time series data [10]. E.g., given k_0 instances of Class 0 and k_1 instances of Class 1 where each instance has n time series, adC will first create n sets where set j consists of time series j from all the $k = k_0 + k_1$ instances. Thus, each set has k_0 time series labeled as Class 0 and k_1 time series labeled as Class 1. We can associate a *Nearest Neighbor* classifier with each of these n sets. The classifier for set j predicts Class 0 (Class 1) for a univariate time series T if the nearest neighbor of T in set j is from Class 0 (Class 1).

To classify a multivariate instance with n time series, adC performs a weighted combination of n classifiers. adC assigns a weight to each classifier by combining its precision on the training data (computed using leave-one-out classification) and the nearest neighbor distance of the new instance from the training data associated with a classifier using Bayes' rule. This weight is called *adjusted confidence score* because it is derived by adjusting the confidence score on training data based on the data in the new instance. Hu et al. [11] show that *adjusted confidence score* performs better than other schemes like majority vote or using the precision on training data as weights in an online streaming setting where a new instance might not belong to any of the classes in the training data.

adC design is similar to SF. Both approaches build a classifier for univariate time series and then combine the votes from multiple classifiers for classification. However, they use different algorithms. adC uses *Nearest Neighbor* whereas SF uses *Shapelet-based Decision Tree*. It is known that *Shapelet-based Decision Tree* is an *eager* version of the *lazy Nearest Neighbor* classifier [25]. In the second step, adC uses weights derived from leave-one-out classification on training data, plus adjustment based on the instance being classified, while SF uses weights derived from a feature selection algorithm. Feature selection algorithms like mRMR account for the correlation or dependencies between different features (shapelets) while computing the weights; in contrast, adC processes the different time series in multivariate instances independently to assign weights to classifiers.

To reduce the classification time, adC uses dictionaries for each of n sets of univariate time series. Intuitively, a dictionary D for set j consists of the subsequences of univariate time series in j that capture the key patterns (e.g. frequent patterns, discriminative patterns, etc.) present in j. adC uses the framework proposed in [11] to learn dictionaries. Typically, dictionaries are significantly smaller in size than

the full training data and using them for nearest neighbor calculations can lead to huge improvements in classification time. adC's dictionaries are analogous to the shapelets in SF. An interesting future work can be to compare the subsequences extracted as shapelets by SF against the subsequences in the dictionaries of adC.

Optimization-based approaches In [6], Ghalwash et al. propose the *MSD* (Multivariate Shapelets Detection) approach. According to MSD, a multivariate shapelet is a collection of subsequences, one from each of the n time series in an instance, where each subsequence has the same start time and length. After imposing these restrictions, Ghalwash et al. adapt the univariate shapelet mining algorithm to the multivariate setting by proposing a multivariate information gain based threshold that replaces its univariate counterpart. This multivariate threshold is based on a vector of distances between a multivariate shapelet and the n dimensional $\mathbf{T_i}$ with its component j equal to the (Euclidean) distance d_j between corresponding dimensions j of the multivariate shapelet and $\mathbf{T_i}$. There are two major weaknesses of this approach. First, the restriction of same start time and same length is arbitrary. Since we do not want FS to impose this restriction (subsequences in FS can start at different time points and can be of different lengths), we do not use MSD for evaluation. Second, just like the NS and CS algorithms, MSD may not work well when a subset of the n time series are a better predictor for a class than all of them. The authors of MSD provide an example of such a scenario from the biomedical domain [6].

In [7], Ghalwash et al. proposed *IPED* (Interpretable Patterns for Early Diagnosis), which improves upon MSD [6]. IPED constructs a binary matrix where the rows represent instances and the columns represent all possible subsequences of the time series data. For a dataset with k instances where each instance has n time series of length L, this binary matrix will have $n \times k \times \sum_{l=minL}^{maxL} l(L-l+1)$ columns ($minL$ and $maxL$ denote the minimum and maximum length of the subsequences). Then, IPED solves a convex-concave optimization problem to select exactly one representative subsequence from each of the n dimensions. After this dimensionality reduction step, we have a multivariate shapelet with n subsequences. IPED then selects a subset of these n subsequences using mixed integer programming. Hence, the final output of IPED is similar to SF – a subset of subsequences or shapelets extracted from n dimensional multivariate time series data. While IPED addresses some shortcomings of MSD, it still suffers from two drawbacks: (1) the exhaustive search over all possible subsequences in the first step is computationally expensive, and (2) the restriction of extracting only one subsequence per dimension in the second step is somewhat arbitrary. The largest dataset used in [7] has 15 time series per instance with only 3200 samples per time series. For large real-world datasets such as ours, the first step of IPED is a computational bottleneck and hence, we do not use this approach in our evaluations.

V. Evaluation

The extended version of all evaluation results presented in this section, as well as additional results, which could not fit here due to space limits, can be found in **the supporting webpage [1] for this paper**.

We first show that our SF algorithm outperforms the baseline approaches of NS and CS using data from the two real-world engineering applications discussed in Section II. We also show that using state-of-the-art feature selection algorithms in the second step of SF is more accurate than simple heuristics like majority voting. Finally, we use another real-world multivariate time series dataset, Wafer, provided by Olszewski [17] to explore the impact of imbalanced labels, i.e., more instances from the normal class than the abnormal class, and noisy/redundant sensors on SF's performance. We z-normalize each time series to have a zero mean and one standard deviation in all experiments unless otherwise specified.

A. Automobile data

This dataset consists of 25 multivariate instances, each having 15 time series, the time series consisting of 800 to 3000 data points. We ran three experiments with different number of training instances. *Split 1* has 7 training and 18 test instances, *Split 2* has 18 training and 7 test instances, and *Split 3* has 12 training and 13 test instances. We chose these splits to study the impact of the amount of training data on SF and other multivariate shapelet algorithms. These splits of the dataset were constructed randomly and the results reported are averaged over multiple runs. Table I shows that SF outperforms NS and CS regardless of whether we use majority voting or frequency and mRMR-based feature selection in the second step.

TABLE I
Automobile: Classification Accuracies

Experiment/ Approaches	Split 1 (7/18)	Split 2 (18/7)	Split 3 (12/13)
NS	0.44	0.48	0.41
CS	0.72	0.86	0.7
SF w/ majority vote	0.89	0.86	0.92
SF w/ freq.	0.89	0.86	0.77
SF w/ mRMR	1.0	0.86	0.92

B. Manufacturing data

The dataset from the manufacturing plant is more complex than the automobile dataset. Recall that a batch goes through three process in sequence $P1 \rightarrow P2 \rightarrow P3$ and a domain expert labels it as normal or abnormal at the end of $P3$. In our experiments, we make a conservative choice and mark all three instances associated with a batch as abnormal if it is marked abnormal at the end of $P3$.

We first demonstrate that feature selection can provide large accuracy gains on this dataset using a brute force approach. We then compare performances of the different feature selection algorithms that can be used in the second step of SF against the

"ground truth" obtained using brute force search. Our results show that these algorithms are competitive.

TABLE II
MANUFACTURING DATASET: LEAVE-ONE-OUT ACCURACY VERSUS NUMBER OF DIMENSIONS USED. NOTE THAT USING THE *right* SUBSET OF TWO DIMENSIONS CAN ACHIEVE 100% ACCURACY IN ALL PROCESSES.

size	P1	P2	P3	Best subsets for P3
1	0.8	1.0	0.9	{2}
2	**1.0**	**1.0**	**1.0**	{1,2}, {2,6}
3	0.8	0.8	0.8	{1,2,5},{2,3,5},{2,4,5}
4	1.0	0.75	0.83	{1,2,5,6}
5	0.9	0.6	0.8	{1,2,3,4,5}
6	1.0	–	0.71	{1,2,3,4,5,6}
7	0.7	–	–	—
8	0.6	–	–	—

The brute force approach for identifying the subset of dimensions that achieve the best classification accuracy is to evaluate all subsets. Instances from $P1$ have 8 time series and hence, the subsets of these sensors can have 8 different cardinalities (the total number of subsets is $255 = 2^8 - 1$). Similarly, $P2$ and $P3$ instances have 5 and 6 time series and hence, their subsets have cardinality of at most 5 and 6. We can see that for three processes there exist at least one subset of dimensions that achieves 100% accuracy but using time series from all the dimensions performs the worst.

Table II presents the results from brute force search for all three processes in more detail. Using the time series from just dimension 2 is sufficient for achieving 100% classification accuracy in P2. There exists a combination of two sensors that can provide 100% classification accuracy for all three processes As we include data from more dimensions, the performance degrades significantly, and using time series data from all the dimensions achieves only 60% accuracy for P1 and P2 and 71% for P3.

Table III presents the leave-one-out (LOO) classification accuracy achieved by various methods on the three processes without doing any feature selection. These methods include the brute force search, the *Frequency* heuristic based on it for identifying the most important dimensions, Naive Shapelets (NS), Concatenated Shapelets (CS), and SF with majority voting.

Instead of the brute force approach which has exponential computational complexity, SF can choose from a suite of feature selection algorithms. The feature selection algorithms assign weights to different features (shapelets in our case). We can interpret these weights as an indication of the discriminative power of these shapelets. We can sum up the weights of shapelets from each time series dimension to determine the weight for each dimension, and use them to rank the time series dimensions. Such a ranking can aid system operators in determining the cause of an anomaly.

TABLE III
LOO ACCURACY WITHOUT FEATURE SELECTION FOR ALL PROCESSES

Algorithm	P1	P2	P3
brute-force	1.0	1.0	1.0
Frequency	0.8	1.0	0.9
NS	0.61	0.62	0.61
CS	0.3	0.1	0.3
SF w/ majority	0.5	0.8	0.7

Table IV compares the accuracy and weights for the 6 time series dimensions for $P3$ from mRMR, MCFS and SPEC against the frequency heuristic based on brute force search. For completeness, SF with majority voting approach is also included. mRMR assigns weights that are qualitatively similar to the weights from Frequency: dimension 2 is top ranked followed by dimension 6. The MCFS algorithm also identifies dimension 2 as the top-ranked. These results demonstrate that we can use SF, with an automatic feature selection algorithm, to extract discriminative features that achieve better classification accuracy than simple heuristics like majority voting, and avoid searching through all possible subsets of dimensions without sacrificing classification accuracy. Table IV presents the results for P3. We explore the properties of weights assigned by mRMR to different time series dimensions in more detail in the next experiment.

TABLE IV
LEAVE-ONE-OUT ACCURACY AND WEIGHTS AFTER FEATURE SELECTION FOR PROCESS P3 (TOP RANKED CLASSIFIER DIMENSION SHOWN IN BOLD)

Method	Classifier weights						Acc.
	c_1	c_2	c_3	c_4	c_5	c_6	
Frequency	5	**9**	3	3	6	3	**0.8**
SF w/ maj.	1	1	1	1	1	1	**0.7**
MCFS	1	**6**	4	3	5	2	**0.8**
SPEC	**0.45**	0.30	0.15	0.33	0.27	0.04	**0.7**
mRMR	0.22	**0.97**	0.22	0.22	0.29	0.24	**0.8**

C. Wafer dataset

We evaluated our proposed approaches on the Wafer dataset [17], which is publicly available[1]. The data is from a process used for manufacturing silicon wafers for semiconductor chips. It consists of 1194 multivariate instances with 6 time series in each instance. It is highly imbalanced - only 127 of the 1194 instances belong to the anomaly class, the rest being normal. This enables the study of how SF performs when there are more normal instances than abnormal ones. Such cases arise in physical systems like manufacturing plants or automobiles because, by design, they do not experience faults or anomalies often. We use this dataset to also study the impact of noise and redundant data on SF's performance.

Effect of Imbalanced Data on Accuracy. The class imbalance, defined as the ratio of anomalous instances to normal instances, for the full Wafer dataset is $1 : 8.4$. We created datasets with different imbalances – $1 : 1$ and $1 : 4$ – by uniform random sampling from the full dataset. For each of these four sets, we ran three different experiments with different amount of training data – 25%, 50%, and 75% of the data. We maintain the class imbalance when creating the training and test sets. Table V shows the accuracy of NS and SF with majority vote, i.e. no feature selection, for the different experiments. Additional results are available on the supporting webpage [1]. There are two take-aways from

[1]At http://www.cs.cmu.edu/~bobski/data/

these experiments. First, the performance of NS and SF is robust to class imbalance and amount of training data for all cases. Second, in all 9 experiments, SF outperforms NS; the performance gap is significant for balanced data ($1:1$ imbalance) as well as imbalanced data ($1:8.4$).

Imbalance	Train/Test Split (%)	NS	SF w/ majority
1:1	25/75	0.89	**0.96**
	50/50	0.85	**0.91**
	75/25	0.94	**0.97**
1:4	25/75	0.94	**0.96**
	50/50	0.95	**0.98**
	75/25	**0.99**	**0.99**
1:8.4 (original dataset)	25/75	0.95	**0.97**
	50/50	0.92	**0.98**
	75/25	0.92	**0.98**

TABLE V
EFFECT OF CLASS IMBALANCE AND AMOUNT OF TRAINING DATA

Feature Selection and Redundant Sensors. We now consider the issue of some of the sensors being redundant, a common scenario in the real world. Specifically, we want to test whether our proposed algorithm, which aggregates the prediction result of multiple sensors by weighting them, is able to pick out the noisy/redundant sensor(s) by assigning a low weight to shapelet features from them. Out of the six sensors, we *modify the data from sensor* 1 by replacing the time series from sensor 1 in each anomalous instance by sensor 1 time series from a normal instance. We did not modify the time series data from rest of the sensors. The normal instances whose sensor 1 time series is used for replacing the sensor 1 data in abnormal instances are not used again in our experiments.

We use a 75/25 percent split for train-test sets. The task is to check whether the shapelet features from sensor 1 have lower weights than the shapelet features from other sensors. After the feature extraction step of SF, we have 20 shapelets in total, four of which are from sensor 1. In the second step of SF, we evaluate three feature selection algorithms, mRMR, SPEC and MCFS. Table VI shows the weights assigned by these three algorithms for the shapelets. The results shown for MCFS are for the supervised version and for SPEC, it is the unsupervised version with a Gaussian kernel of width 0.1. These were the best parameters from all combinations we tried. We can observe that the mRMR weights for all four shapelets from sensor 1 are very low compared to other sensors. Thus, mRMR can automatically identify the redundant sensor time series data. Though the accuracy of SF with all three methods are comparable, the major advantage of using SF with mRMR method is that it automatically eliminates noisy and redundant data. Hence, the shapelet features identified by SF with mRMR are potentially more useful for identifying root cause(s) of the anomaly.

TABLE VI
WEIGHTS OF VARIOUS FEATURE SELECTION METHODS. S.ID IS SENSOR ID AND SH.ID IS SHAPELET ID. SENSOR 1 DATA IS REDUNDANT. NOTE THAT MRMR ASSIGNS LOWER WEIGHTS TO SENSOR 1 FEATURES, AUTOMATICALLY IDENTIFYING THE NOISY SENSOR.

s.id	sh.id	mRMR	MCFS	SPEC
1	1	**0.019**	7	0.0009
	2	**0.034**	3	0.0053
	3	**0.04**	12	0
	4	**0.038**	8	0.0054
2	5	0.287	14	0.005
3	6	0.227	8	0.2115
4	7	0.315	20	0
	8	0.237	18	0
	9	0.173	15	0.0002
	10	0.261	5	0.0092
	11	0.149	3	0.0001
5	12	0.105	16	0.0003
	13	0.295	18	0.0003
	14	0.139	4	0.0001
6	15	0.189	13	0.0004
	16	0.109	17	0.0094
	17	0.158	9	0.0022
	18	0	19	0
	19	0.053	10	0.0337
	20	0.229	2	0.0016

VI. RELATED WORK

We described approaches closely related to our work ([6], [11] and [7]) in Section IV. Here, we briefly review other related works.

The common shapelet extraction approach, proposed by Ye and Keogh [25] uses an information gain criteria to determine the shapelet. An unsupervised algorithm to find shapelets using a separation gap measure instead of information gain has also been proposed in [26]. Alternate measures for quality of shapelets, such as Kruskal-Wallis and Mood's median tests, have been explored in [13]. Lines et al. [14] propose to disconnect the shapelet extraction approach from the classification step through a transformation step which uses the distance from the shapelet as input data for classification. This is similar to our weight learning approach using the distance from the shapelet to build the data matrix for feature selection. Deng et al. [4] propose to build a time series forest for classification using both entropy gain and distance metrics (referred to as the *entrance* metric). However, they do not use shapelets for classification. Spiegel et al. [22] adopt an alternative approach for finding recurring patterns (not shapelets) in multivariate data. They segment the multivariate time series using singular value decomposition (SVD), and use hierarchical agglomerative clustering on the resulting segments. The quality, length, and number of their discovered patterns is dependent on the SVD parameters, and subject to the computational complexity of SVD. Also, since finding the critical inflection points in their algorithm requires the data to be smooth (second and third derivatives are computed), it places an assumption on the nature of the data which shapelet-based methods do not.

For certain domains like medical diagnosis, in addition to classification accuracy and interpretability, early detection is an important goal [6], [7]. The concept of *local shapelets*, proposed by Xing et al. [24], is also related to early classification. He et al. [9] provide early work on early prediction for imbalanced multivariate time series. Prieto et al. [20] propose stacking individual univariate classifiers to build an ensemble classifier for multivariate time series. None of these

approaches work directly with multivariate data from complex physical systems and utilize feature selection methods to combine their constituent univariate classifiers.

Executing shapelet-based approaches can be slow, especially when using long time series instances, since the complexity is quadratic in the length of time series (m) but linear in the number of time series (n), i.e. $O(nm^2)$. Therefore, efforts have focused on efficient pruning techniques including Logical Shapelets [16] and Fast Shapelets methods [21]. Another way to reduce complexity is to restrict the set of shapelet candidates being evaluated based on certain constraints - such an approach is presented in McGovern et al. [15] to build shapelet tries for weather prediction. Gordon et al. [8] propose to optimize the evaluation order of shapelet candidates using a randomized model for shapelet sampling. Chang et al. [3] propose to use GPUs for fast, parallel implementation of shapelet mining approaches.

VII. Conclusion

We analyzed the problem of multivariate time series classification that arises in Big Data applications. Using two real-world engineering applications, we highlighted the technical challenges arising from data characteristics as well as the need to extract discriminative features from Big Data. We described a new multivariate shapelet mining algorithm called Shapelet Forests (SF). It combines the state-of-the-art shapelet-based algorithm for univariate time series [21] to build an ensemble of classifiers, one for each time series dimension in our multivariate instances. To perform a weighted combination of these classifiers, we used feature selection algorithms such as mRMR [19] that can account for correlations between shapelets extracted from different time series dimensions. Our approach adds a new tool to the existing literature that learns classifiers' weights using their "accuracy on the training data" [11]. We evaluated SF using three real-world datasets and also compared it against baseline approaches for mining multivariate shapelets. Our evaluation shows that SF performs better than simple baseline approaches and achieves high accuracy. It is also able to automatically identify the noisy sensors in the multivariate data and assign them lower weights for prediction. As part of our future work, we plan to investigate how to account for dependencies between different time series dimensions when extracting shapelets instead of SF's current approach of processing each dimension separately.

References

[1] Supporting webpage http://www-scf.usc.edu/~patri/bigdata14.html.

[2] D. Cai, C. Zhang, and X. He. Unsupervised feature selection for multi-cluster data. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 333–342, New York, NY, USA, 2010. ACM.

[3] K.-W. Chang, B. Deka, W.-M. W. Hwu, and D. Roth. Efficient pattern-based time series classification on gpu. In *Proceedings of the 12th IEEE International Conference on Data Mining*, 2012.

[4] H. Deng, G. Runger, E. Tuv, and M. Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 2013.

[5] T.-c. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[6] M. Ghalwash and Z. Obradovic. Early classification of multivariate temporal observations by extraction of interpretable shapelets. *BMC bioinformatics*, 13(1):195, 2012.

[7] M. F. Ghalwash, V. Radosavljevic, and Z. Obradovic. Extraction of interpretable multivariate patterns for early diagnostics. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 201–210. IEEE, 2013.

[8] D. Gordon, D. Hendler, and L. Rokach. Fast randomized model generation for shapelet-based time series classification. *arXiv preprint arXiv:1209.5038*, 2012.

[9] G. He, Y. Duan, T. Qian, and X. Chen. Early prediction on imbalanced multivariate time series. In *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management*, CIKM '13, pages 1889–1892, New York, NY, USA, 2013. ACM.

[10] B. Hu, Y. Chen, and E. Keogh. Time series classification under more realistic assumptions. In *SDM*, 2013.

[11] B. Hu, Y. Chen, J. Zakaria, L. Ulanova, and E. Keogh. Classification of multi-dimensional streaming time series by weighting each classifier's track record. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 281–290, Dec 2013.

[12] M. W. Kadous. *Temporal classification: Extending the classification paradigm to multivariate time series*. PhD thesis, The University of New South Wales, 2002.

[13] J. Lines and A. Bagnall. Alternative quality measures for time series shapelets. *Intelligent Data Engineering and Automated Learning*, 7435:475–483, 2012.

[14] J. Lines, L. M. Davis, J. Hills, and A. Bagnall. A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 289–297. ACM, 2012.

[15] A. McGovern, D. Rosendahl, R. Brown, and K. Droegemeier. Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction. *Data Mining and Knowledge Discovery*, 22(1):232–258, 2011.

[16] A. Mueen, E. Keogh, and N. Young. Logical shapelets: an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1154–1162. ACM, 2011.

[17] R. T. Olszewski. *Generalized Feature Extraction for Structural Pattern Recognition in Time-series Data*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2001. AAI3040489.

[18] O. P. Patri, A. Panangadan, C. Chelmis, and V. K. Prasanna. Extracting discriminative features for event-based electricity disaggregation. In *Proceedings of the 2nd Annual IEEE Conference on Technologies for Sustainability*. IEEE, 2014.

[19] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238, 2005.

[20] O. J. Prieto, C. J. Alonso-González, and J. J. Rodríguez. Stacking for multivariate time series classification. *Pattern Analysis and Applications*, pages 1–16, 2013.

[21] T. Rakthanmanon and E. Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. *Proceedings of the 13th SIAM International Conference on Data Mining*, 2013.

[22] S. Spiegel, J. Gaebler, A. Lommatzsch, E. De Luca, and S. Albayrak. Pattern recognition and classification for multivariate time series. In *Proceedings of the fifth international workshop on knowledge discovery from sensor data*, pages 34–42. ACM, 2011.

[23] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[24] Z. Xing, J. Pei, S. Y. Philip, and K. Wang. Extracting interpretable features for early classification on time series. In *SDM*, pages 247–258, 2011.

[25] L. Ye and E. Keogh. Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data Mining and Knowledge Discovery*, 22(1-2):149–182, 2011.

[26] J. Zakaria, A. Mueen, and E. Keogh. Clustering time series using unsupervised shapelets. In *Proceedings of the 12th IEEE International Conference on Data Mining*, 2012.

[27] Z. Zhao and H. Liu. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 1151–1157, New York, NY, USA, 2007. ACM.