A methodology for Energy Multivariate Time Series Forecasting in Smart Buildings based on Feature Selection

Aurora González[a], Fernando Jiménez[a], Antonio F. Gómez-Skarmeta[a]

[a]*Department of Information and Communication Engineering. Faculty of Informatics. University of Murcia, 30100, Murcia, Spain*

Abstract

The massive collection of data via emerging technologies like the Internet of Things (IoT) requires finding optimal ways to reduce the created features that have a potential impact on the information that can be extracted through the machine learning process. The mining of knowledge related to a concept is done on the basis of the features of data. The process of finding the best combination of features is called feature selection. In this paper we deal with multivariate time-dependent series of data points for energy forecasting in smart buildings. We propose a methodology to transform the time-dependent database into a form that standard machine learning algorithms can process, and then, apply different types of feature selection methods for regression tasks. We used *Weka* for the tasks of database transformation, feature selection, regression, statistical test and forecasting. The proposed methodology improves *MAE* by 59.97% and *RMSE* by 40.75%, evaluated on training data, and it improves *MAE* by 42.28% and *RMSE* by 36.62%, evaluated on test data, on average for 1-step-ahead, 2-step-ahead and 3-step-ahead when compared to not applying any feature selection methodology.

Keywords: Feature Selection, Energy Efficiency, Time Series, Smart Buildings, Smart Cities.

1. Introduction

Energy efficiency is the goal to optimise the amount of energy required to provide products and services. Energy consumption is increasing with the growing population and intensified in highly populated parts of cities [1]. Energy efficiency is in the interest of everyone, from individuals to governments, since it yields economical savings, reduces greenhouse gas emissions and alleviates energy poverty [2]. In order to achieve energy efficiency, smart grids, open data platforms and networked transport systems are proliferating for managing and monitoring resources automatically. This provides the emergence of smart cities, which thanks to the collection of data using sensors that are interconnected through the internet (*Internet of Things*) allow the extraction of insights that are necessary in order to provide better services to the citizens that also include energy efficiency.

The huge amounts of data that are collected via the IoT are consequently analysed in order to extract the knowledge necessary for achieving energy efficiency. However, in order to realise such analysis it is desirable to reduce the dimensionality of the data for easing the models performance. In order to do so there exist several approaches such as *segmentation* and *representation of attributes* [3] or *feature selection* [4]. We are going to focus on feature selection since it has shown its effectiveness in many applications by building simpler and more comprehensive models, improving learning performance, and preparing clean, understandable data.

In this work, we use *time series data* from the Chemistry Faculty of the University of Murcia to generate energy consumption *forecasts* [5, 6]. *Time series forecasting* is the process of using a model to generate forecasts for future events based on known past events. Time series data has a natural temporal ordering. This differs from typical machine learning applications where each data point is an independent example of the concept to be learned, and the ordering of data points within a data set does not matter. For this reason, standard machine learning methods should not be used directly to analyze time series data. In this paper, we propose a methodology to, firstly, transform the time series into a form that standard machine learning algorithms can process, and then, systematically apply a set of feature selection methods for regression that includes *univariate*, *multivariate*, *filter* and *wrapper* methods [7]. Time series data is transformed by removing the temporal ordering of individual input examples and adding a set of delays to the input which are called *lagged attributes* and provide the temporal information. The methodology also allows dealing with *intervention attributes*, which are to be considered external to the data transformation and closed-loop forecasting processes. This approach to time series forecasting is more powerful and more flexible than classical statistics techniques such as *ARMA* and *ARIMA* [8]. Feature selection methods are applied for the selection of both lagged and intervention attributes. *Random Forest*, *instance-based learning* and *linear regression* algorithms are used for regression with the different reduced databases. Finally, the best reduced database together with the best regression algorithm are used for the predictions *1-step-ahead*, *2-step-ahead* and *3-step-ahead* evaluated in training data and test data, and the results are compared with the predictions obtained with the original database. The experiments have been carried out using the *Waikato Environment for Knowledge Analysis* (*Weka*) [9].

With this background the paper has been organized as follows: section 2 defines the concept of feature selection and their categorization, shows the feature selection mechanisms in the *Weka* machine learning software and describes the data set used for experiments; section 3 proposes a methodology for the energy efficiency analysis in smart buildings based on feature selection; section 4 analyzes and discusses the results, and finally section 6 concludes the paper.

2. Background

2.1. Feature Selection

Feature Selection (FS) is defined in [4] as the process of eliminating features from the database that are irrelevant to the task to be performed. FS facilitates data understanding, reduces the measurement and storage requirements, the computational process time, and the size of a data set, so that model learning becomes an easier process. An FS method is basically a *search strategy* where the performance of candidate subsets is measured with a given *evaluator*. The search space for candidate subsets has cardinality $O(2^w)$, where w is the number of features. A *stopping criterion* establishes when the FS process must finish. It can be defined as a control procedure that ensures that no further addition or deletion of features produces a better subset, or it can be as simple as a counter of iterations. FS methods are typically categorized into *wrapper*, *filter* and *embedded*, *univariate* and *multivariate* methods. *Wrapper methods* [10] use a predetermined learning algorithm to determine the quality of selected features according to an evaluation metric [11]. *Filter methods* apply statistical measures to evaluate the set of attributes [12, 13, 14]. *Embedded methods* achieve model fitting and FS simultaneously [15]. *Multivariate methods* evaluate features in batches. *Univariate methods* evaluate each feature independently. Figure 1 illustrates graphically the FS flow. Figure 2 shows general schemes for multivariate and univariate FS.

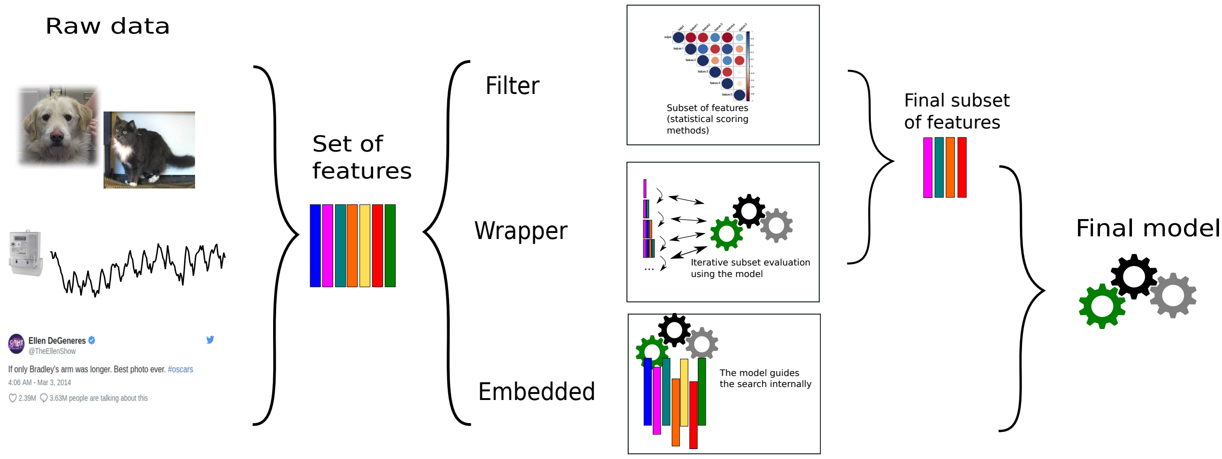

Figure 1: The feature selection flow.

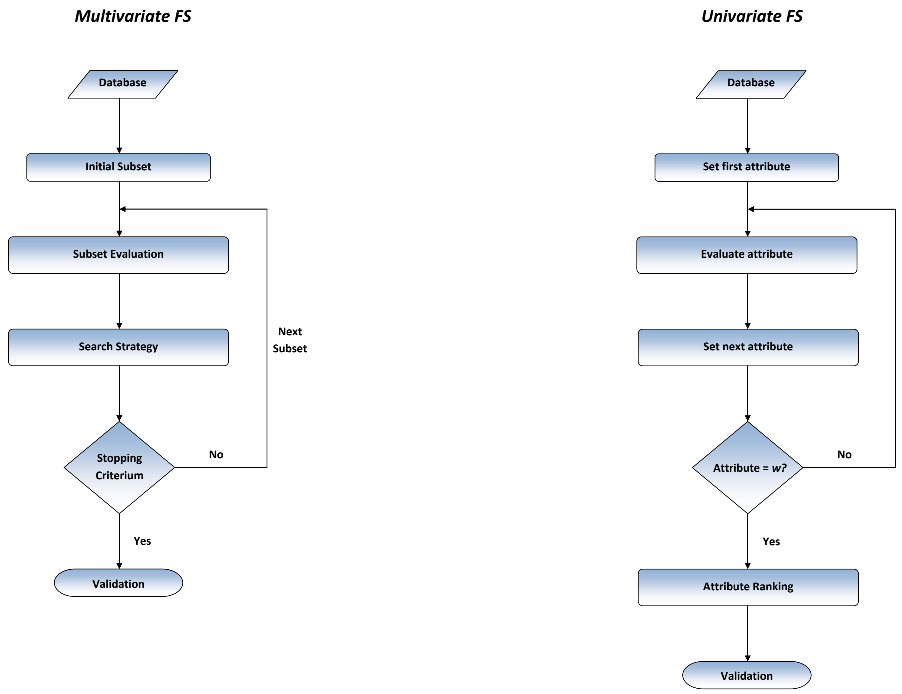

Figure 2: General schemes for multivariate and univariate feature selection.

2.2. Related Work

We have carried out an extensive search in order to find other academic works that have solved a similar problem than ours. Together with the works that address FS for energy consumption time series, we have also considered important to review FS for energy consumption when not treated as time series, and FS for time series problems in general, i.e. other approaches not specifically related to energy consumption.

The first paper that studied how the selection of subsets of features associated with building energy behaviours influences a machine learning model performance for energy consumption prediction used some filter methods for FS and support vector regression for forecasting [16]. A bit later, in the thesis [17], *Fast Correlation-Based Filter* (*FCBF*) is used for FS in load prediction error problems in four building areas. A meteorological dataset from several locations and also, the geographical factor are exploited by selecting variables from different locations. The baseline comparisons are done with *e-SVR*. According to this work, how the relationships between features change with distance motivates a greedy FS method for the electrical load forecasting. In the work [18], *correlation* and *principal components analysis* (*PCA*) are used for FS and transformation.

Feature selection for time series prediction has been carried out using neural networks[19]. By combining contemporaneous and lagged realisations of the independent variables and lagged dependent variables more general models of dynamic regression, autoregressive (*AR*) transfer functions and intervention models are constructed. Other studies have searched for the optimal time-windows and time lags for each variable based on feature pre-processing and sparse learning in order to configure the input dataset [20].

In other works, the forecasting of solar radiation time series in enhanced by using a train set of bootstrapped Support Vector Machines in order to perform FS [21]. They assure that this method is more robust than a regular FS approach because using the later, small changes on the train set may produce a huge difference on the selected attributes. Other studies related to solar radiation prediction mask the inputs as a FS step [22]. They create their own features by defining night, sunrise, day and sunset according to the moment that their instruments perceive those. This provides certain improvements on forecast accuracy. A data-driven multi-model wind prediction methodology using a two-layer ensemble machine learning technique is developed in [23]. A deep FS framework is employed where four different approaches are used in order to get the input vector: *PCA*, *Granger Causality Test*, *Autocorrelation and Partial Autocorrelation Analysis*, and *Recursive Feature Elimination*. Another ensembles way of selecting features in presented in [24] and it is used for predicting the amount of incoming calls for an emergency call center in a time series manner. They use five algorithms (*ReliefF*, *PCA*, *Freq. Discretization*, *Information Gain* and *K-means*) that are different in nature and combine the rankings computed grouping similar approaches and computing new weights as the mean of the individual weights. After that, all variables that are ranked among the top five positions in at least three of the groups compound the selected features. A possible way to perform FS for change detection in multivariate time series is the *Scenario-Based Random Forest* (*SBRF*) algorithm [25]. The aim of change detection is to figure out for a new scenario which label the current segment has at a give time-index by using only the information from the time series up to this point. The *SBRF* classifier is an extension of the Random Forest algorithm which estimates the generalization error on scenario level and not on time-stamp level. The methodology is used in a car crash detection application. Moreover, in the thesis work [26] they present three case studies in which FS is a step in the model creation. They used the following methods: *sequential forward/ backward selection* (*SFS*, *SBS*), *sequential forward/ backward floating selection* (*SFFS*, *SBFS*), the *n best features selection* (*nBest*) and the *best individual features*.

The main data characteristics of energy time series have been specifically analysed in [27]. To explore such data from different perspectives they consider two main categories: nature (nonstationarity, nonlinearity and complexity characteristics) and pattern (cyclicity, mutability or saltation, and randomicity or noise pattern). After that, FS for electricity load forecasting was done in a time series manner using correlation and instance based methods [28]. In [29] it is presented a survey on data mining techniques for time series forecasting of electricity. The survey focuses on the characteristics of the models and their configuration. *Wrapper* methods, *Artificial Neural Networks*, *mutual information*, *autocorrelation* and *ranking based methods* are mentioned as FS techniques used in the prediction of energy consumption. Finally, the work [2] uses temperature time series together with day of the week in order to estimate energy consumption.

2.3. Energy efficiency dataset

The reference building in which the energy consumption forecasting has been carried out is the Chemistry Faculty of the University of Murcia, which is a building used as a pilot for the H2020 ENTROPY project[1].

The dataset is composed of 5088 observations of 50 attributes that are measured hourly from 2016-02-02 00:00:00 until 2016-09-06 23:00:00, where time-stamps from 2016-02-05 00:00:00 until 2016-05-07 23:00:00 are missing data. Table 1 shows the number, name and sources of the dataset attributes. The output attribute is the *energy consumption* measured in KWh. Attributes *datetime* ("yyyy-MM-dd HH:mm:ss"), *season* (1–4), *day of the week* (1–7), and *holiday* (0,1) have been extracted from the date's observation. We have used meteorological data gathered from several sources and stations with the purpose to select the attributes from the most explanatory source according to our feature extraction analysis.

Weather Underground[2] is a web service that through its API provides the following real values: *temperature* (°C), *apparent temperature* (°C), *dew point* (°C), *humidity* (%), *wind speed* (m/s), *mean sea level pressure* (mbar), *visibility* (km) and *precipitations in last hour* (mm). We also use *one-hour predictions* for the first six previous attributes, together with *probability of precipitations* (%), *sky cover* (%) and *wind direction* (degrees) .

IMIDA[3] (The Research Institute of Agriculture and Food Development of Murcia) provides real time records of weather. We have selected two weather stations regarding proximity to the building: MO12 and MU62 and from each of them we have collected the following variables: *temperature* (mean, minimum and maximum) (°C), *humidity* (mean, minimum and maximum) (%), *radiation* (mean and maximum) (w/m^2), *wind speed* (mean and maximum) (m/s^2), *wind direction* (mean) (degrees), *precipitation* (mm), *dew point* (°C) and *vapour pressure deficit* (kPa).

3. A methodology for energy multivariate time series forecasting based on feature selection

We have followed the methodology shown in the Figure 3 to perform the energy time series forecasting. The following six steps have been systematically applied: database transformation, feature selection, regression, statistical

[1] http://entropy-project.eu
[2] https://www.wunderground.com/
[3] http://www.imida.es/

Number	***Name***	***Data source***
1–8	realWU_temp, realWU_feels, realWU_dewp, realWU_hum, realWU_wspd, realWU_visib_km, realWU_mslp, realWU_prep_1h	Weather Underground
9–17	pr_temp, pr_feels, pr_dewp, pr_hum, pr_pop, pr_wspd, pr_wdir_deg, pr_sky, pr_mslp	Weather Underground
18–33	stMO12_IMI_tmed, stMO12_IMI_tmax, stMO12_IMI_tmin, stMO12_IMI_hrmed, stMO12_IMI_hrmax, stMO12_IMI_hrmin, stMO12_IMI_radmed, stMO12_IMI_radmax, stMO12_IMI_vvmed, stMO12_IMI_vvmax, stMO12_IMI_dvmed, stMO12_IMI_prec, stMO12_IMI_dewpt, stMO12_IMI_dpv, stMU62_IMI_tmed, stMU62_IMI_tmax,	IMIDA MO12
34–45	stMU62_IMI_tmin, stMU62_IMI_hrmed, stMU62_IMI_hrmax, stMU62_IMI_hrmin, stMU62_IMI_radmed, stMU62_IMI_radmax, stMU62_IMI_vvmed, stMU62_IMI_vvmax, stMU62_IMI_dvmed, stMU62_IMI_prec, stMU62_IMI_dewpt, stMU62_IMI_dpv	IMIDA MU62
46	energy	Output attribute
47–50	season, day_of_the_week, holiday, datetime	Date's observations

Table 1: Attributes and data sources of the energy consumption dataset used in this paper.

tests, decision making and forecasting. Next, each step is described separately, and some of the names of the *Weka* classes and methods that are required throughout the process are indicated.

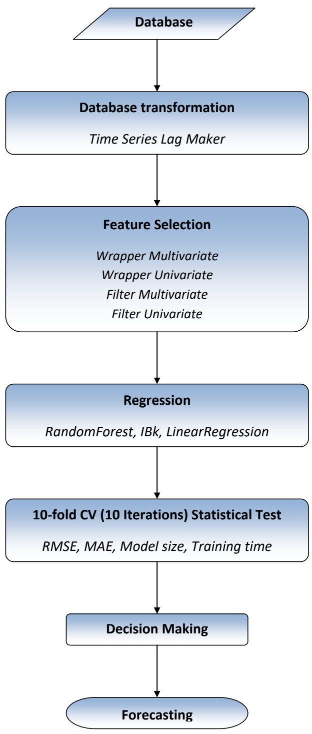

Figure 3: Methodology for feature selection for energy time series forecasting.

3.1. Database transformation

The first step of our methodology is to transform the database by creating lagged versions of variables for use in the time series problem. For this, the following steps are carried out:

1. Set an *artificial time-stamp* with start value 1. We use an artificial time index for convenience. In this way, no instances are inserted in the training data for the missing time-stamps.
2. Set the attributes to *lag*. The system can jointly model multiple attributes to lag simultaneously in order to capture dependencies between them. Because of this, modelling several series simultaneously can give different results for each series than modelling them individually. The rest of the attributes (*non lagged* attributes) are considered as *intervention* attributes (also called *overlay data*). We set attributes 1 to 46 as lagged attributes. Attributes 47, 48 and 49 are intervention attributes.
3. Set the minimum previous time step to create a lagged field. We set to 0 the *minimum lag length* to create. A value of 0 means that a lagged variable will be created that holds target values at time 0.
4. Set the maximum previous time step to create a lagged variable. We set to 3 the *maximum lag length* to create. A value of 3 means that a lagged variable will be created that holds target values at time -3. All time periods between the minimum and maximum lag will be turned into lagged variables. In this way, for example the variable *energy* will be transformed into 4 lagged variables *Lag_energy+0* (equivalent to the variable *energy*), *Lag_energy-1*, *Lag_energy-2* and *Lag_energy-3*.
5. Perform database transformation. A database of 189 attributes has been generated with the transformation.
6. Save transformed database with the name *TransformedDatabaseAux*. This auxiliary transformed database will be used later in the forecasting phase.
7. Remove *datetime* attribute. When using an artificial time index, the attribute *ArtificialTimeIndex* is added to the database, so the attribute *datetime* must be removed.
8. Save the final transformed database with the name *TransformedDatabase*. The final number of attributes of the transformed database is 188.

We use the class *weka.classifiers.timeseries.core.TSLagMaker* for this task. Data transformation can be done from the plugin tab in Weka's graphical "Explorer" user interface, or and using the *API* through a *Java* program.

3.2. Feature selection

Once the task of transforming the database is done, the next step is to apply FS on the *TransformedDatabase2* database. In *Weka*, FS is implemented with the class *weka.attributeSelection.AttributeSelection* through two components: the *search strategy* (*weka.attributeSelection.ASSearch* abstract class) and the *evaluator* (*weka.attributeSelection.ASEvaluation* abstract class). This allows users and programmers to configure a multitude of different methods for FS, both filter and wrapper, univariate and multivariate. Evaluators with names ending in *SubsetEval* configure multivariate methods, whereas those with names ending in *AttributeEval* configure univariate methods. For multivariate wrapper FS methods, the *weka.attributeSelection* package has the class *weka.attributeSelection.WrapperSubsetEval* which evaluates attribute sets by using a learning scheme with cross-validation and a performance measure. For univariate wrapper FS methods, the *weka.attributeSelection.ClassifierAttributeEval* class evaluates the worth of an attribute by using a user-specified classifier, cross-validation and a performance evaluation measure to use for selecting attributes. Since the FS and classification processes must be executed in batch mode, *Weka* offers the class *weka.classifiers.meta.AttributeSelectedClassifier* which is a meta-classifier where dimensionality of data is reduced by attribute selection before being passed on to a learning algorithm. Table 16 summarizes the packages and classes for FS in *Weka* used in this paper.

We applied eight different FS methods for regression shown in Table 2 and graphically in Figure 8. In Table 2, *Database #Id* denotes the identifier of the reduced database generated with each FS method. Each FS method is the result of a specific choice among the search strategy and the evaluator. We considered for this research five wrapper FS methods and three filter FS methods. Among them, five FS methods are multivariate and three FS methods are univariate. Table 14 shows the parameters used for each FS method. Next we show the search strategies and evaluators considered in this paper.

Database #Id.	*Type of FS method*	*Name*	*Search strategy*	*Evaluator*
#1	Wrapper Multivariate	*MOES-RF-MAE*	*MultiObjectiveEvolutionarySearch*	*RandonForest (MAE)*
#2	Wrapper Multivariate	*MOES-RF-RMSE*	*MultiObjectiveEvolutionarySearch*	*RandonForest (RMSE)*
#3	Wrapper Multivariate	*MOES-IBk-RMSE*	*MultiObjectiveEvolutionarySearch*	*IBk (RMSE)*
#4	Wrapper Multivariate	*MOES-LR-MAE*	*MultiObjectiveEvolutionarySearch*	*LinearRegression (MAE)*
#5	Wrapper Univariate	*RANKER-RF-RMSE*	*Ranker*	*RandonForest (RMSE)*
#6	Filter Multivariate	*GS-CFSSE*	*GreedyStepwise*	*CfsSubsetEval*
#7	Filter Univariate	*RANKER-RFAE*	*Ranker*	*ReliefFAttributeEval*
#8	Filter Univariate	*RANKER-PCA*	*Ranker*	*PrincipalComponents*

Table 2: Proposed feature selection methods for energy time series forecasting.

3.2.1. Search Strategies

As multivariate FS methods, we use a *probabilistic search strategy* and a *deterministic search strategy*. *MultiObjectiveEvolutionarySearch* [30] is the probabilistic strategy, and *GreedyStepwise* [31] is the deterministic strategy. *MultiObjectiveEvolutionarySearch* use multi-objective evolutionary computation where two objectives are optimized: the first one is a performance metric or statistical measure chosen by user with the evaluator, while the second one is the attribute subset cardinality, and it is to be minimized. The final output is given by the non-dominated solutions in the last population having the best fitness score for the first objective. *MultiObjectiveEvolutionarySearch* class has two multi-objective evolutionary algorithms implemented, *ENORA* and *NSGA-II*. *ENORA* is our *MOEA*, on which we are intensively working over the last decade. We have applied *ENORA* to constrained real-parameter optimization [32], fuzzy optimization [33], fuzzy classification [34], feature selection for classification [35] and feature selection for regression [36]. In this paper, we apply it to feature selection for regression in times series forecasting. *NSGA-II* algorithm has been designed by K. Deb et al. and has been proved to be a very powerful and fast algorithm in multi-objective optimization contexts of all kinds. In [36] is statistically tested that *ENORA* performs better than *NSGA-II* in terms of *hypervolume* [37, 38] for regression tasks, for which we have decided to use *ENORA* in this work. *GreedyStepwise* performs a greedy forward or backward search through the space of attribute subsets, stopping when the addition (forward direction) or deletion (backward direction) of any of the remaining attributes results in a decrease in evaluation, thus, it has no backtracking capability.

For univariate FS methods, *Ranker* method [39] is required. *Ranker* method ranks attributes by their individual evaluations. A threshold, or the number of attributes to retain, allows reducing the attribute set.

3.2.2. Evaluators

We considered the multivariate filter evaluator *ConsistencySubsetEval* [40]. *ConsistencySubsetEval* scores a subset of features as a whole, by projecting the training instances according to the attribute subset, and considering the consistency of class values in the obtained instance sets. As far as univariate filter evaluators are concerned, *ReliefAttributeEval* [41] and *PrincipalComponents* [42] were considered. *RelieffAttributeEval* evaluates the worth of an attribute by repeatedly sampling an instance and considering the value of the given attribute for the nearest instance of the same and different class. Can operate on both discrete and continuous class data. *PrincipalComponents* performs a principal components analysis and transformation of the data. Dimensionality reduction is accomplished by choosing enough eigenvectors to account for some percentage of the variance in the original data (default 95%). Attribute noise can be filtered by transforming to the principal components space, eliminating some of the worst eigenvectors, and then transforming back to the original space.

We use the wrapper *WrapperSubsetEval* [10] for multivariate FS methods and *ClassifierAttributeEval* [43] for univariate FS methods in conjunction with the predictors *RandomForest* [44], *IBk* [45] and *LinearRegression* [46], and with the metrics *root mean squared error* (*RMSE*) and *mean absolute error* (*MAE*) [47]. *RandomForest* is an *ensemble learning* method which constructs a forest of random trees with controlled variance, for classification or regression purposes. *IBk* is a simple instance-based learner that uses the class of the nearest k training instances for the class of the test instances and it is also valid for regression. *LinearRegression* uses the *Akaike* criterion for model selection, and is able to deal with weighted instances. Note that not all regression algorithms can be used as evaluators in wrapper FS methods due to their high computational time. *RandomForest*, *IBk* and *LinearRegression* are learning algorithms that offer a good compromise between performance and computational time so they are suitable as evaluators in wrapper FS methods.

3.3. Regression

Once FS was made, the next step was to perform regression with the reduced and *TransformedDatabase2* databases using different regression algorithms. We considered *RandomForest*, *IBk* and *LinearRegression* since these algorithms were used as evaluators in the wrapper FS methods. Additionally we used *Support Vector Machines* [48] and *Gaussian Processes* [49], which are widely used for time series forecasting [50], concretely the *Weka* implementations *SMOreg* and *GaussianProcesses*. *SMOreg* [51] implements the support vector machine for regression. The parameters can be learned using various algorithms, being *RegSMOImproved* the most popular algorithm. *GaussianProcesses* implements Gaussian processes for regression without hyperparameter-tuning. To make choosing an appropriate noise level easier, this implementation applies normalization/standardization to the target attribute as well as the other attributes. Both *SMOreg* and *GaussianProcesses* can use *Polykernel*, *PrecomputedKernelMatrixKernel*, *Puk*, *RBFKernel* or *StringKernel*. Table 15 shows the parameters used for the regression methods. Tables 3 and 4 show the evaluation in full training set for the *RMSE* and *MAE* metrics respectively.

	#1	*#2*	*#3*	*#4*	*#5*	*#6*	*#7*	*#8*	*TransformedDatbase*
RandomForest	5.0930	5.0286	5.2923	5.5701	5.9543	7.2083	5.3704	13.5680	7.8809
IBk	3.0201	3.5134	2.4937	1.3826	2.7927	1.5093	1.4044	0.0000	2.1045
LinearRegression	19.5455	18.4759	18.7110	18.3092	18.7878	22.1723	18.2264	53.5429	17.2416
SMOref	20.2988	19.1824	19.2648	19.0136	19.3193	23.1186	19.1566	55.5580	18.4857
GaussianProcesses	21.9302	21.7321	24.7275	19.4525	18.9750	22.1774	18.4000	54.5592	17.4686

Table 3: *RMSE* with full training set.

	#1	*#2*	*#3*	*#4*	*#5*	*#6*	*#7*	*#8*	*TransformedDatbase*
RandomForest	2.5667	2.6015	2.7341	2.8990	3.1639	3.7101	2.7528	8.5778	4.7050
IBk	0.0730	0.0824	0.0465	0.0271	0.0559	0.0231	0.0284	0.0000	0.0419
LinearRegression	11.2387	10.0955	10.2126	9.6797	10.1295	13.2144	10.4735	38.2477	10.1297
SMOref	10.0226	8.9893	9.0665	8.9401	9.0363	11.5677	8.9673	36.6050	8.7132
GaussianProcesses	15.2061	15.0741	17.8833	11.9989	10.5405	13.3437	10.9358	38.7237	10.5050

Table 4: *MAE* with full training set.

3.4. Statistical test

In order to detect over-fitting and prediction ability, the regression models have also been evaluated with cross-validation. Tables 5, 6, 7 and 8 show the evaluation in 10-fold cross-validation, 3 repetitions (a total of 30 models with each regression algorithm in each database), for the metrics *RMSE*, *MAE*, *Serialized_Model_Size* and *User_Time_training*[4] respectively. The result of the experiment has been analysed through a *paired t-test (corrected)*, with 0.05 significance, being *#1* the test base. For each result, a mark $*$ denotes that the result is statistically worse than the test base; similarly, a mark *v* denotes a statistically better result, and no mark denotes no statistically meaningful difference.

	#1	*#2*	*#3*	*#4*	*#5*	*#6*	*#7*	*#8*	*TransformedDatbase*
RandomForest	12.6685	12.9133	13.3814 $*$	14.4111 $*$	15.4203 $*$	18.7996 $*$	13.9174 $*$	36.7834 $*$	21.3455 *v*
IBk	17.7612	20.8112 $*$	17.2423	25.0447 $*$	25.8680 $*$	25.7792 $*$	22.7562 $*$	37.8315 $*$	29.5936 $*$
LinearRegression	19.3960	18.3234 *v*	18.5017 *v*	18.1808 *v*	18.6416	22.0898 $*$	18.1092 *v*	53.6083 $*$	17.7597 *v*
SMOref	20.0636	18.8337 *v*	18.9237 *v*	18.6714 *v*	18.9697 *v*	23.0016 $*$	18.8051 *v*	55.5770 $*$	18.2458 *v*
GaussianProcesses	21.9231	21.7133	24.7083 $*$	19.4114 *v*	18.8832 *v*	22.1160	18.3440 *v*	54.6361 $*$	17.8482 *v*

Table 5: *RMSE* with 10-fold cross-validation (3 repetitions).

[4]Intel (R) Core (TM) i5-4460 @ 3.20 GHz 3.20 GHz RAM 8.00 GB Operating Systems 64 bits, processor x64.

	#1	#2	#3	#4	#5	#6	#7	#8	TransformedDatbase
RandomForest	5.8264	6.0012 ∗	6.2785 ∗	6.8621 ∗	7.5242 ∗	9.0191 ∗	6.4675 ∗	23.3071 ∗	12.6164 ∗
IBk	8.8796	10.0150 ∗	8.6797	13.1307 ∗	13.3098 ∗	12.7038 ∗	11.0927 ∗	17.4372 ∗	14.3159 ∗
LinearRegression	11.2708	10.1276 v	10.2363 v	9.7287 v	10.1738 v	13.2454 ∗	10.5091 v	38.3269 ∗	10.6126 v
SMOref	10.0410	9.0122 v	9.0806 v	8.9702 v	9.0669 v	11.5835 ∗	8.9962 v	36.7292 ∗	8.9314 v
GaussianProcesses	15.3332	15.1402 v	17.9369 ∗	12.0857 v	10.6294 v	13.3943 v	11.0307 v	38.8031 ∗	10.9028 v

Table 6: *MAE* with 10-fold cross-validation (3 repetitions).

	#1	#2	#3	#4	#5	#6	#7	#8	TransformedDatbase
RandomForest	11.9955	11.9149 v	13.4332 ∗	12.6169 ∗	14.7657 ∗	16.3795 ∗	14.9757 ∗	20.3033 ∗	16.7962 ∗
IBk	0.5064	0.5796 ∗	0.4330 v	0.8735 ∗	0.5798 ∗	0.4329 v	0.5799 ∗	0.5804 ∗	0.7081 ∗
LinearRegression	0.1278	0.1274 v	0.1275 v	0.1285 ∗	0.1285 ∗	0.1275 v	0.1285 ∗	0.1278 ∗	0.1604 ∗
SMOref	0.1734	0.7654 ∗	0.6609 v	0.1060 ∗	0.1028 ∗	0.8805 ∗	1.1013 ∗	0.7658 ∗	7.6196 ∗
GaussianProcesses	168.4590	168.4900 ∗	168.3855 v	168.7841 ∗	168.7528 ∗	168.6051 ∗	168.8259 ∗	168.4904 ∗	175.3427 ∗

Table 7: *Serialized_Model_Size* ($\times 10^6$ bytes) with 10-fold cross-validation (3 repetitions).

	#1	#2	#3	#4	#5	#6	#7	#8	TransformedDatbase
RandomForest	0.9474	1.0349 ∗	0.7792 v	1.3432 ∗	0.9714	0.5708 v	0.7802 v	1.6078 ∗	3.0609 ∗
IBk	0.0005	0.0005	0.0000	0.0000	0.0005	0.0000	0.0016	0.0000	0.0005
LinearRegression	0.0042	0.0109	0.0026	0.0125 ∗	0.0089	0.0063	0.0115	0.0057	4.2172 ∗
SMOref	31.9255	29.0380 v	26.4307 v	62.5958 ∗	87.1901 ∗	75.5521 ∗	141.1615 ∗	9.0995 v	1626.4151 ∗
GaussianProcesses	115.4714	115.3620	115.4219	115.5542	110.7714 v	110.4302 v	110.5990 v	110.6505 v	114.0536

Table 8: *UserCPU_Time_training* (seconds) with 10-fold cross-validation (3 repetitions).

3.5. Decision making

Looking at tables 5 to 8 we can make a decision for choosing the best reduced database and, therefore, the best FS method. The best results have been obtained with the FS method *MOES-RF-MAE* (database *#1*) when *RandomForest* is used as regression algorithm, which show statistically significant differences with respect to the rest of the analysed FS methods for the *MAE* performance metric. For *RMSE* performance metric, FS method *MOES-RF-MAE* is also superior to the rest of FS methods, with statistically significant differences except for the FS method *MOES-RF-RMSE*. With respect to the *Serialized_Model_Size* and *UserCPU_Time_training* performance metrics, the results of the FS method *MOES-RF-MAE* by using *RandomForest* are acceptable in comparison to the rest of the methods. We can then choose the FS method *MOES-RF-MAE* and the database *#1* for the final forecasting process.

Table 9 shows the selected attributes with *MOES-RF-MAE*. Tab. 9 shows the selected attributes and their ranks and importances for each of the datasets. The rank and importance of the attributes has been obtained through a univariate wrapper feature selection method, where the search strategy is the *ranker* method, and the *evaluator* is *ClassifierAttributeEval* with *classifier = RandomForest* (with default parameters), *evaluationMeasure = MAE*, and *leaveOneAttributeOut = true*. An attribute is evaluated by measuring the impact of leaving it out from the full set.

Input attribute	Rank	Importance
Lag_energy-1	1	7.398
Lag_stMO12_IMI_radmax+0	2	1.337
holiday	3	0.367
Lag_energy-3	4	0.357
ArtificialTimeIndex	5	0.302
Lag_stMO12_IMI_radmed-3	6	0.273
Lag_pr_feels-2	7	0.248
Lag_pr_temp-2	8	0.172

Table 9: Selected attributes with *MOES-RF-MAE* (database *#1*) and their ranks.

3.6. *Forecasting*

Finally, in this section we analyse the prediction ability of the forecaster obtained with the selected attributes. We use the class *weka.classifiers.timeseries.WekaForecaster* for this task. Forecasting can be done from the plugin tab in Weka's graphical "Explorer" user interface, or using the *API* through a *Java* program. When an evaluation is performed, firstly the forecaster is trained on the data, and then it is applied to make a forecast at each time point (in order) by stepping through the data. These predictions are collected and summarized, using *MAE* and *RMSE* metrics, for each future time step predicted. We use in this paper three time units to forecasts, i.e. all the 1-step-ahead, 2-steps-ahead and 3-steps-ahead predictions are collected and summarized. This allows us to see, to a certain degree, how forecasts further out in time compare to those closer in time.

Tables 10 and 12 show the evaluation of the forecaster, with the database *#1*, on training data (70%) and test data (30%) respectively. The last 500 training data and the first 500 test data of these evaluations are also shown graphically in figures 4 and 6 respectively. To verify if the FS process has been effective both for the reduction of the complexity of the model and for the increase of its predictive capacity, the forecasting process has also been carried out on the database *TransformedDatase* (with all lagged variables and all overlay variables). Tables 11 and 13 show the evaluation of the forecaster with the database *TransformedDatase*, and figures 5 and 7 show graphically the evaluation of the last 500 training data and the first 500 test data respectively.

	1-step-ahead	*2-steps-ahead*	*3-steps-ahead*	*Average*
Number of instances	3559	3558	3557	-
MAE	2.6684	4.3897	5.8962	4.3181
RMSE	5.3008	9.4352	13.0256	9.2539

Table 10: Evaluation on training data (3562 instances) with *RandomForest* - database *#1*.

	1-step-ahead	*2-steps-ahead*	*3-steps-ahead*	*Average*
Number of instances	3559	3558	3557	-
MAE	4.4041	9.6858	18.2695	10.7865
RMSE	7.5987	14.0861	25.1676	15.6175

Table 11: Evaluation on training data (3562 instances) with *RandomForest* - *TransformedDatabase*.

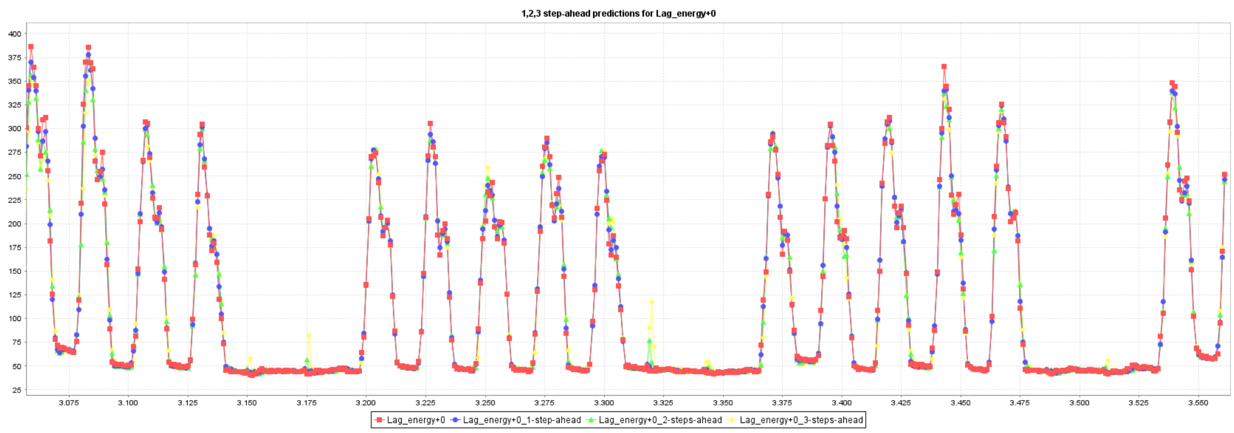

Figure 4: 1,2,3-step-ahead predictions for Lag-energy+0 evaluated on the last 500 training data with *RandomForest* - database *#1*.

4. Analysis of results and discussion

When observing the results of the experiments carried out using the proposed methodology, the following statements can be derived:

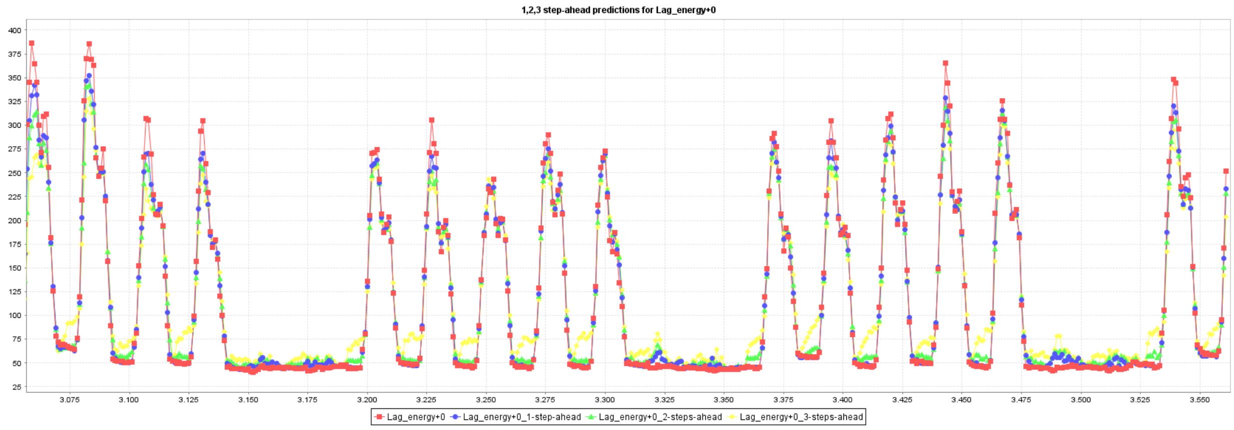

Figure 5: 1,2,3-step-ahead predictions for Lag-energy+0 evaluated on the last 500 training data with *RandomForest* - *TransformedDatabase*.

	1-step-ahead	*2-steps-ahead*	*3-steps-ahead*	*Average*
Number of instances	1526	1525	1524	-
MAE	10.9941	20.4655	32.7499	21.4032
RMSE	16.0509	28.7680	44.8343	29.8844

Table 12: Evaluation on test data (1526 instances) with *RandomForest* - database *#1*.

	1-step-ahead	*2-steps-ahead*	*3-steps-ahead*	*Average*
Number of instances	1526	1525	1524	-
MAE	26.7583	34.7768	49.7004	37.0785
RMSE	36.5563	45.0787	59.8209	47.1520

Table 13: Evaluation on test data (1526 instances) with *RandomForest* - *TransformedDatabase*.

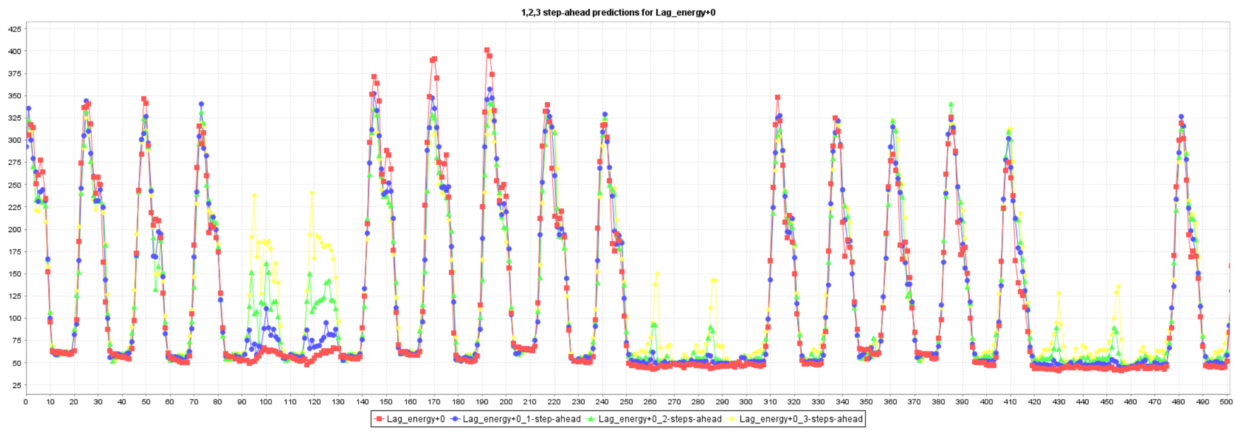

Figure 6: 1,2,3-step-ahead predictions for Lag-energy+0 evaluated on first 500 test data with *RandomForest* - database *#1*.

A. Regarding the FS process:

- As expected, wrapper FS methods show better performance than filter FS methods, and multivariate FS methods show better performance than univariate FS methods. Multivariate methods can identify interaction amongst features simultaneously, specially wrapper-based FS methods [52]. To make it possible, multivariate methods

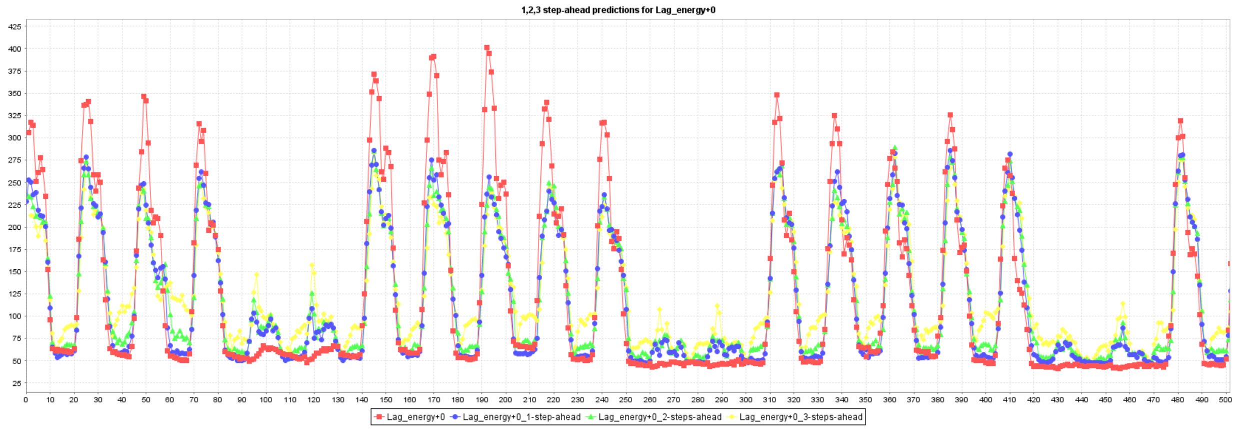

Figure 7: 1,2,3-step-ahead predictions for Lag-energy+0 evaluated on first 500 test data with *RandomForest* - *TransformedDatabase*.

evaluate the relevance of sets of features to determine which are the bests according to certain performance measure for a given task. However, multivariate wrapper feature selection methods present a high computation costs, since the number of possible subsets of feature is very high (2^w, being w the number of features) making the problem of finding the best subsets to be NP-Hard. To reduce the computational time, some deterministic search strategies, such as *GreedyStepwise*, can be used. The main disadvantage of these deterministic search techniques is that hidden and basic interactions could be missed due to the way the search space is traversed [53]. Probabilistic search techniques, such as *MultiObjectiveEvolutionarySearch*, can overcome this difficulties by allowing to generate new subsets in different locations of the search space guided by a metaheuristic. In this paper, we propose to use a multivariate wrapper feature selection method where the search strategy is based on multi-objective evolutionary computation, thus intrinsically overcoming the problem of interactions between features.

- For wrapper FS methods, the *RandomForest* evaluator has proven more effective than *IBk* and *LinearRegression* based evaluators. *SMOreg* and *GaussianProcesses* are discarded as evaluators for wrapper methods because of their excessive computational time. Run time of *RandomForest* is acceptable for wrapper FS methods setting the number of iterations to 10 (-I 10), and this method is not very sensitive to the variation of its parameters. However, *RandomForest* generates regression models larger than *IBk*, *LinearRegression* and *SMOreg*.

- *IBk* is very prone to over-fitting. Although in the evaluation on full training data the best results have been obtained with *IBk*, these results become poor when the evaluation is done on cross-validation, which indicates that *IBk* over-fits the regression models.

- *LinearRegression*, *SMOreg* and *GaussianProcesses* are not prone to over-fitting, but it has not been efficient for this problem.

- *MAE* has shown better behaviour than *RMSE* as metric performance in evaluators for wrapper FS methods. This can be seen in Table 5: the FS method *MOES-RF-MAE* (database *#1*) produces better results than the method *MOES-RF-RMSE* (database *#2*) when evaluated on cross-validation with *RandomForest* using the *RMSE* metric.

B. Regarding the forecasting process:

Tables 10 to 13 show how 1,2,3-steps-ahead predictions using the reduced database *#1* improve the 1,2,3-steps-ahead predictions using the database without performing feature selection. Using the averages of the 1,2,3-steps-ahead predictions (shown also in tables 10 to 13) we can calculate the percentage differences between the average predictions by doing feature selection and without doing so. With our methodology, *MAE* is improved by 59.97% and *RMSE* by 40.75%, evaluated on training data, and *MAE* is improved by 42.28% and *RMSE* by 36.62%, evaluated on test data.

5. Comparison with other methods proposed in literature

The metrics RMSE and MAE are two of the most common metrics used to measure accuracy for continuous variables and they are appropriate for model comparisons because they express average model prediction error in units of the variable of interest. However, in order to compare energy consumption prediction within several papers that do not use the same dataset or the same values of energy to be predicted it is not useful to compare such metrics whose magnitude depend on the range of the output data.

For that reason, we choose the coefficient of variance of the RMSE. CVRMSE is a non-dimensional measure calculated by dividing the RMSE of the predicted energy consumption by the mean value of the actual energy consumption. For example, a CVRMSE value of 5% would indicate that the mean variation in actual energy consumption not explained by the prediction model is 5% of the mean value of the actual energy consumption [**?**].

In the work with similar objectives [18], the preprocessing is carried out through correlation and *Principal Components Analysis* [42] and each day is divided in three moments alluding to occupation: morning, afternoon and night. That way, 3 different models are trained and the results are the following: *Random Forest* is selected at night and in the afternoon providing a *RMSE* of 1 and 3.87 *KWh* and *Bayesian Regularized Neural Networks* [54] is selected for the morning with *RMSE* = 7.08 *KWh*. In that sense, we could say that our FS approach overcomes this method in general. In the work [2], the temperature time series together with day of the week are used in order to estimate energy consumption. Results show again Random Forest as the outstanding model and the daily *CVRMSE* = 9%.

For current and future comparisons with further research, we obtained an hourly CVRMSE = 20 % and we have also averaged it per day obtaining a daily CVRMSE = 11 % for the 1-step case.

Multivariate ARIMA: we have also carried on the energy consumption forecasting using the traditional time series method ARIMA with exogenous regressors [55]. Results are much worst than using out machine learning oriented approach. Using our selected features, mean MAE is 119 and mean RMSE is 126. This results are way worst than ours but still better than using all variables with ARIMA: MAE increases between 35 and 55 KWh and RMSE incresaes between 37 and 58 Kwh.

6. Conclusions

In this work we have proposed a methodology for energy multivariate time series forecasting. The methodology is based on, firstly, database transformation into a form that standard machine learning algorithms can process, and then, systematically apply a set of feature selection methods for regression. The methodology deals with both lagged and intervention variables, unlike other works in the literature where only lagged variables are treated or the time series problem is univariate. The results of the experiments carried out show that the proposed methodology effectively reduces both the complexity of the forecast model and their *RMSE* and *MAE* in 1,2,3-steps-ahead predictions. The results of our methodology improve those obtained with other works reported in the literature, as well as those obtained with the *marima* package for multivariate time series forecasting.

Acknowledgment

This work has been sponsored by MINECO through the PERSEIDES project (ref. TIN2017-86885-R) and grant BES-2015-071956 and by the European Comission through the H2020-ENTROPY-649849 EU Project.

Appendix

References

[1] M. Akcin, A. Kaygusuz, A. Karabiber, S. Alagoz, B. B. Alagoz, C. Keles, Opportunities for energy efficiency in smart cities, in: Smart Grid Congress and Fair (ICSG), 2016 4th International Istanbul, IEEE, 2016, pp. 1–5.

[2] A. González-Vidal, A. P. Ramallo-González, F. Terroso-Sáenz, A. Skarmeta, Data driven modeling for energy consumption prediction in smart buildings, in: Big Data (Big Data), 2017 IEEE International Conference on, IEEE, 2017, pp. 4562–4569.

[3] A. Gonzalez-Vidal, P. Barnaghi, A. F. Skarmeta, Beats: Blocks of eigenvalues algorithm for time series segmentation, IEEE Transactions on Knowledge and Data Engineering.

[4] H. Liu, H. Motoda, Feature Selection for Knowledge Discovery and Data Mining, Kluwer Academic Publishers, Norwell, MA, USA, 1998.

Database #Id.	*Parameters*
#1	-E “weka.attributeSelection.WrapperSubsetEval -B weka.classifiers.trees.RandomForest -F 5 -T 0.01 -R 1 -E DEFAULT – -P 100 -I 10 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1” -S “weka.attributeSelection.MultiObjectiveEvolutionarySearch -generations 500 -population-size 100 -seed 1 -a 0 ”
#2	-E “weka.attributeSelection.WrapperSubsetEval -B weka.classifiers.trees.RandomForest -F 5 -T 0.01 -R 1 -E MAE – -P 100 -I 10 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1” -S “weka.attributeSelection.MultiObjectiveEvolutionarySearch -generations 500 -population-size 100 -seed 1 -a 0 ”
#3	-E “weka.attributeSelection.WrapperSubsetEval -B weka.classifiers.lazy.IBk -F 5 -T 0.01 -R 1 -E DEFAULT – -K 1 -W 0 -A “weka.core.neighboursearch.LinearNNSearch -A “weka.core.EuclideanDistance -R first-last”” -S “weka.attributeSelection.MultiObjectiveEvolutionarySearch -generations 500 -population-size 100 -seed 1 -a 0”
#4	-E “weka.attributeSelection.WrapperSubsetEval -B weka.classifiers.functions.LinearRegression -F 5 -T 0.01 -R 1 -E MAE – -S 0 -R 1.0E-8 -num-decimal-places 4” -S “weka.attributeSelection.MultiObjectiveEvolutionarySearch -generations 500 -population-size 100 -seed 1 -a 0”
#5	-E “weka.attributeSelection.ClassifierAttributeEval -execution-slots 1 -B weka.classifiers.trees.RandomForest -F 5 -T 0.01 -R 1 -E DEFAULT – -P 100 -I 10 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1” -S ”weka.attributeSelection.Ranker -T -1.8E308 -N 10”
#6	-E “weka.attributeSelection.CfsSubsetEval -P 1 -E 1” -S “weka.attributeSelection.GreedyStepwise -T -1.8E308 -N -1 -num-slots 1”
#7	-E “weka.attributeSelection.ReliefFAttributeEval -M -1 -D 1 -K 10” -S ”weka.attributeSelection.Ranker -T -1.8E308 -N 10”
#8	-E “weka.attributeSelection.PrincipalComponents -R 0.95 -A 5” -S “weka.attributeSelection.Ranker -T -1.8E308 -N 10”

Table 14: Parameters of the proposed feature selection methods for energy time series forecasting.

[5] L. Dannecker, Energy Time Series Forecasting: Efficient and Accurate Forecasting of Evolving Time Series from the Energy Domain, 1st Edition, Springer Vieweg, 2015.

[6] C. Deb, F. Zhang, J. Yang, S. E. Lee, K. W. Shah, A review on time series forecasting techniques for building energy consumption, Renewable and Sustainable Energy Reviews 74 (2017) 902 – 924. doi:https://doi.org/10.1016/j.rser.2017.02.085.

[7] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. 3 (2003) 1157–1182.

[8] R. Adhikari, R. K. Agrawal, An introductory study on time series modeling and forecasting, CoRR abs/1302.6613. arXiv:1302.6613.

[9] I. H. Witten, E. Frank, M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques (Third Edition), The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, Boston, 2011.

[10] R. Kohavi, G. H. John, Wrappers for feature subset selection, Artificial intelligence 97 (1-2) (1997) 273–324.

[11] N. Japkowicz, M. Shah, Evaluating Learning Algorithms: A Classification Perspective, Cambridge University Press, New York, NY, USA, 2011.

[12] A. G. Karegowda, A. S. Manjunath, M. A. Jayaram, Comparative study of attribute selection using gain ratio and correlation based feature

Name	*Parameters*
RandomForest	-P 100 -I 100 -num-slots 1 -K 0 -M 1.0
	-V 0.001 -S 1
IBk	-K 1 -W 0
	-A “weka.core.neighboursearch.LinearNNSearch
	-A “weka.core.EuclideanDistance -R first-last””
LinearRegression	-S 0 -R 1.0E-8 -num-decimal-places 4
SMOreg	-C 1.0 -N 0
	-I “weka.classifiers.functions.supportVector.RegSMOImproved
	-T 0.001 -V -P 1.0E-12 -L 0.001 -W 1”
	-K “weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007”
GaussianProcesses	-L 1.0 -N 0
	-K “weka.classifiers.functions.supportVector.PolyKernel
	-E 1.0 -C 250007” -S 1

Table 15: Parameters of the regression methods.

Name	Description
weka.classifiers.timeseries.core.TSLagMaker	Class for creating lagged versions of target variable(s) for use in time series forecasting
weka.attributeSelection	Package for feature selection
weka.attributeSelection.AttributeSelection	Class for feature selection
weka.attributeSelection.ASSearch	Abstract class for search strategy
weka.attributeSelection.ASEvaluation	Abstract class for evaluation
weka.classifiers.AbstractClassifier	Abstract classifier
weka.classifiers.SingleClassifierEnhancer	Abstract utility class, extends *AbstractClassifier*
weka.classifiers.meta.AttributeSelectedClassifier	Meta-classifier for feature selection + classification/regression, extends *SingleClassifierEnhancer*
weka.attributeSelection.GreedyStepwise	Class for greedy stepwise search strategy, extends *ASSearch*
weka.attributeSelection.MultiObjectiveEvolutionarySearch	Class for multi-objective evolutionary search strategy, extends *ASSearch*
weka.attributeSelection.PSOSearch	Class for particle swarm optimization search strategy, extends *ASSearch*
weka.attributeSelection.Ranker	Class to rank attributes in univariate feature selection methods, extends *ASSearch*
weka.attributeSelection.WrapperSubsetEval	Class for multivariate wrapper feature selection methods, extends *ASEvaluation*
weka.attributeSelection.ConsistencySubsetEval	Class for multivariate filter feature selection methods, extends *ASEvaluation*
weka.attributeSelection.ClassifierAttributeEval	Class for univariate wrapper feature selection methods, extends *ASEvaluation*
weka.attributeSelection.ReliefFAttributeEval	Class for univariate filter feature selection methods, extends *ASEvaluation*
weka.attributeSelection.PrincipalComponents	Class for univariate filter feature selection methods, extends *ASEvaluation*
weka.classifiers.trees.RandomForest	Class for constructing a forest of random trees, extends *weka.classifiers.meta.Bagging*
weka.classifiers.lazy.IBk	Class that implements an instance-based learning algorithm, extends *weka.classifiers.Classifier*
weka.classifiers.functions.LinearRegression	Class for using linear regression for prediction, extends *weka.classifiers.AbstractClassifier*
weka.classifiers.timeseries.WekaForecaster	Class that implements time series forecasting using a *Weka* regression scheme

Table 16: Packages and classes for feature selection in Weka used in this paper.

selection, International Journal of Information Technology and Knowledge Management 2 (2) (2010) 271–277.
[13] M. A. Hall, Correlation-based feature selection for machine learning, Tech. rep., University of Waikato (1999).
[14] A. Ahmad, L. Dey, A feature selection technique for classificatory analysis, Pattern Recognition Letters 26 (1) (2005) 43–56.
[15] S. Salzberg, C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993, Machine Learning 16 (3) (1994) 235–240. doi:10.1007/BF00993309.
[16] H.-X. Zhao, F. Magoulès, Feature selection for predicting building energy consumption based on statistical learning method, Journal of Algorithms & Computational Technology 6 (1) (2012) 59–77.
[17] O. Utterbäck, Feature selection methods with applications in electrical load forecasting, Master's Theses in Mathematical Sciences.
[18] A. González-Vidal, V. Moreno-Cano, F. Terroso-Sáenz, A. F. Skarmeta, Towards energy efficiency smart buildings models based on intelligent data analytics, Procedia Computer Science 83 (2016) 994–999.
[19] S. F. Crone, N. Kourentzes, Feature selection for time series prediction–a combined filter and wrapper approach for neural networks, Neurocomputing 73 (10-12) (2010) 1923–1936.
[20] S. Hido, T. Morimura, Temporal feature selection for time-series prediction, in: 2012 21st International Conference on Pattern Recognition (ICPR 2012), IEEE, 2012, pp. 3557–3560.
[21] O. García-Hinde, V. Gómez-Verdejo, M. Martínez-Ramón, C. Casanova-Mateo, J. Sanz-Justo, S. Jiménez-Fernández, S. Salcedo-Sanz, Feature selection in solar radiation prediction using bootstrapped svrs, in: Evolutionary Computation (CEC), 2016 IEEE Congress on, IEEE,

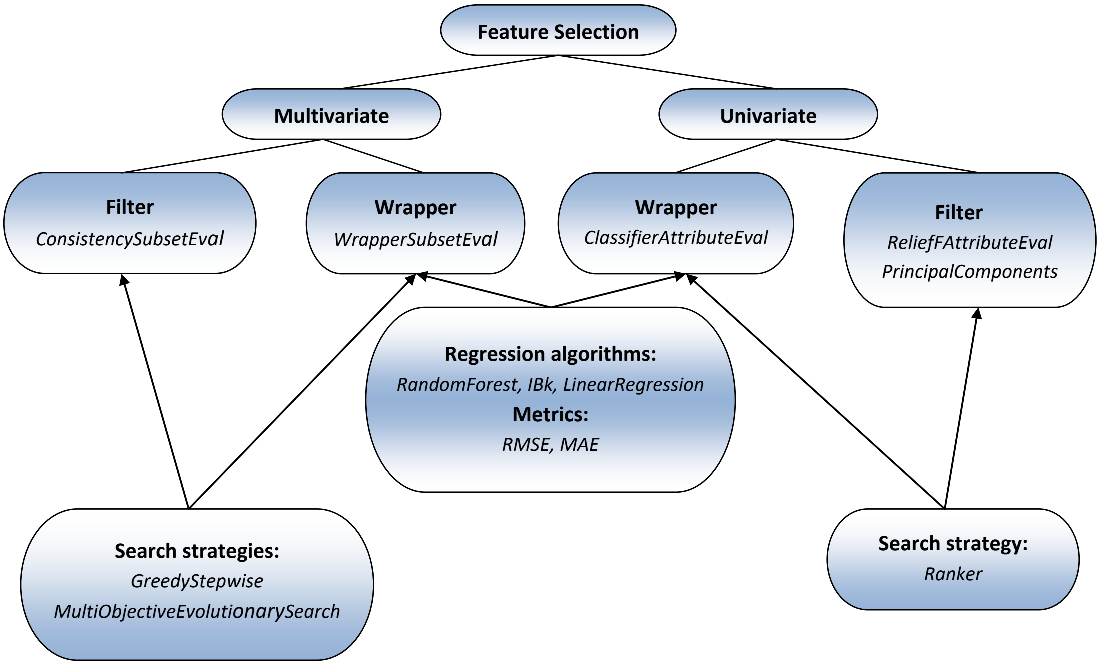

Figure 8: Organization chart of the proposed feature selection methods for energy time series forecasting.

2016, pp. 3638–3645.

[22] D. OLeary, J. Kubby, Feature selection and ann solar power prediction, Journal of Renewable Energy 2017.

[23] C. Feng, M. Cui, B.-M. Hodge, J. Zhang, A data-driven multi-model methodology with deep feature selection for short-term wind forecasting, Applied Energy 190 (2017) 1245–1257.

[24] R. G. Pajares, J. M. Benítez, G. S. Palmero, Feature selection for time series forecasting: A case study, in: Hybrid Intelligent Systems, 2008. HIS'08. Eighth International Conference on, IEEE, 2008, pp. 555–560.

[25] M. Botsch, J. A. Nossek, Feature selection for change detection in multivariate time-series, in: Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on, IEEE, 2007, pp. 590–597.

[26] E. Ferreira, Model selection in time series machine learning applications, PhD Thesis. University of Oulu Graduate School. University of Oul.

[27] L. Tang, C. Wang, S. Wang, Energy time series data analysis based on a novel integrated data characteristic testing approach, Procedia Computer Science 17 (2013) 759–769.

[28] I. Koprinska, M. Rana, V. G. Agelidis, Correlation and instance based feature selection for electricity load forecasting, Knowledge-Based Systems 82 (2015) 29–40.

[29] F. Martínez-Álvarez, A. Troncoso, G. Asencio-Cortés, J. C. Riquelme, A survey on data mining techniques applied to electricity-related time series forecasting, Energies 8 (11) (2015) 13162–13193.

[30] F. Jiménez, G. Sánchez, J. García, G. Sciavicco, L. Miralles, Multi-objective evolutionary feature selection for online sales forecasting, Neurocomputing.

[31] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 2nd Edition, Prentice-Hall, 2003.

[32] F. Jiménez, A. Gómez-Skarmeta, G. Sánchez, K. Deb, An evolutionary algorithm for constrained multi-objective optimization, in: Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress, Vol. 2 of CEC '02, IEEE Computer Society, Washington, DC, USA, 2002, pp. 1133–1138.

[33] F. Jiménez, G. Sánchez, P. Vasant, A multi-objective evolutionary approach for fuzzy optimization in production planning, J. Intell. Fuzzy Syst. 25 (2) (2013) 441–455.

[34] F. Jiménez, G. Sánchez, J. M. Juárez, Multi-objective evolutionary algorithms for fuzzy classification in survival prediction, Artificial Intelligence in Medicine 60 (3) (2014) 197–219.

[35] F. Jiménez, E. Marzano, G. Sánchez, G. Sciavicco, N. Vitacolonna, Attribute selection via multi-objective evolutionary computation applied to multi-skill contact center data classification, in: Proc. of the IEEE Symposium on Computational Intelligence in Big Data (IEEE CIBD 15), IEEE, 2015, pp. 488–495.

[36] F. Jiménez, G. Sánchez, J. García, G. Sciavicco, L. Miralles, Multi-objective evolutionary feature selection for online sales forecasting, Neurocomputing 234 (2017) 75–92.

[37] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: empirical results, Evolutionary Computation 8 (2) (2000) 173 – 195.

[38] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, V. Grunert da Fonseca, Performance assessment of multiobjective optimizers: An analysis and review, IEEE Transactions on Evolutionary Computation 7 (2002) 117–132.

[39] J. Novakovic, Toward optimal feature selection using ranking methods and classification algorithms, Yugoslav Journal of Operations Research 21 (1).

[40] H. Liu, R. Setiono, A probabilistic approach to feature selection - a filter solution, in: Proceedings of the 13th International Conference on Machine Learning (ICML), Vol. 96, 1996, pp. 319–327.

[41] K. Kira, L. A. Rendell, A practical approach to feature selection, in: Proceedings of the Ninth International Workshop on Machine Learning, ML92, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992, pp. 249–256.

[42] H. Abdi, L. J. Williams, Principal component analysis, Wiley Interdisciplinary Reviews: Computational Statistics 2 (4) (2010) 433–459. doi:10.1002/wics.101.

[43] R. Schafer, Accurate and efficient general-purpose boilerplate detection for crawled web corpora, Language Resources and EvaluationOnline first. doi:10.1007/s10579-016-9359-2.

[44] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32.

[45] D. W. Aha, D. Kibler, M. K. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1) (1991) 37–66. doi:10.1023/A:1022689900470.

[46] X. Yan, Linear Regression Analysis: Theory and Computing, World Scientific Publishing Company Pte Limited, 2009.

[47] C. J. Willmott, K. Matsuura, Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, Climate Research 30 (1) (2005) 79–82, cited By (since 1996)149.

[48] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (3) (1995) 273–297.

[49] C. E. Rasmussen, C. K. I. Williams, Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning), The MIT Press, 2005.

[50] G. Rubio, H. Pomares, L. J. Herrera, I. Rojas, Kernel methods applied to time series forecasting, in: F. Sandoval, A. Prieto, J. Cabestany, M. Graña (Eds.), Computational and Ambient Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 782–789.

[51] S. K. Shevade, S. S. Keerthi, C. Bhattacharyya, K. R. K. Murthy, Improvements to the smo algorithm for svm regression, IEEE Transactions on Neural Networks 11 (5) (2000) 1188–1193. doi:10.1109/72.870050.

[52] L. Chuang, C. Ke, C. Yang, A hybrid both filter and wrapper feature selection method for microarray classification, CoRR abs/1612.08669. arXiv:1612.08669.

[53] L. C. Molina, L. Belanche, À. Nebot, Feature selection algorithms: A survey and experimental evaluation, in: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan, 2002, pp. 306–313.

[54] B. F., W. D., Bayesian Regularization of Neural Networks, Vol. 458, Livingstone D.J. (eds) Artificial Neural Networks. Methods in Molecular Biology, Humana Press, 2008, Ch. 3, pp. 23–42.

[55] H. Spliid, Multivariate ARIMA and ARIMA-X Analysis, CRAN, license GPL-2, Version 2.2, RoxygenNote 5.0.1 (2017).