Energy Consumption Prediction Models

Mateusz Borowiak, Łukasz Królik,
Piotr Kurzawa, Kacper Stępień

July 4, 2017

Abstract

Producing adequate amount of energy plays a significant role in today's energy economics. It is related with emergence of smart grids and increasing impact of renewable energy sources. In this paper, we discuss usage of popular machine learning algorithms to forecast power consumption basing on weather data.

1 Energy Data

Data for the research is Campbell Creek Research House 3 data for 2013-2014[2] and come from Oak Ridge National Laboratory, Building Technologies Research and Integration Center. Contains end use breakdowns of energy use and various indoor environmental conditions collected at a 15 minute time stamp from 1st October 2013 to 1st October 2014.

Data was prepared to use in test - the empty values problem was fixed, and all NA values were approximated with neighbour samples.

2 Algorithms Overview

To solve our problem it was necessary to find model, which could predict a labelled value in base of another variables. This problem is kind of regression problem. We applied 3 popular approaches of finding model - support vector machines, decision trees and neural networks.

2.1 Reggresion

Regression is concerned with modeling the relationship between variables that is iteratively refined using a measure of error in the predictions made by the model.

Regression methods are a workhorse of statistics and have been co-opted into statistical machine learning. This may be confusing because we can use regression to refer to the class of problem and the class of algorithm. Really, regression is a process.

The most popular regression algorithms are:

- Ordinary Least Squares Regression (OLSR),
- Linear Regression,
- Logistic Regression,
- Stepwise Regression,
- Multivariate Adaptive Regression Splines (MARS),
- Locally Estimated Scatterplot Smoothing (LOESS).

2.2 Support Vector Machines

Support Vector Machines (SVM) are set of algorithms, which try to determine border between different decision classes. Formal definition is claiming that there is a classificator, which dividing p-dimention decision space, by use of (p-1) dimention hyperplanes. It is important to separate classes in wide margins. It isn't easy way, because usually borders aren't clear, and objects from another classes can be mixed. It is necessary to allow locate single objects from one class in other side. SVM can be also used for solving non-linear classification problems. The solution is using kernels - a functions which transforms objects to higher hyperplane, what can simplify the process of finding borders. The angle and position of hyperplanes and margins is calculated by decreasing value of cost function by using quadratic programming (QP).

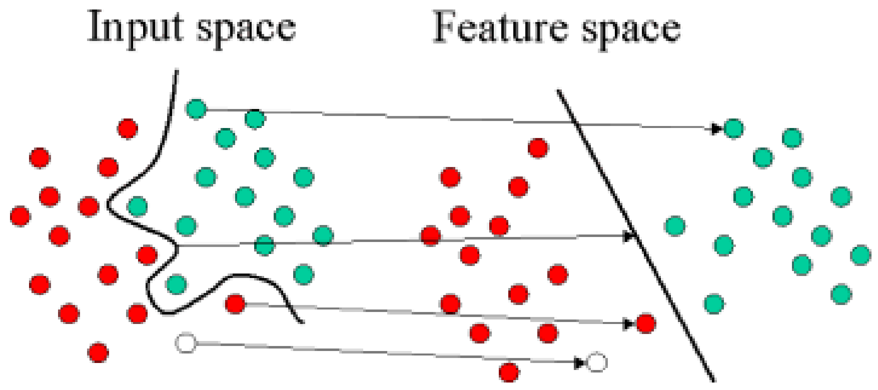

Figure 1:

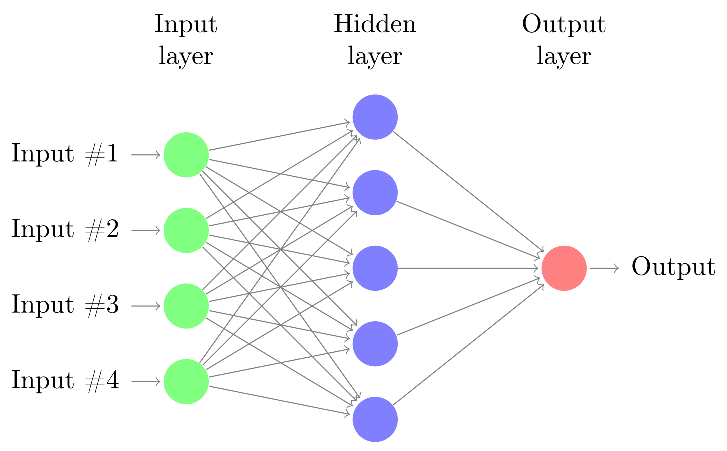

Figure 2: Simple neural network.

2.3 Decision Trees

Decision trees methods use tree-like graphs to represent information. Nodes indicate decision alternatives and leaves symbolize outcome. In every step of this family of algorithms we split data into two or more as far as possible homogeneous subsets. Tree decides where to split basing on algorithms like a Gini Index or Chi-Square.

The most popular decision trees algorithms are:

- Classification and Regression Tree (CART),
- Random Forest,
- Chi-square Automatic Interaction Detection (CHAID),
- Iterative Dichotomiser 3 (ID3),

2.4 Neural networks

Neural networks (NNs) can be defined as a function of a set of derived inputs called *hidden nodes*. They are nonlinear functions of the original inputs of neural network and they are organized in *hidden layers*. Each of the nodes contains a *activation function* - transformation of a linear combination of the input variables.

Patterns are presented to the network via the input layer, which communicates to hidden layers using a system of weighted connections. Hidden layers then link to output layer where the answer is output.

The two most popular types of neural networks are Feedforward neural network and Recurrent neural network.

3 Implementation Of Selected Algorithms

To predict energy consumption we selected algorithms from each category in chapter 2. As a pure regression algorithm we have Gaussian Processes. Support vector machine family represent SVM nu,epsilon regression and variant with Least Square method named LS-SVM. Tree based algorithms are Random Forest and Bayesian Regression Trees. From neural network type we have Feed Forward Neural Network and Bayesian Regularized Neural Networks.

3.1 Gaussian Processes

This function fits the following GP model, $y(x) = \; + Z(x)$, $x \in [0, 1]d$, where Z(x) is a GP with mean 0, V $ar(Z(xi)) = \sigma 2$, and $Cov(Z(xi), Z(xj)) = \sigma$ 2Rij . Entries in covariance matrix R are determined by corr and parameterized by beta, a d-vector of parameters. For computational stability R-1 is replaced with R -1 δlb , where $R\delta lb = R + \delta$ lbI and δlb is the nugget parameter described in Ranjan et al. (2011).

3.2 SVM epsilon regression

ksvm uses John Platt's SMO algorithm for solving the SVM QP problem an most SVM formulations. On the spoc-svc, kbb-svc, C-bsvc and eps-bsvr formulations a chunking algorithm based on the TRON QP solver is used. For multiclass-classification with k classes, k>2, ksvm uses the 'one-against-one'-approach, in which k(k-1)/2 binary classifiers are trained; the appropriate class is found by a voting scheme, The spoc-svc and the kbb-svc formulations deal with the multiclass-classification problems by solving a single quadratic problem involving all the classes. If the predictor variables include factors, the formula interface must be used to get a correct model matrix. In classification when prob.model is TRUE a 3-fold cross validation is performed on the data and a sigmoid function is fitted on the resulting decision values f. The data can be passed to the ksvm function in a matrix or a data.frame, in addition ksvm also supports input in the form of a kernel matrix of class kernelMatrix or as a list of character vectors where a string kernel has to be used. The plot function for binary classification ksvm objects displays a contour plot of the decision values with the corresponding support vectors highlighted. The predict function can return class probabilities for classification problems by setting the type parameter to "probabilities". The problem of model selection is partially addressed by an empirical observation for the RBF kernels (Gaussian , Laplace) where the optimal values of the statistics. When using an RBF kernel and setting kpar to "automatic", ksvm uses the sigest function to estimate the quantiles and uses the median of the values.

3.3 SVM nu regression

nu regression is newer version of SVR. The most important difference between this algorithms is nu parameter, which is used to apply a penalty to the optimization for points which were not correctly predicted.

The previous method (epsilon-SVR) use parameter epsilon, which could be number from range [0, inf]. It means that there is no penalty associated with points which are predicted within distance epsilon from the actual value.

This method use nu from range [0,1]. This parameter can simplify the model, by decreasing amount of support vectors. This value represent limit, how many best support vector will be used from all samples.

3.4 LS-SVM

LS-SVM [13] is variant of SVM where Least Square method were used to approximate the position of hyperplanes. The algorithm is based on the minimization of a classical penalized least-squares cost function. This approach decompose quadratic programming problem to problem of linear programming of set linear equations. The most popular method to solve this problem is KKT. LS-SVM is more basic and efficient method than classical SVM.

It was possible to use LS-SVM method, with linking library Liquid-SVM and calling the function lsSVM. The parameters of LsSVM function are:

- X - input matrix (or dataFrame) of training data
- Y - output matrix (or dataFrame) of response labels
- kpar- kernel parameters with set sigma parameter value to 2. This value was calculated automatic. It means the standard deviation of kernel.
- gamma - parameter controls the shape of the separating hyperplane. Increasing gamma usually increases number of support vectors.

3.5 Feedforward neural network

FFNN[7] is an artificial neural network where in contrast to recurrent neural network, signals are moving forward way only. From input nodes through hidden layers to output nodes. There is no loops or cycles in network. For learning phase it uses model developed by Widrow and Hoff ADALINE which stands for Adaptive Linear Element. Its function uses weight w and bias b: $y_i = wx_i + b$. Linear error is calculated by subtracting desired output and the output of the linear combiner.

Listing 1: Sample usage of nnet package with sample iris data

```
library("nnet")

ir <- rbind(iris3[,,1],iris3[,,2],iris3[,,3])
targets <- class.ind( c(rep("s", 50), rep("c", 50),
                        rep("v", 50)) )
samp <- c(sample(1:50,25), sample(51:100,25),
           sample(101:150,25))
nnetOutput <- nnet(ir[samp,], targets[samp,],
                     size = 2, rang = 0.1, decay = 5e-4, maxit = 200)
```

In this paper as a implementation of FFNN we used *nnet*[8] package from *CRAN* repository. Moreover we use *cmeans* clustering algorithm from *e1071* package. First, we divide our data into 100 fuzzy clusters. Next we run neural net algorithm with initial weights of network set to maximum value of fit to cluster. It helps us with randomness nature of neural networks algorithms.

3.6 Random Forest

Random Forest is tree based algorithm where instead of creating one tree, we are constructing multiple trees. It is known as a ensemble method because each model "votes" their outcome, so number of weaker classifiers can form a powerful model. Hence, one of advantages of this algorithm is reducing over-fitting.

Definition 1 (Random Forest) *A random forest is a classifier consisting of a collection of tree structured classifiers* $\{h(x, \theta_k), k = 1, ...\}$ *where the* $\{\theta_k\}$ *are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x .[9]*

In the experiment we used R language implementation from *randomForest*[10] package. Number of decision trees to be grown was set to 200.

Listing 2: Invocation of random forest algorithm in experiment. Unlike other algorithms there is no distinction between learning and predicting phase, beacause results are already available in $predicted component of random forest output object

```
randomForestOutput<-randomForest(dataToTrain, yTraining,
                                  dataToTest, yTest, ntree=200)
randomForestOutput$predicted
```

3.7 Bayesian Additive Regression Trees

Bayesian additive regression trees (BART) were introduced in paper [3]. They belong to group of ensemble-of-trees methods, which recently became popular choices for forecasting in both regression and classification problems. Unlike other methods such as random forests, it relies on an underlying Bayesian probability model rather than a pure algorithm.

BART is a Bayesian approach to nonparametric function estimation using regression trees, which rely on recursive binary partitioning of p-dimentional predictor space (where p is number of variables) into a set of hyperrectangles. It can be considered a sum-of-trees ensemble, in which each tree is constrained by a prior to be a weak learner. The formula of the BART model is presented in Equation 1.

$$Y = f(X) + \epsilon \approx \tau_1^M(X) + ... + \tau_m^M(X), \epsilon \sim N_n(0, \sigma^2 I_n) \tag{1}$$

In the above formula there are m distinct regression trees. Each of them are composed of a tree structure (denoted by τ) and the parameters at the terminal nodes (often called *leaves*), denoted by M. Together, they represent an entire tree with its structure and set of leaf parameters.

Structure of a tree includes information on how any observation recurses down the tree. For each internal node of the tree, there is a *splitting rule*:

$$x_j < c, \tag{2}$$

where x_j is a splitting variable and c is a splitting value. During the processing, an observation moves to the left child node if the above condition is satisfied (otherwise, it moves to the right child node). The process continues until a terminal node is reached, and then the observation reveives the leaf values of the terminal node. The sum of the leaf values arrived at by recursing down all m trees is the observation's predicted value.

BART consists of set of priors for the structure and the leaf parameters and a likelihood for data in the terminal nodes. The aim of the priors is to provide regularization - they are preventing any single regression tree from dominating the total fit.

In this paper we used a popular implementation of BART algorithm for R language called *bartMachine* [4]. The package is available on *CRAN* repository, it is also included in popular *caret* framework.

Listing 3: Sample usage of bartMachine package

```
library("bartMachine")
set.seed(11)
n = 200
p = 5
X = data.frame(matrix(runif(n*p), ncol=p))
y = 10*sin(pi*X[,1]*X[,2]) + 20
        * (X[,3] -.5)^2 + 10* X[,4] + 5 * X[,5] + rnorm(n)

bart_machine = bartMachine(X, y)
summary(bart_machine)
```

3.8 Bayesian Regularized Neural Networks

Bayesian Regularized Neural Networks (also known as BRNN, *Bayesian Regularization for Feed-Forward Neural Networks*) is a variant of feed-forward neural networks, introduced in MacKay's work [5]. In order to avoid overfitting during the increase of number of inputs and neurons in a single layer neural network, he used penalized estimation using Bayesian approaches. In his work, MacKay fit a two layer neural network and developed algorithms used to obtain estimates of all parameters using Empirical Bayes approach:

Definition 2 (Empirical Bayes approach) *Let θ be the vector of weights, biases and connections strengths and $p(\theta|\sigma_\theta^2) = MN(\theta, \sigma_\theta^2)$ be a prior distribution, where MN stands for the multivariate normal distribution, and σ_θ^2 is a variance common to all elements of θ. In Empirical Bayes approach, these two steps are repeated iteratively until covergence:*

- *Obtain conditional posterior modes of the elements in θ, assuming that σ_θ^2 and σ_e^2 are known. These modes are obtained by minimizing the augmented sum of squares;*
- *Update the variance components by maximizing $p(y|\sigma_\theta^2, \sigma_e^2)$.*

Implementation of this algorithm for R language is available in the popular *brnn* package [6]. In addition to above work, it uses the Nguyen and Widrow algorithm to assign initial weights and the Gauss-Newton algorithm to perform the optimization.

Listing 4: Sample usage of brnn package

```
library("brnn")

x1 = seq(0,0.23,length.out=25)
y1 = 4*x1+rnorm(25,sd=0.1)
x2 = seq(0.25,0.75,length.out=50)
y2 = 2-4*x2+rnorm(50,sd=0.1)
x3 = seq(0.77,1,length.out=25)
y3 = 4*x3-4+rnorm(25,sd=0.1)
x = c(x1,x2,x3)
y = c(y1,y2,y3)

out = brnn(y~x,neurons=2)
```

4 Experiment

The aim of the experiment is to check which of selected algorithm is the best in predicting energy usage. From the entire Campbell Creek Research House 3 dataset we chose 7 attributes for computations – power consumption and outdoor weather factors:

- Total Energy Consumption
- Outside temperature, Weather station,
- Precipitation,

- Humidity, Weather station,
- Solar radiation, Weather station,
- Horizontal wind speed, Weather station,
- Wind direction, Weather station,

Of course Total Energy Consumption was the result of combination of other attributes. In order to reduce amount of time needed for computations we limited number of samples. We took first 5000 samples as a training set and next 5000 samples as test set. In other words training set contains samples from 1st October 2013 to 22nd November and test set from 22nd November to 13rd January 2014. All methods was tested multiple times to pick out values of parameters (e.g. epsilon in svm or number of layers in neural network) that guarantees best results. As a measure of how good algorithm is, we used mean squared error. To provide results that could be reproduced before invocation of neural network algorithms we called *set.seed(1)* function which set the seed of R‘s random number generator. Times of learning phase and predicting phase were measured ten times with *system.time()* function except random forest algorithm where predicting phase was done during invocation of function creating random forest classifier.

5 Results

Results of experiment are visualized in Figure 3, besides name of method in legend there are values of Mean squared error. Black line represents actual values of energy usage in test set. Graph is pretty much mixed, but we can read off that Bayesian Regularized Neural Networks (orange) predicted a lot of outliers values. Furthermore values from Feed Forward Neural Network (purple) clearly fluctuate under actual ones. It also can be seen that prediction from Bayesian Additive Regression (purple) Trees relatively project real power consumption.

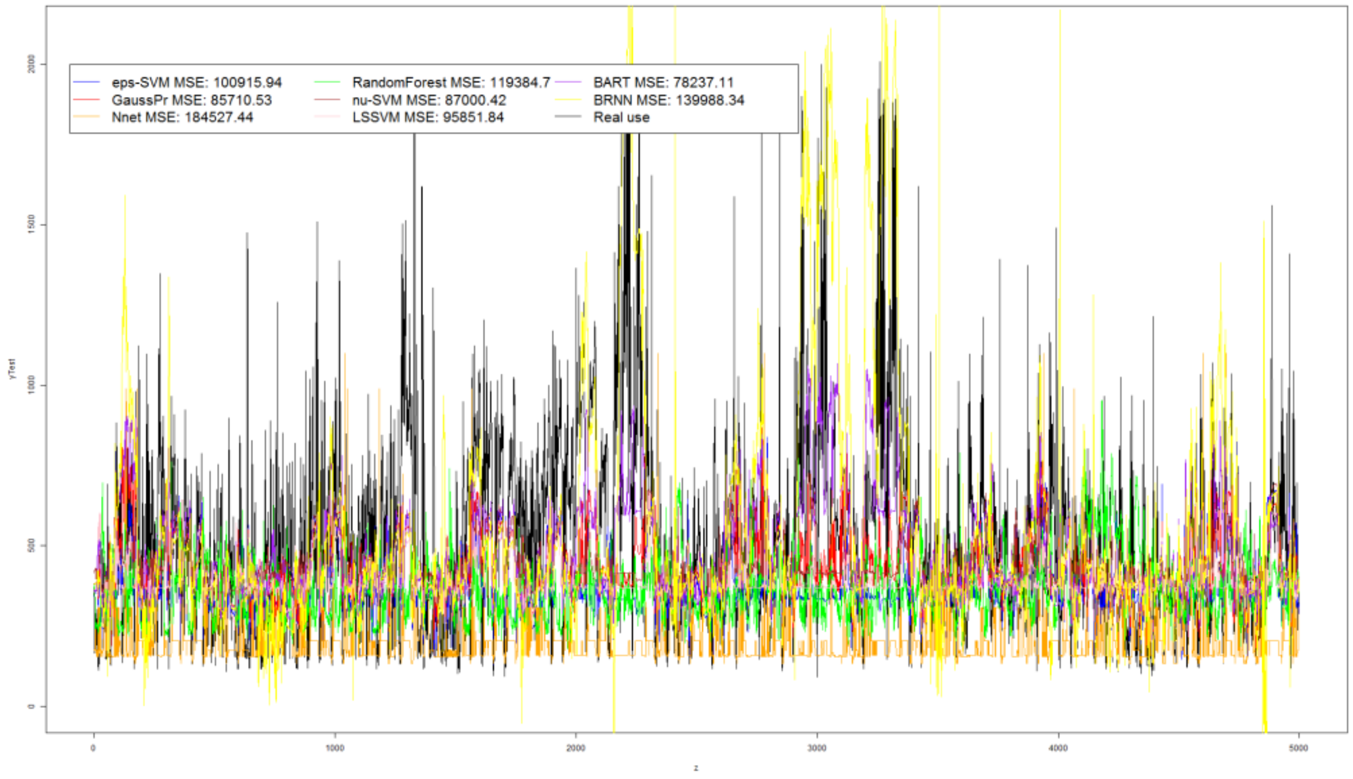

Figure 3: The graph depicts energy consumption: real (black line) and predicted by 8 used algorithms. The power consumption in Watt-hours is on Y-axis, on X-axis we have time with 15 minutes step.

For more clarity of chart we used *xts* package to group 15 minutes step outputs of sensors into days. It generated Figure 4.We can see that it confirms statements based on Figure 3. Furthermore we can observed that Random Forest algorithm (green) has interesting tendency, when real usage consumption peaks Random Forest predicts low values and when real usage reaches minimum then Random Forest peaks. This anomaly can be seen between February and March. Moreover it shows similar runs of SVM-family methods.

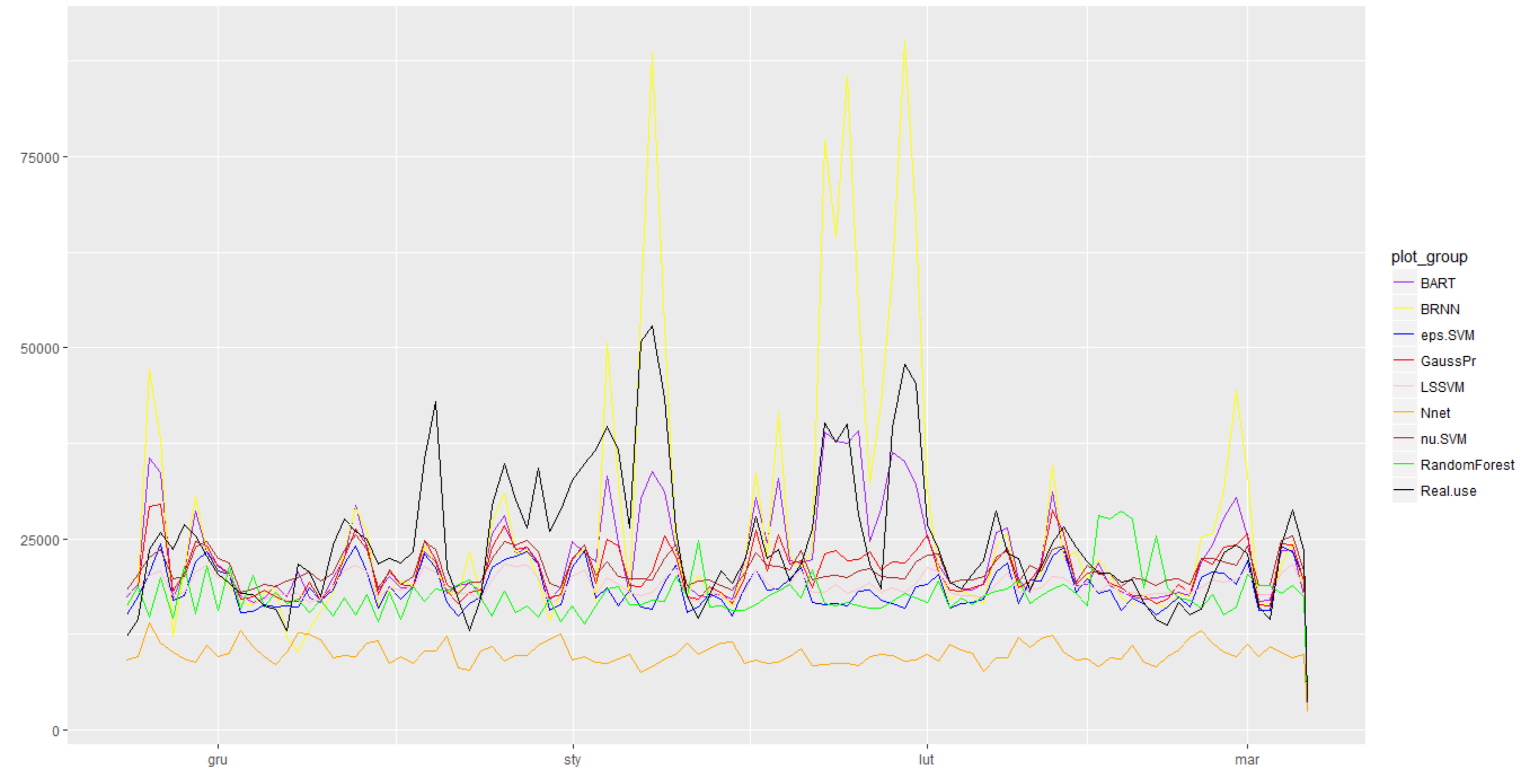

Figure 4: This graph presents the same energy consumption data as graph from Figure 3 – actual (black line) and predicted by 8 used algorithms, but this time data is grouped by day. Consequently on Y-axis we have daily power consumption in Watt-hours and on X-axis time with monthly labels.

Table 1 presents times of generating model for each method. The best was SVM nu regression algorithm with average of 5 seconds. Next there were EPS svm regression about 12 and 20 seconds. The worst was Feed Forward Neural Network, it took over an hour to generate its classifier.

Table 2 shows times of predicting phase for each method. The best score obtained Bayesian Regularized Neural Networks, prediction under one second also achieved Feed Forward Neural Network. The worst was Bayesian Additive Regression Trees with average about 20 seconds.

Measurement	Gauss	Eps-svm	Nu-svm	LSSVM	Nnet	RF	BART	BRNN
1	383.86	12.96	5.17	46.51	3835.81	23.63	282.18	30.8
2	371.4	12.92	4.89	52.04	3776.37	18.48	164.11	29.51
3	352.26	9.93	4.98	45.35	3777.55	18.97	162.14	29.04
4	378.41	12.83	2.22	46.78	3024.64	17.8	162.53	28.89
5	369.06	12.97	4.9	45.89	3871.75	19.26	210.1	34.35
6	408.5	11.81	5.02	43.21	3928.06	18.74	159.79	29.05
7	254.36	4.67	5.55	41.53	4246.39	21.46	159.11	29.39
8	397.89	13.61	1.77	39.41	3771.94	12.21	163.98	29.56
9	281.66	12.76	5.26	45.95	4053.45	20.74	161.58	29.63
10	475.98	15.03	6.25	46.11	3896.9	25.41	161.1	29.55
AVG	367.34	11.95	4.60	45.28	3818.29	19.67	178.66	29.98
STD DEV	62.52	2.86	1.44	3.40	315.42	3.58	39.45	1.62

Table 1: Time measurement in seconds for generating model of each model.

Measurement	Gauss	Eps-svm	Nu-svm	LSSVM	Nnet	RF	BART	BRNN
1	5,78	5,11	1,58	2,39	0,82		27,31	0,01
2	5,78	5,22	1,52	4,42	0,86		18,61	0,01
3	5,82	3,28	1,53	2,16	0,83		18,62	0,02
4	5,71	5,08	1,57	2,48	0,83		18,53	0,02
5	5,89	5,13	1,52	2,95	0,78		25,53	0,02
6	2,13	5,5	1,49	2,32	0,92		18,55	0,02
7	2,17	1,93	1,64	2,31	0,89		18,91	0,01
8	6,01	5,34	0,51	2,79	0,82		18,5	0,01
9	5,42	5,03	1,59	3,39	0,92		18,63	0,02
10	6,67	7,33	1,75	2,64	0,86		18,75	0,01
AVG	5,14	4,90	1,47	2,79	0,85		20,19	0,02
STD DEV	1,61	1,42	0,35	0,68	0,05		3,31	0,01

Table 2: Time measurement in seconds for predicting power consumption by each model except Random Forest where prediction phase is contained in generating model.

Method	Gauss	Eps-svm	Nu-svm	LSSVM	Nnet	RF	BART	BRNN
MSE	87218,15	100915,9	87000,42	87182,2	161302,6	119832	76291,92	139988,3

Table 3: Mean squared error values for each method. The lower value of estimator, the better algorithm predicted energy usage.

	Gauss	Eps-svm	Nu-svm	LSSVM	Nnet	RF	BART	BRNN
Gauss	x	n	n	n	n	n	n	n
Eps-svr		x	y	n	n	y	n	y
Nu-svr			x	n	n	n	n	n
LSVM				x	n	n	n	y
Nnet					x	n	n	n
RF						x	n	y
BART							x	n
BRNN								x

Table 4: T-student matrix. Y - null hypothesis that the data in x − y comes from a normal distribution with mean equal to zero and unknown variance is not true, n - is true

Table 4 shows comparisons of two-sample t-student test for algorithms execution time. A two-sample location test of the null hypothesis such that the means of two populations are equal. All such tests are usually called Student's t-tests, though strictly speaking that name should only be used if the variances of the two populations are also assumed to be equal; the form of the test used when this assumption is dropped is sometimes called Welch's t-test. These tests are often referred to as "unpaired" or "independent samples" t-tests, as they are typically applied when the statistical units underlying the two samples being compared are non-overlapping.

6 Conclusion

In this paper we described some popular machine learning algorithms and use them for predicting power consumption. The best score achieved Bayesian Additive Regression Trees. Results under 100000 obtained also Gaussian Processes, SVM nu regression and LS-SVM. Neural Networks family methods disappointed with low score plus computing of Feed Forward Neural Network lasted too long. In term of quality for time ratio the best was SVM nu regression with second score. Entire computation took average about only 6 seconds for this method.

References

[1] Richard E. Edwards, Joshua New, Lynne E. Parker Predicting future hourly residential electrical consumption: A machine learning case study *Elsevier Energy and Buildings*

[2] Long-term energy/environment data for ORNL Research House # 3

[3] Hugh A. Chipman, Edward I. George, Robert E. McCulloch BART: Bayesian Additive Regression Trees

[4] A. Kapelner, J. Bleich bartMachine: Machine Learning with Bayesian Additive Regression Trees

[5] D. J. C. MacKay A practical Bayesian framework for backpropagation networks.

[6] P. Pérez-Rodríguez, D. Gianola An R package for fitting Bayesian regularized neural networks with applications in animal breeding

[7] Simon Haykin FeedForward Neural Networks: An Introduction

[8] Brian Ripley, William Venables Feed-Forward Neural Networks and Multinomial Log-Linear Models

[9] Leo Breiman Random Forests. Statistics Department University of California Berkeley

[10] Breiman and Cutler's Random Forests for Classification and Regression

[11] https://www.rdocumentation.org/packages/kernlab/versions/0.9-25/topics/ksvm

[12] http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

[13] http://www.esat.kuleuven.be/sista/natoasi/suykens.pdf

[14] https://en.wikipedia.org/wiki/Student