Modeling and Estimating of Load Demand of Electricity Generated from Hydroelectric Power Plants in Turkey using Machine Learning Methods

Bahtiyar DURSUN[1], Fatih AYDIN[2], Metin ZONTUL[3], Selcuk SENER[4]
[1]*Faculty of Technology, Kirklareli University, Kirklareli, 39060, Turkey*
[2]*Faculty of Engineering, Kirklareli University, Kirklareli, 39000, Turkey*
[3]*Faculty of Engineering, Istanbul Aydin University, Istanbul, 34153, Turkey*
[4]*Vocational School of Anadolu Bil, Istanbul Aydin University, Istanbul, 34153, Turkey*
bahtiyar.dursun@klu.edu.tr, fatih.aydin@klu.edu.tr, metinzontul@aydin.edu.tr, selcuksener@aydin.edu.tr

***Abstract*—In this study, the electricity load demand, between 2012 and 2021, has been estimated using the load demand of the electricity generated from hydroelectric power plants in Turkey between 1970 and 2011. Among machine learning algorithms, Multilayer Perceptron, Locally Weighted Learning, Additive Regression, M5Rules and ZeroR classifiers are used to estimate the electricity load demand. Among them, M5Rules and Multilayer Perceptron classifiers are observed to have better performance than the others. ZeroR classifier is a kind of majority classifier used to compare the performances of other classifiers. Locally Weighted Learning and Additive Regression classifiers are Meta classifiers. In the training period conducted by Locally Weighted Learning and Additive Regression classifiers, when Multilayer Perceptron and M5Rules classifiers are chosen respectively, it is possible to obtain models with the highest performance. As a result of the experiments performed using M5Rules and Multilayer Perceptron classifiers, correlation coefficient values of 0.948 and 0.9933 are obtained respectively. And, Mean Absolute Error and Root Mean Squared Error value of Multilayer Perceptron classifier are closer to zero than that of M5Rules classifier. Therefore, it can be said the model performed by Multilayer Perceptron classifier has the best performance compared to the models of other classifiers.**

***Index Terms*—electricity load forecasting, machine learning, multilayer perceptron, rule based learning, time series prediction.**

I. INTRODUCTION

Accurate models for electric power load forecasting are essential to the operation and planning of a utility company. Load forecasting helps an electric utility to make important decisions including decisions on purchasing and generating electric power, load switching, and infrastructure development. Load forecasts are extremely important for energy suppliers, ISOs, financial institutions, and other participants in electric energy generation, transmission, distribution, and markets. Load forecasts can be divided into three categories: short-term forecasts which are usually from one hour to one week, medium forecasts which are usually from a week to a year, and long-term forecasts which are longer than a year. Accurate and robust load forecasting is of great importance for power system operation. It is the basis of economic dispatch, hydro-thermal coordination, unit commitment, transaction evaluation, and system security analysis among other functions. Because of its importance, load forecasting has been extensively researched and a large number of models were proposed during the past several decades, such as Box-Jenkins models, ARIMA models, Kalman filtering models, and the spectral expansion techniques-based models. Generally, the models are based on statistical methods and work well under normal conditions, however, they show some deficiency in the presence of an abrupt change in environmental or sociological variables which are believed to affect load patterns. Also, the employed techniques for those models use a large number of complex relationships, require a long computational time, and may result in numerical instabilities. Therefore, some new forecasting models were introduced recently. As a result of the development of Artificial Intelligence (AI), Expert System (ES), Machine Learning (ML) and Artificial Neural Networks (ANN) have been applied to solve the STLF problems [1].

Machine learning is a field of study that aims to give computers the ability to learn without programming explicitly [2]. The key difference between machine learning and traditional artificial intelligence is to make the system gain intelligent behaviors without explicit programming but learning from the data. During Knowledge Discovery, machine learning is widely used for the implementation of induction algorithms which is one of the steps of Knowledge Discovery [3]. Inductive machine learning algorithms can learn patterns from labeled data which has a known outcome [4]. In this study, pattern discovery is performed using labeled data. So, unknown outcome of a test data can be estimated.

There are many studies about machine learning techniques to forecast load. Some of them are summarized below. Negnevitsky and his colleagues evaluated main forecasting techniques used for power system applications [5]. Available forecasting techniques have been discussed with focus on electricity load and price forecasting as well as wind power prediction. Forecasting problems have been classified based on time frame, application specific area and forecasting techniques.

Digital Object Identifier 10.4316/AECE.2014.01019

Appropriate examples based on data pertaining to the Victorian electricity market, Australia and the PJM electricity market, U.S.A. are used to demonstrate the functioning of the developed neural network (NN) method based on similar days approach to predict hourly electricity load and price, respectively [5]. Fan and his colleagues proposed a novel forecasting model for day ahead electricity load forecasting. The proposed model adopts an integrated architecture to handle the non-stationary of time series. They used two machine learning techniques: Bayesian clustering by dynamics (BCD) and support vector regression (SVR). The effectiveness of the proposed model is demonstrated with actual data taken from the New York ISO and the Western Farmers Electric Cooperative in Oklahoma. [6]. Ying-Chun Guo and his colleagues proposed a model of support vector machine for forecasting electricity load. The model overcomes the disadvantages of general artificial neural network (ANN), for instance, it is not easy to converge, liable to trap in partial minimum and unable to optimize globally, and the generalization of the model is not good, etc. The SVM model makes sure that the forecasting is optimized globally. Subsequently, examples of electricity load data from Hebei province of China are used to illustrate the performance of the proposed model [7]. Swief and his colleagues proposed a new algorithm to predict both load and price values. A machine learning techniques such as Principle Component Analysis, and k nearest neighbor points, are applied as pre-processing techniques to assist the Support Vector Machine. The proposed model is implemented to build a closed loop dynamic model to forecast both 24 hours a head short term load and price values, which is a common study in markets. The model has been trained and validated using data from, Australia electricity market. Moreover, the model has been tested using data from North England electricity market [8].

One of the most widely used theories in the field of machine learning is "no free lunch" [31], [32]. The starting point of this theory is the principle that there is not only one single algorithm which can solve all the real world problems. Therefore, different algorithms may be superior to one another in terms of their performances when modeling a real world problem. In this study, accordingly, machine learning algorithms with different approaches were selected and compared with each other. The method followed in this study is seen in other machine learning studies, as well. In the estimation of "initial public offering (IPO)" made by Luque et al. [33] common machine learning algorithms are compared to each other and the algorithm that increases the system performance is selected. That is to say, it is tried to determine the algorithm that increases the performance of the system for this real world problem. In addition, in "Free Space Optical (FSO)" classification problem made by Prakash et al. [34], the statistical learning approach which is also one of the machine learning approaches, is used.

RBFNetwork is used as the most convenient algorithm for the classification of FSO. In another study made by Martišius et al. [35] real-time learning of EEG signals, which is an important real world problem, is performed. In this study, too, another machine learning algorithm Voted Perceptron is used. In addition, in order to reduce the training time in real-time applications, a modified version of Voted Perceptron algorithm is provided. Besides, Ismail et al. [36] introduced a new procedure to determine an optimum activation function for an artificial neural network in their study. In artificial neural networks, several models can be performed by altering the parameters. When the number of the parameters and the values they obtain are considered, it is seen that a large number of different models are possible. Therefore, it is very important to have a method in order to define the optimum activation function. As a result, implementing a strict procedure is wrong when determining Machine Learning algorithms and their methods for the solutions of real world problems. In light of all the above-mentioned studies, an intuitive, experimental and flexible procedure is used when modeling the electricity load demand generated in hydroelectric power plants.

In this study, using the load demand data of electricity, which is generated by hydroelectric power plants between 1970 and 2011, obtained from TEIAS, the electricity load demand between 2012 and 2021 will be estimated by machine learning methods. Machine learning classifiers that are used in experiments are Multilayer Perceptron, Locally Weighted Learning, Additive Regression, M5Rules and ZeroR. In addition, performances of these classifiers will be compared in this study. At the end of the study, the most appropriate classifier will be selected and the electricity load demand of Turkey between the years 2012 and 2021 will be estimated.

II. Classification Algorithms Used in the Experiments

In this study, Turkey's electricity load profile is forecasted using the last electricity load demand data. It is determined to use five different learning algorithms by the software called Weka software. Weka (Waikato Environment for Knowledge Analysis) is a popular suite of machine learning software written in Java, developed at the University of Waikato. WEKA is free software available under the GNU General Public License. All data are analyzed by using Weka software with ten-fold cross-validation [9] strategy for five different learning algorithms. These algorithms are shown in Table I.

TABLE I. CLASSIFIERS USED IN THE EXPERIMENTS

No	Classifiers	Learning Type
1	Multilayer Perceptron	Function Learning
2	Locally Weighted Learning	Lazy Learning
3	Additive Regression	Meta Learning
4	M5Rules	Rule-based Learning
5	ZeroR	Rule-based Learning

At the end of the tests, ZeroR majority classifier, which is one of the basic classifiers, was compared to four different classifiers that have different learning approaches and best learning experiences. Classification results were obtained using Weka 3.6.0 stable version x64.

A. Multilayer perceptron

The Multilayer Perceptron used in Weka is a feed forwarding and back propagated neural network system which is connected with many units with weighted links. The units as known as Neurons are consisted of some layers. These layers are; input layer, one or more hidden layer(s) and finally the output layer.

There is a sample topology for Multilayer perceptron in Fig. 1. At this Figure, the input layer takes an external activation vector and passes the values computed with the weighted links to the first hidden layer. This situation is shown for the first neuron in the Eq. (1). If there are other hidden layer(s) in the network, the calculations between hidden layers are computed as in the previous process, and the calculated values are passed to the next hidden layer. This situation is shown for the first neuron in the Eq. (2). Finally, the output layer is computed with the data coming from the hidden layer(s). This situation is shown in the Eq. (3). There is an activation function used for the computing process of all the units. In Weka, Exact Sigmoid Function is used as an activation function. In Eq. (4) Sigmoid Function is shown. If the classes are numeric, the output node becomes an unthresholded linear unit.

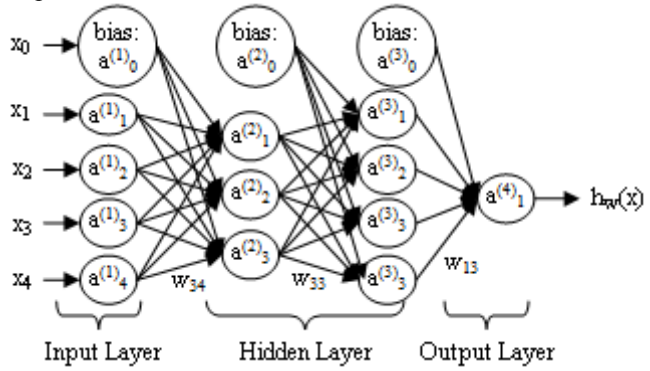

Figure 1. A typical feed forward neural network with two hidden layer

$$a_i^{(k+1)} = g\left(\sum_{j=0}^{n} w_{ij}^{(k)} x_j\right) \tag{1}$$

$$a_i^{(k+1)} = g\left(\sum_{j=0}^{n} w_{ij}^{(k)} a_j^{(k)}\right) \tag{2}$$

$$h_w(x) = a_i^{(k+1)} = g\left(\sum_{j=0}^{n} w_{ij}^{(k)} a_j^{(k)}\right) \tag{3}$$

$$g(x) = \frac{1}{1+e^{-x}} \tag{4}$$

The computation needed for the neuron value is shown in the Eq. (5). This Eq. (5) shows a general expression for the computation of the neuron value. In this expression, the g(.) function shows the activation function, and also shows the neuron number in the layer before the layer of k value.

$$a_j^{(i)} = g\left(\sum_{k=0}^{m} w_{jk}^{(i-1)} a_k^{(i-1)}\right) \tag{5}$$

In the multilayer perceptron, the main structure of the network is identified with the experiments. Also, the experience of an expert is important in the identification process. If the units' weight outside the network needs to be changed, it is only possible with the use of some mathematical algorithms. The method used by Multilayer Perceptron in Weka is Gradient Descent [10]. With this method, it is possible to reduce the value between the prediction and the actual value. This method is also known as squared error loss function. This fact is shown in Eq. (6). In Eq. (6), m shows the value of training data. J(w) function is named as cost function. To employ the Gradient Descent, the general expressions in Eq. (7) should come true. In Eq. (7), cost function is partially derivative in regards of weight vector, and this value is multiplied with learning rate. The result is subtracted from the weight vector and the update is completed.

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_w\left(x^{(i)}\right) - y^{(i)}\right)^2 \tag{6}$$

$$w_{ij}^{(l)} := w_{ij}^{(l)} - \alpha \frac{\partial}{\partial w_{ij}} J(w) \tag{7}$$

In the Eq. (7), learning rate is used to update the weights. It is aimed for J(w) cost function to take a minimum value that way, but the local minimum convergence of the cost function happens very slow. Because of this, momentum value is used as a trick, and the slowness is prevented.

B. Locally weighted learning

Locally Weighted Learning (LWL) classifier is an algorithm developed to resolve the structural problems such as attribute independence and low performance algorithms that are the weaknesses of NaiveBayes classifier [12], [13], [14], [15]. Assumption of Bayes Theorem, which considers the input attribute statistically independent according to output class, is practically impossible. However, Bayes Theorem shows an astonishing success over many classification problems in the field of machine learning.

The probabilistic model for a NaiveBayes classifier is a conditional model. For this probabilistic model, having $F_1, F_2, ..., F_n$ as feature variables and C as dependent variable of outcome or class, the conditional model is calculated as in Eq. (8) by using Bayes Theorem.

$$p(C|F_1,\ldots,F_n) = \frac{p(C).p(F_1,\ldots,F_n|C)}{p(F_1,\ldots,F_n)} \tag{8}$$

When the attributes have discrete and continuous values, having x as continuous value; the data firstly divided into classes. Then, means and variance of x in each class is calculated. Finally, the probability values from the class C = c is calculated as shown in Eq. (9) for a normal distribution X = x.

$$p(X = x|C = c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{\frac{(x-\mu_c)^2}{2\sigma_c^2}} \tag{9}$$

LWL classifier is a lazy classifier just like NaiveBayes classifier. Like all other lazy classifiers, LWL classifier, as well, stores the training data and does not create a model for learning until classification time. When a new instance is going to be classified, a new NaiveBayes model is constructed by using the set of weighted training instances in the locale of test instance.

LWL classifier uses Euclidean distance to calculate the weights. Let d_i be assumed as the distance to the closest neighbor x_i, and all the attributes assumed to be normalized between 0 and 1 before the distances are calculated. In addition, let the nominal attributes be binarized. Accordingly, let f(.) function be a weighting function which is f(y)=0 for all $y \geq 1$. As a result, w_i weight for each xi is calculated as shown in Eq. (10). In addition, f(.) function

that is decreasing monotonically is shown in Eq. (11). This is a linear weighting function.

$$w_i = f\left(d_i / d_k\right) \tag{10}$$

$$f_{linear}(y) = 1 - y \qquad for\ y \in [0,1] \tag{11}$$

LWL classifier uses Laplace estimator when estimating the conditional probability to avoid zero-frequency problem (for nominal attributes) through weighting process. Since the weight of total instances, which are used to generate NaiveBayes model, is approximately k, weights can be scaled in LWL classifier. Accordingly, if the data in xi training instance that proves $d^i \leq d_k$ condition, then the rescaled w_i weights are computed as shown Eq. (12). n value in Eq. (12) is the total value of training instances.

$$w_i' = \frac{w_i . r}{\sum_{q=0}^{n} w_q} \tag{12}$$

Local learning helps to mitigate the effect of attribute dependency which is present in data as a whole. This kind of design structure provides a high performance unless there is a strong dependency between attributes during the process of assigning a test instance to a class, since NaiveBayes classifier requires little data in training process when the neighborhood of test instances is low. Accordingly, the possibility of confrontation with strong dependencies becomes low. LWL classifier, on the other hand, allows selecting the test instance as data-dependent based on the distance of k.th nearest neighbor. So, it brings an alternative for k.th nearest neighbor algorithm, since, k value affects the variance of the classifier. Moreover, fine-tuned k value provides good results.

The equation which is used to compute the conditional probability for the data whose weights are calculated is shown in Eq. (14). In Eq. (14), if x = y, I(x = y) specifier is going to be 1. Otherwise it is going to be 0. In addition, o represents the total number of classes, m represents the number of attributes and a represents the attributes.

$$p(c_1) = \frac{1 + \sum_{i=0}^{m} I(c_i = c_1) w_i'}{o + \sum_{i=0}^{m} w_i'} \tag{13}$$

$$p(a_j | c_1) = \frac{1 + \sum_{i=0}^{m} I(a_j = a_{ij}) I(c_i = c_1) w_i'}{m_i + \sum_{i=0}^{m} I(a_j = a_{ij}) w_i'} \tag{14}$$

If the data contains a numeric value, to estimate the mean and the variance, a normality assumption is done [14] based on weighted data or Fayad and Irani's [16] MDL based discretization scheme. As a result, LWL classifier has taken NaiveBayes classifier one step further.

C. Additive regression

Additive Regression (AR) classifier is an ensemble classifier, because of this it uses a base classifier. This basic classifier classifies the training data. AR classifier starts with an empty ensemble at the start of the program. Later, it adds new members to the incorporate sequentially in the learning phase. This addition phase is not done randomly. Only the models which make the guessing performance maximum are added to the ensemble. To optimize the performance of the ensemble, created model by the next member focuses on the poor training instances. The ensembles of the education data created by the AR classifier are shown at Eq. (16). In Eq. (16), function f is a member of the ensemble. These members also correspond to the models created by the basic classifier. α expression at Eq. (15) corresponds to a fixed value. ε expression at Eq. (16) expresses the mistakes done by the ensemble.

$$Y = \alpha + f_1(X_1) + f_2(X_2) + \cdots + f_n(X_n) + \varepsilon \tag{15}$$

$$Y = \alpha + \sum_{i=1}^{n} f_i(X_i) + \varepsilon \tag{16}$$

AR classifier uses a standard regression model for numerical guesses for the classifier [10]. This regression model identifies the problems at the education data. The error value is the difference between the actual and the prediction value. Later, a second model is created to correct these errors.

D. M5Rules

M5 Rules classifier is an algorithm developed by Quinlan which creates rules from the tree based model [17] .The models created by M5 are multivariate linear models. M5 classifier is able to make rules out of very high dimensionality training data with hundreds of attributes. As the size of the data gets bigger, the cost of the estimations increases in a very fast way. The model trees created by M5 classifier are generally a lot smaller than the regression trees.

While the initial tree is constructed in M5 Classifier, the split criteria of tree branches are determined by calculating the standard deviations of class values reaching a node as a measure of node errors and by calculating expected reduction in error as a result of each attribute tested in nodes [18]. Then, the attribute with the maximum error reduction is selected. The reduction in the standard deviation is calculated as in the Eq. (17). In Eq. (17) the expression of T expresses the education cluster. This training data is later divided as T_1, T_2, …, T_i when the nodes separate.

$$SDR = sd(T) - \sum_i \frac{|T_i|}{|T|} sd(T_i) \tag{17}$$

In the second stage, there is the pruning of the tree process. At first, the average values of the absolute difference between the predicted value and the actual class value are calculated for each training data in this stage. These average values estimate the expected errors for the unforeseen classes. In order to compensate the error, the calculated average values are multiplied with the factor of (n + v) / (n - v). In this factor, the expression of n expresses the number of training data that reaches to the nodes and the expression of v expresses the parameter values of the class values in the nodes.

The last stage of modeling process of the M5 classifier is the smoothing process. At first, the sharp discontinuity between the adjacent linear models of the pruned tree is determined and smoothed. The leaf model is used to compute the estimated value in the process. The values are filtered on the path to the root. The smoothing process is done with the combination of the leaves and the estimated

value by the linear models for the nodes. The necessary calculation for this process is shown in the Eq. (18). In Eq. (18), p' represents the estimated pass up value, p represents the estimated pass value, q represents the estimated value of the model in the node. n represents the number of education examples and k represents a fixed value.

$$p' = \frac{np + kq}{n + k} \tag{18}$$

Later, with the usage of tree model created by the M5 classifier, the rules are extracted. This situation shows that M5 is a classifier which is capable of extracting new information from the model [18].

E. ZeroR

The classifier which is known as "0-rules" is called as ZeroR classifier in WEKA. ZeroR that is also known as majority classifier is mainly used as base classifier to measure the performance of other classifiers. That is to say, regarding the performance criteria of the classifiers, it is very important for all the other classifiers to have a greater accuracy rate than ZeroR. ZeroR classifier operates as follows; the class with the highest frequency among training data is assumed as the output value for all the data. So, the rate of the class with the highest frequency gives the approximate accuracy rate [18].

III. Feature Selection

Feature selection is one of the crucial steps of pattern recognition and artificial intelligence problems [19]. One of the factors affecting considerably the decision making processes of Machine Learning algorithms is whether the qualities are suitable. There are two basic approaches to choose a good attribute subset. The first is an independent evaluation depending upon data's general characteristics. Second one is to evaluate the attribute subset by using Machine Learning algorithm used for learning function. The first approach is called filter method, because the current attribute subset is filtered to make attribute subsets affecting the result positively before the learning process. The second approach is called wrapper method. The results of learning algorithm are observed during the feature selection process with this method [20]. The qualities increasing the performance of algorithm are added to quality universe after this observation.

In this study, it is created a quality universe of training subset by using the feature selection method mentioned above. Filter method approach was firstly used for feature selection. The attribute set chosen considering this approach is just the year feature. Training data is prepared by using this introduction feature. Next step is wrapper approach. In this approach, training data is learnt by Machine Learning algorithm used in classification and the results are examined. At the end of the process, the classifiers used in the experiments are examined whether they learned well concerning the training criteria of the classifiers' training data. If the learning process is enough, the chosen introduction feature will not be changed. Otherwise, it will be changed and the process continues in this way.

IV. Evaluation Criteria of Learning Algorithms

A variety of criteria is needed to decide how successful a classifier is at the end of a learning process. The criterion used to measure the success of the learning system that is the subject of this study are as follows: Correlation Coefficient, Mean Absolute Error, Square Root of Mean Square Error, and Bias-Variance Decomposition. K-fold cross validation method was applied in all experiments to guarantee the certainty of the results of the experiments.

A. Correlation coefficient

Correlation coefficient (CC) is a coefficient which indicates the relationship between independent variables of power and direction. It changes between -1 and +1. The positive values of Correlation coefficient specify a direct directional relation, and the negative values specify a reverse directional relation. If Correlation coefficient is zero it indicates that there isn't any relationship between variables. Detailed explanation for Correlation coefficient value is shown in Table II given by Cohen [21].

TABLE II. COHEN'S CORRELATION TABLE

Correlation	Negative	Positive
Low	-0,29 to -0,10	0,10 to 0,29
Middle	-0,49 to -0,30	0,30 to 0,49
High	-0,50 to -1,00	0,50 to 1,00

Eq. (19) is the correlation coefficient, which measures the statistical correlation between the a's and the p's.

$$CC = \frac{S_{PA}}{\sqrt{S_P S_A}} \tag{19}$$

$$S_{PA} = \frac{\sum_i (p_i - \bar{p})(a_i - \bar{a})}{n-1} \tag{20}$$

$$S_P = \frac{\sum_i (p_i - \bar{p})^2}{n-1} \tag{21}$$

$$S_A = \frac{\sum_i (a_i - \bar{a})^2}{n-1} \tag{22}$$

B. Mean Absolute Error

Mean absolute error (MAE) is an alternative: just average the magnitudes of the individual errors without taking account of their sign [20].

MAE metric is used when the variance between two values is important. MAE formula is shown in Eq. (23).

$$MAE = \frac{|p_1 - a_1| + \cdots + |p_n - a_n|}{n} \tag{23}$$

C. Root Mean Squared Error

Root Mean Squared Error is calculated by taking the square root of Mean Squared Error. Mean Squared Error (MSE) is a common used metric. A lot of mathematical techniques (such as linear regression) use MSE since it is easily implemented. It is also used to evaluate the performance of classifiers in Machine Learning [20].

Error rate of an estimator occurs because of an arbitrary estimation or because of a lack of accurate information [22]. If MSE and RMSE values are close to zero, error rate

becomes the minimum. In addition, for each learning process, acceptable error rates of MSE or RMSE are different.

MSE and RMSE calculations are shown in Eq. (24) and Eq. (25). Accordingly, p represents predicted value; a represents actual value.

$$MSE = \frac{(p_1 - a_1)^2 + \cdots + (p_n - a_n)^2}{n} \quad (24)$$

$$RMSE = \sqrt{\frac{(p_1 - a_1)^2 + \cdots + (p_n - a_n)^2}{n}} \quad (25)$$

D. Bias Variance Decomposition

Bias-Variance decomposition is a key tool to understand Machine Learning algorithms. In recent years, use of Bias-Variance in experimental studies is gradually increasing. Bias and variance concepts help to explain how simple estimators are superior to complex ones and how model sets are superior to simple models. Among other statistical error functions, Bias-Variance decomposition is also created for quadratic loss [23]. This creation is shown in a study made by Geman, Bienenstock and Doursat [24]. In this study, mean squared error of machine learning is stated as the sum of variance and square of bias.

Having a as actual value and p as predicted value for each instance, presentation of squared error as bias-variance decomposition is shown in Eq. (26). When the noise is omitted, mean squared error becomes Eq. (27).

$$E\left[(p_i - a_i)^2\right] = Var(Noise) + Bias^2 + Var(p_i) \quad (26)$$

$$MSE = Var(p) + Bias^2 \quad (27)$$

E. Cross Validation

There are different ways to determine whether the training data is sufficient enough to be learned by a classifier or not. One of these ways is k-fold Cross Validation. k-fold Cross Validation (k-fold CV) provides a method to evaluate the accuracy of a classifier by the division of the data into k numbered equal parts. So, the classifier is tested and trained k times. So, the classifier is always trained k-1 times [3]. That is to say, k-1 amount is used for training among k sub-part. Remaining folds are used as training data [25]. Accuracy estimation of the classifier is average accuracy for k numbered k-folds [3].

Since there is not enough test data to measure the generalization ability of the models produced by the classifiers, cross validation method can be used. That is to say, if the dataset is not enough to be separated as training set, test set and validation set, then CV method is used. CV method enables us to evaluate the generalization ability of the model over uncalculated data. In short, CV is a model validation method. In this respect, the studies of Kohavi [26] and Lendasse et al. [27] show that CV method can be used as an important tool to test the generalization ability of the models.

Using k-fold CV, data limitations can be made for training data [20]. So, it is possible to sort out too much data from training data. Thus, it will provide high performance for Memory Based Learning algorithms such as kNN. Using k-fold CV method, change of bias-variance or underfitting-overfitting can be balanced [28].

k-fold CV method is high-cost in terms of calculation. However, it is a very useful method when estimating the error rate of classifier [29]. Moreover, when comparing two different learning algorithms with limited data, k-fold CV is also used [28]. An important problem for k-fold CV is the unbalance between the instance classes spread over training set and test sets [20]. Because of this class unbalance, the classifier cannot perform a complete learning. Therefore, error rates of the classifier may become high. In order to solve this problem, each class should be evenly distributed between test and training sets. This ensures a solution to the problem and it is called "stratification". Stratification provides a precaution against unbalanced representation in test and training sets [20]. Weka implements stratification process during k-fold CV method. This, also, provides good results [30].

It is clear that 10 is the most accurate value to estimate the error rate most accurately for common tests over various data sets by using different learning methods. That is why 10 is chosen as the value of k. Theoretical proofs also support this idea [20].

When evaluating the performance of Machine Learning classifiers, Leave-one-out cross-validation (LOOCV), which is the customized form of k-fold CV method, is used occasionally. This method is also based on using only one observation instance from original instance as validation data. That is to say, one instance among original instances becomes the validation instance while the remaining instances are used as training instances. This method is repeated until all the instances are used as validation instance. The method is particularly used to show the difference between the models that are produced by the classifiers. If the models generate highly different results in each case, then the original instances should be reviewed, since there is the possibility of noisy data.

V. Classification Results and Discussions

In this study, the electricity load demand, between 2012 and 2021, is attempted to be estimated by using 42 years' load demand data of the electricity generated from hydroelectric power plants in Turkey between 1970 and 2011. These values are shown in Table III. Four Machine Learning classifiers and one base-classifier are used for these estimations. ZeroR majority classifier is used as base-classifier. Other four classifiers are supposed to have a better generalization of training data compared to the model produced by ZeroR majority classifier. The best way to realize this is to use correlation coefficient and error criterion. Since there is not sufficient data set to evaluate the generalization ability of the model produced by the classifiers, cross validation is used as the method.

TABLE III. ELECTRICITY LOAD OF TURKEY BETWEEN 1970-2011

No	Year	Electricity Load (MW)
1	1970	725.4
2	1971	871.6
3	1972	892.6
4	1973	985.4
5	1974	1449.2
6	1975	1779.6
7	1976	1872.6
8	1977	1872.6
9	1978	1880.8
10	1979	2130.8
11	1980	2130.8
12	1981	2356.3
13	1982	3082.3

14	1983	3239.3
15	1984	3874.8
16	1985	3874.8
17	1986	3877.5
18	1987	5003.3
19	1988	6218.3
20	1989	6597.3
21	1990	6764.3
22	1991	7113.8
23	1992	8378.7
24	1993	9681.7
25	1994	9864.6
26	1995	9862.8
27	1996	9934.8
28	1997	10102.6
29	1998	10306.5
30	1999	10537.2
31	2000	11175.2
32	2001	11672.9
33	2002	12240.9
34	2003	12578.7
35	2004	12645.4
36	2005	12906.1
37	2006	13062.7
38	2007	13394.9
39	2008	13828.7
40	2009	14553.4
41	2010	15831.2
42	2011	17137.1

It is necessary to compare the performances of other classifiers used in experiments with regard to ZeroR classifier. Performance rates of ZeroR classifier at the end of the training are shown in Table IV. According to these results, ZeroR composed a weak model because CC value is below zero. MAE and RMSE values are far above zero, as well. In addition, it does not improve the generalization ability to have different values, other than 10, for k value. So, it is clear that cross validation is a reliable method to test the model.

TABLE IV. PERFORMANCE VALUES OF ZEROR CLASSIFIER

k-Value	CC	MAE	RMSE
2	-0.2140	4684.8213	5226.7903
5	-0.3375	4569.0112	5060.4924
10	-0.4388	4525.2142	5008.0290
15	-0.7706	4646.0769	5111.4931
20	-0.8475	4599.9721	5082.7230
25	-0.8275	4565.9727	5052.6861
30	-0.8878	4550.9579	5043.0578
35	-0.9308	4537.6448	5024.7386
42	-1	4531.3057	5020.6019

When we analyze the results from Table IV, we see that k value with least error is 10. Besides, CC value when k = 10, is the approximate average value of all other CC values. This shows that when evaluating the performance of classifiers, not only the CC value, but also MAE and RMSE values should also be considered. It is inadequate to consider only CC value because MAE and RMSE can have the highest values while CC has the lowest value. Yet, MAE and RMSE values should be very close to zero in order to have the lowest error rate.

ZeroR classifier's predictions when k = 0 and actual values are shown in Table V. According to these predictions, ZeroR has generated a quite bad model.

TABLE V. PREDICTIONS OF ZeroR CLASSIFIER

k-fold	Instance	Actual	Predicted	Error
1	1	9681.7	7108.916	-2572.784
	2	10537.2	7108.916	-3428.284
	3	10102.6	7108.916	-2993.684
	4	5003.3	7108.916	2105.616
	5	9934.8	7108.916	-2825.884
2	1	7113.8	7216.881	103.081
	2	11175.2	7216.881	-3958.319
	3	2356.3	7216.881	4860.581
	4	12240.9	7216.881	-5024.019
	5	8378.7	7216.881	-1161.819
3	1	11672.9	7224.042	-4448.858
	2	12645.4	7224.042	-5421.358
	3	3239.3	7224.042	3984.742
	4	6218.3	7224.042	1005.742
4	1	3874.8	6923.65	3048.85
	2	12578.7	6923.65	-5655.05
	3	15831.2	6923.65	-8907.55
	4	12906.1	6923.65	-5982.45
5	1	1779.6	7572.421	5792.821
	2	2130.8	7572.421	5441.621
	3	6764.3	7572.421	808.121
	4	9862.8	7572.421	-2290.379
6	1	13062.7	7379.139	-5683.561
	2	9864.6	7379.139	-2485.461
	3	3082.3	7379.139	4296.839
	4	1872.6	7379.139	5506.539
7	1	3877.5	7510.445	3632.945
	2	892.6	7510.445	6617.845
	3	985	7510.445	6525.045
	4	17137.1	7510.445	-9626.655
8	1	10306.5	7401.529	-2904.971
	2	13394.9	7401.529	-5993.371
	3	1449.2	7401.529	5952.329
	4	1880.8	7401.529	5520.729
9	1	871.6	7795.274	6923.674
	2	3874.8	7795.274	3920.474
	3	6597.3	7795.274	1197.974
	4	725.4	7795.274	7069.874
10	1	2130.8	7260.632	5129.832
	2	1872.6	7260.632	5388.032
	3	13828.7	7260.632	-6568.068
	4	14553.4	7260.632	-7292.768

In the experiments, the second classifier to determine the classifier with the best generalization ability is Multilayer Perceptron. The performance results of this classifier at the end of the experiment are shown in Table VI. When the results are observed carefully, it is seen that CC values of the experiments 3 and 4 increases relatively, while MAE and RMSE values decrease.

This is generally seen as a progress for predictions, but in some predictions it shows that the result is notably moving away from the actual value. In such case, model to be chosen may vary depending on the problem. According to the problem handled, primarily, error rates of the models with the highest CC values are compared. Model with the lowest error rate is chosen as ultimate model. Accordingly, model that gives the results of experiment 4 is our model.

TABLE VI. PERFORMANCE RESULTS ACCORDING TO PARAMETER CHANGES OF MULTILAYER PERCEPTRON

No	Parameter	CC	MAE	RMSE
1	Default	0,9883	576.6183	752.1817
2	hiddenLayer=8	0,9900	511.6861	694.5571
3	hiddenLayer=8 momentum=0,4	0.9908	489.1224	664.8059
4	hiddenLayer=8 momentum=0,4 trainingTime=5000	0.9954	381.9972	477.4231

Multilayer Perceptron classifier has one hidden layer and (attribute + class)/2 hidden neurons as default. In Artificial Neural Network models, increasing neuron or layer number improves the generalization ability of the model, while slowing down the learning process. In the experiment 4, hidden layer number in the model is 1 and neuron number in this hidden layer is 8. Increasing or decreasing of this value does not linearly increase or decrease the performance of the classifier, since in Neural Network models there is not only

one local minimum or local maximum. Therefore, in order to find global minimum and global maximum, the results should be observed after making some changes for the parameters. As a result of the changes of hidden layer numbers in "hiddenLayer" parameter and neuron numbers in this layer, the best result is obtained when hiddenLayer parameter has the value of 8. In addition, while searching for the parameter set that make classifier's performance best, we created the parameter set by anchoring a parameter and observing how other parameters change the result. "Momentum" parameter in parameter set adjusts the value which allows the weights to be updated during the training period of Multilayer Perceptron classifier. Its default value is 0,2. A performance increase is observed when this value is changed to 0,4. Apart from this, "trainingTime" parameter also increases the performance of the classifier to some certain extent. "trainingTime" parameter is the number of epoch that is necessary to train the classifier. While an increase in the number of epochs leads to a performance increase, decrease in the number of epochs means low classifier performance. However, this parameter does not linearly increase the classifier performance like the other parameters. This is due to the fact that Neural Network classifiers just like Multilayer Perceptron generate non-linear models. In Table VII, the model which is generated by Multilayer Perceptron classifier according to 4th experiment's parameters is shown.

TABLE VII. MULTILAYER PERCEPTRON CLASSIFIER'S MODEL

Layer	Node	Input	Weight
Hidden Layer	1	Threshold	-2.3442403191596277
		Year (Attribute)	0.12354083047678663
	2	Threshold	-2.3325852192376173
		Year (Attribute)	-0.18458044656817094
	3	Threshold	-6.465664570436303
		Year (Attribute)	4.964698558527786
	4	Threshold	-2.194373886436189
		Year (Attribute)	-1.3094074725790412
	5	Threshold	-2.328428190417552
		Year (Attribute)	-0.2698199164553271
	6	Threshold	-0.3389971904117853
		Year (Attribute)	-5.9476127452751175
	7	Threshold	-2.343452361179468
		Year (Attribute)	0.13867800434684255
	8	Threshold	-2.3359487946477753
		Year (Attribute)	-0.17125525070746103
Output Layer	1	Threshold	0.373348246603906
		Node-1	0.12668910263226776
		Node-2	-0.14152286472732936
		Node-3	3.416079289452131
		Node-4	-0.9895855396357082
		Node-5	-0.2121039034366091
		Node-6	-0.9976594346722308
		Node-7	0.1404520255410198
		Node-8	-0.13049112546948766

Another classifier used in the experiments is LWL classifier. Performance results of LWL classifier at the end of training is shown in Table VIII. According to these results, when LWL classifier uses Multilayer Perceptron as base classifier, it obtains the best results. CC value is bigger than 0.9 for all the base classifiers shown in Table VIII. Other classifiers below this value are not shown in Table VIII. Besides, LWL classifier uses DecisionStump as default base classifier. In addition to this, default parameters are used for all base classifiers. That is to say, there are not any changes in parameters, since it is observed that there is no remarkable performance change with the help of a few parameter changes.

TABLE VIII. PERFORMANCE RESULTS OF LWL CLASSIFIER AT THE END OF THE TEST

No	Base-Learner	CC	MAE	RMSE
1	DecisionStump (Default)	0.8895	1877.6546	2241.5772
2	LinearRegression	0.9859	657.2554	821.2632
3	MultilayerPerceptron	0.9862	609.2091	819.4034
4	DecisionTable	0.9815	643.9478	943.7720
5	REPTree	0.9588	970.8804	1419.8048

Fourth classifier used in the experiments is Additive Regression which is a meta classifier. Performance results of Additive Regression classifier are shown in Table IX. According to these results, it is seen that CC value is 0.9924 when M5Rules classifier is chosen as the base learner. Besides these results, MAE and RMSE results are also low compared to the results obtained by other classifiers. Because of this increase in error rates, the model created by 6th experiment can be considered to give the best performance.

TABLE IX. RESULTS OF ADDITIVEREGRESSION CLASSIFIER AT THE END OF THE TRAINING

No	Base-Learner	Parameter	CC	MAE	RMSE
1	Decision Stump	Default	0.9723	938.3252	1154.8086
2	LinearRegression	Default	0.9859	657.2554	821.2632
3	LWL	Default	0.9723	938.3252	1154.8086
4	Multilayer Perceptron	Default	0.9887	577.9342	741.1283
5	Decision Table	Default	0.9815	643.9478	943.772
6	M5Rules	Default	0.9924	456.7968	607.7613
7	M5P	Default	0.9921	490.7379	617.385
8	REPTree	Default	0.9728	881.7804	1152.2625

The last classifier used in experiments is M5Rules. This is a rule based classifier. In Table X, results of M5Rules classifier are shown. It is observed that parameter changes for the classifier lead to low performance results. Only the results of the experiment 5 perform a model with better performance, compared to those of other experiments. This is clearly seen in Table X.

TABLE X. PERFORMANCE RESULTS OF M5RULES CLASSIFIER AT THE END OF TRAINING

No	Parameter	CC	MAE	RMSE
1	Default	0.9926	457.6161	600.8713
2	unpruned=True	0.9910	498.8223	686.6973
3	useUnsmoothed=True	0.9924	458.8917	609.9547
4	buildRegressionTree=True	0.8364	2344.7211	2835.5465
5	minNumInstances=6	0.9929	428.6979	591.8440

In Table XI, estimations made by M5Rules that has the parameters from the 5th experiment and actual values are shown. When the results are observed, it is seen that the difference between predicted value and actual value is very low for some instances. For other instances, it can be said that error rate is of a reasonable level, as well. Besides, the best estimations are made by M5Rules classifier among the other four classifiers with default parameter. Because of this, it can be said that the best model is produced by M5Rules. So, the model can be said to have the generalization ability to estimate the electricity load between the years 2012-2021.

TABLE XI. M5RULES CLASSIFIER'S PREDICTIONS

k-fold	Instance	Actual	Predicted	Error
1	1	9681.7	8357.776	-1323.924
	2	10537.2	10777.922	240.722
	3	10102.6	9971.575	-131.025
	4	5003.3	4032.89	-970.41
	5	9934.8	9567.297	-367.503
2	1	7113.8	7668.286	554.486

	2	11175.2	11234.157	58.957
	3	2356.3	2927.011	570.711
	4	12240.9	12027.176	-213.724
	5	8378.7	8064.253	-314.447
3	1	11672.9	11649.4	-23.5
	2	12645.4	12840.712	195.312
	3	3239.3	3345.97	106.67
	4	6218.3	4470.66	-1747.64
4	1	3874.8	3768.699	-106.101
	2	12578.7	12395.055	-183.645
	3	15831.2	15163.024	-668.176
	4	12906.1	13186.367	280.267
5	1	1779.6	1550.068	-229.532
	2	2130.8	2686.693	555.893
	3	6764.3	7253.593	489.293
	4	9862.8	9240.916	-621.884
6	1	13062.7	13651.862	589.162
	2	9864.6	8797.229	-1067.371
	3	3082.3	3109.8	27.5
	4	1872.6	1760.974	-111.626
7	1	3877.5	4031.229	153.729
	2	892.6	877.939	-14.661
	3	985.4	1103.615	118.215
	4	17137.1	15429.776	-1707.324
8	1	10306.5	10465.077	158.577
	2	13394.9	14094.098	699.198
	3	1449.2	1329.156	-120.044
	4	1880.8	2228.096	347.296
9	1	871.6	512.134	-359.466
	2	3874.8	3550.006	-324.794
	3	6597.3	6840.006	242.706
	4	725.4	278.599	-446.801
10	1	2130.8	2466.395	335.595
	2	1872.6	2018.959	146.359
	3	13828.7	14528.371	699.671
	4	14553.4	14934.795	381.395

In Table XII, the model constructed by M5Rules is shown. When we look at this model, we see that it is a smoothed linear model. It consists of two rules.

TABLE XII. M5RULES CLASSIFIER'S MODEL

No	Rules
1	IF year > 552218400000 THEN electricityLoad = 0 * year - 745.0728 [24/11.654%]
2	electricityLoad = 0 * year + 424.0746 [18/25.956%]

Performance values of cross validation method that is used when evaluating the generalization ability of M5Rules classifier are shown in Table XIII, according to k value. In this table, results of LOOCV method are also shown. According to these results, it is seen that the model constructed by M5Rules classifier is not changed. So, it can be said that M5Rules classifier's model has the generalization ability over test instances, as well.

TABLE XIII. PERFORMANCE RESULTS OF M5RULES CLASSIFIER ACCORDING TO K-VALUES

No	k-value	CC	MAE	RMSE
1	2	0.9861	692.5857	905.2848
2	5	0.9932	418.9938	584.0278
3	10 (Default)	0.9929	428.6979	591.8440
4	20	0.9929	426.8816	590.9269
5	30	0.9930	420.3135	587.9991
6	42 (LOOCV)	0.9928	429.1589	595.5545
Average		0,9918	469.4386	642.6062

In Fig. 2, a comparison of the estimations made by Multilayer Perceptron and M5Rules classifiers using the data set. In this comparison, LWL and Additive Regression classifiers are Meta learners and that is why they are not presented in this comparison. These meta learners are not shown in the comparison graph since they use Multilayer Perceptron and M5Rules classifiers as base classifiers when making the best estimations. When Fig. 2 is observed, it is seen that Multilayer Perceptron and M5Rules classifiers make estimations close to actual values starting by the middle of the instances. This shows that classifiers' performances increase with the increase in the number of the instances. Another way to prove that the performance of the classifiers change according to the training examples is to create the Happy Graph. The Happy Graph is widely used to find training examples which improves the performance of the classifiers to the maximum. This situation also shows that classifier models with less training clusters, but the same performance can be created.

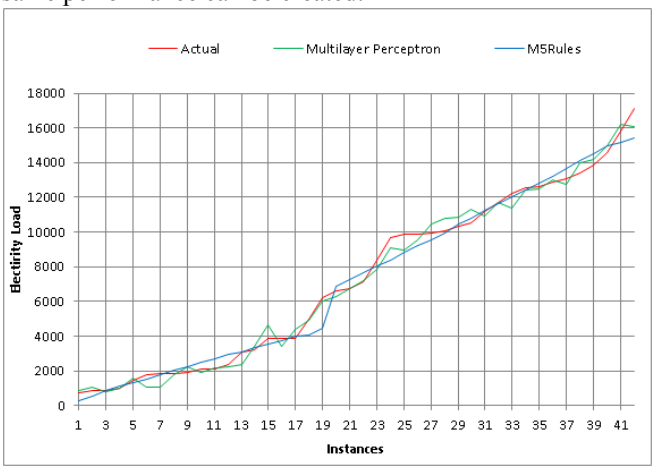

Figure 2. Comparison of actual values and estimated values of Multilayer Perceptron and M5Rules classifiers

In Fig. 3, the difference in the RMSE value depending upon the changes in the dataset of the Multilayer Perceptron and M5Rules classifier is shown. This is called The Happy Graph of the classifiers. According to this graph, a performance increase for both of the classifiers is clear when the data set includes a data range between 10 and 19 (the data between 1993 and 2011). In cross validation method, choosing "k" value as 10 leads to have a data set with a number of data greater than 10.

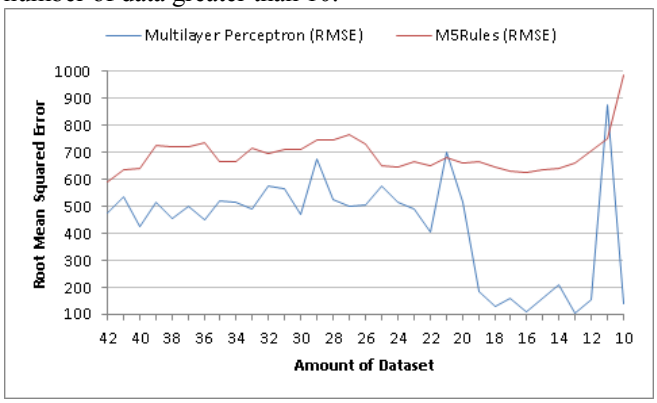

Figure 3. RMSE alterations according to data amount of Multilayer Perceptron and M5Rules classifiers.

Moreover, the reason for reducing the data quantity of the data set, instead of increasing it, is the continuous increase of the electricity generated by hydroelectric power plants by use of developing technology. Namely, it is because the number and the efficiency of these power plants is increasing year by year. The data set always includes the recent data. That is, the data set with 42 amounts of data includes the data between 1970 and 2011 and the data set

with 10 amounts of data includes the data between 2002 and 2011. In Fig. 4, alteration in MAE value according to the alteration in the data set of M5Rules and Multilayer Perceptron is shown. According to this graph, the estimation of Multilayer Perceptron is much better than that of M5Rules classifier.

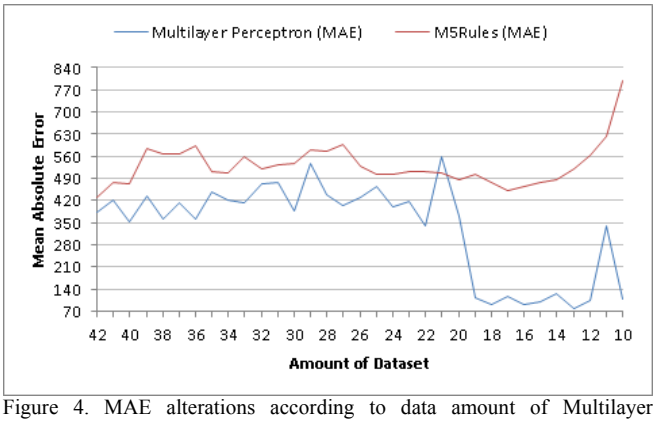

Figure 4. MAE alterations according to data amount of Multilayer Perceptron and M5Rules classifiers.

RMSE relation and data amount of the model that is generated by Multilayer Perceptron classifier which produces a data set according to both cross validation method and training method are shown in Fig. 5. This relation shows whether the errors made by Multilayer Perceptron's model are because of the bias or the variance. If the difference between $error_{cv}$ and $error_{training}$ through x-axis is high, then a high variance can be uttered. As a result, it can be said that the model performed by Multilayer Perceptron learned the data set well enough. In short, high variance or bias cannot be mentioned.

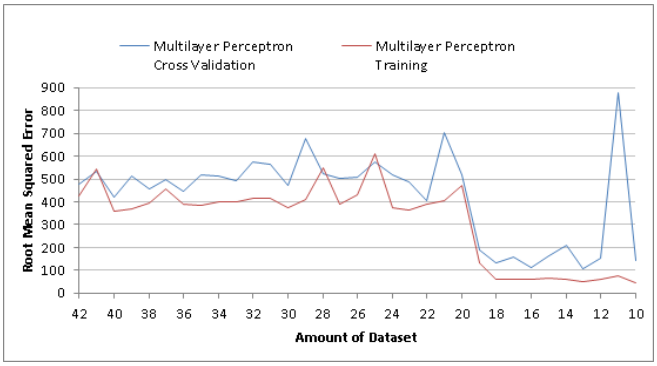

Figure 5. RMSE relation and data amount of the model that is generated by Multilayer Perceptron classifier (Bias/Variance of the model generated)

In Fig. 6, RMSE relation and data amount of the model that is generated by M5Rules classifier which produces a data set according to both cross validation method and training method are shown. This relation shows whether the errors made by M5Rules classifier's model are because of the bias or the variance. If the difference between $error_{cv}$ and $error_{training}$ through x-axis is high, then a high variance can be uttered. If the difference between is not very high, but $Error_{cv}$ is high, once again a high variance can be uttered as well. After all, M5Rules classifier's errors are because of bias. However, when compared to Multilayer Perceptron, M5Rules classifier can be said to have relatively high bias.

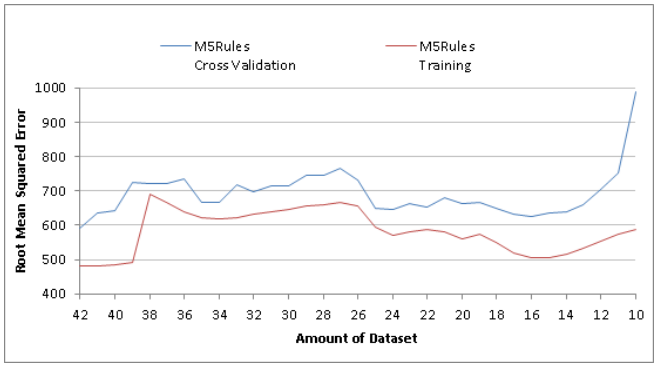

Figure 6. RMSE relation and data amount of the model that is generated by Multilayer Perceptron classifier (Bias/Variance of the model generated)

Consequently, it is observed that Multilayer Perceptron classifier has a better performance than that of M5Rules upon this dataset. As a result of this, reliability of Multilayer Perceptron classifier is greater for the load demands concerning the years 2012-2021. In Fig. 7, load demand estimations of M5Rules and Multilayer Perceptron classifiers for the years 2012-2021 are given. Amount of data is 19 in the data set for these estimations. While determining this value, the results in Fig. 3 and Fig. 4 are taken into consideration.

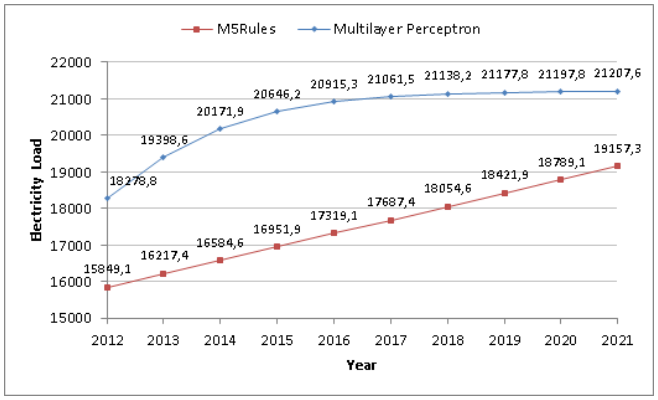

Figure 7. Load demand of M5Rules and Multilayer Perceptron classifiers for the years 2012-2021

After having decided the data set (19 data) for the training of Multilayer Perceptron, some alterations were made on the parameters of the classifier and the optimum model tried to be acquired. New parameters are shown in Table XIV in this respect.

TABLE XIV. PERFORMANCE RESULTS ACCORDING TO PARAMETER CHANGES OF THE CLASSIFIERS

Classifiers	Parameter	CC	MAE	RMSE
Multilayer Perceptron	Default	0.9590	480.5121	600.3940
	hiddenLayer=8 learningRate=0.4 momentum=0.4 trainingTime=2000	0.9933	148.6022	266.4394
M5Rules	Default	0.9480	503.8243	667.7422

Estimations of Mutilayer Perceptron according to these parameters are shown in Fig. 8.

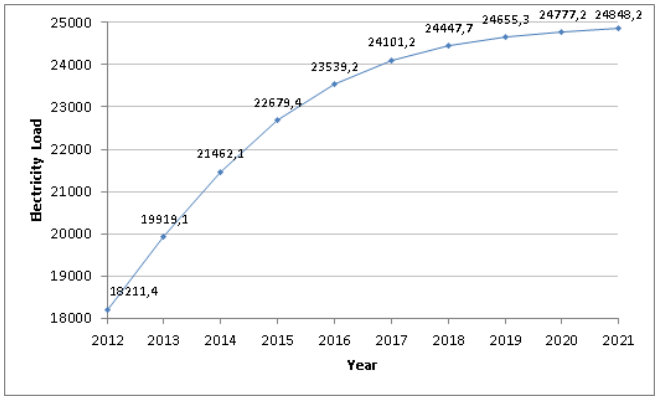

Figure 8. Electricity Load demand predictions of Multilayer Perceptron classifiers for the years 2012-2021

Finally, load demand estimation for the year 2011 was calculated as 17283.7 by using the model performed by Multilayer Perceptron with the parameters shown in Table XIV and having the data between 1993-2010 as the training data and the data about 2011 as the test data. The real value for the year 2011 is 17137.1. According to this, the absolute difference between is just 146.6. This is lower than the value of MAE=148.6 shown in Table XIV. It is seen that the estimation of the model performed by Multilayer Perceptron classifier is very close to the real electricity load demand of the year 2011. As a result, it can be said that the model performed by Multilayer Perceptron classifier has generalization ability over the test data.

VI. CONCLUSION

The electricity load demand, between 2012 and 2021, was estimated by using the load demand of the electricity generated from hydroelectric power plants in Turkey between 1970 and 2011. Among machine learning algorithms, Multilayer Perceptron, Locally Weighted Learning, Additive Regression, M5Rules and ZeroR classifiers are used to estimate the electricity load demand. Locally Weighted Learning and Additive Regression classifiers contain base classifiers by their very nature. That is to say, these are meta classifiers. However, Locally Weighted Learning and Additive Regression classifiers are also lazy learners. Models with the best performance are generated during the training phase carried out by Locally Weighted Learning and Additive Regression classifiers when Multilayer Perceptron and M5Rules are chosen respectively as base classifiers. However, since these classifiers generate models with good performance already, Locally Weighted Learning and Additive Regression classifiers have not been evaluated.

As a result of the experiments performed by using M5Rules and Multilayer Perceptron classifiers, it was seen that M5Rules cannot produce a good model because of the bias and the model performed by Multilayer Perceptron was suitable for the data set. Therefore, the classifier for the estimation was determined as Multilayer Perceptron. In addition, the CC, MAE and RMSE values of the model performed by Multilayer Perceptron was obtained as 0.9933, 148.6022 and 266.4394 respectively. Besides, the difference between the real value and the estimated value for the test data of the year 2011 was obtained as 148,6. Consequently, the complexity of the model performed by Multilayer Perceptron classifier corresponds to the complexity of the data set and this brings the classifier high generalization ability. For this reason, it was seen that Multilayer Perceptron can be used to learn the data set.

REFERENCES

[1] Y. Rui, A. A. El-Keib, "A review of ANN-based short-term load forecasting models", In Proceedings of the 27th Southeastern Symposium on System Theory. IEEE Computer Society, Washington, DC, USA, 1995, pp. 78-82.

[2] A. L. Samuel, "Some studies in machine learning using the game of checkers", IBM Journal of Research and Development, vol. 3, no. 3, pp. 210-229, 1959.

[3] R. Kohavi, F. Provost, "Glossary of Terms", Machine Learning, vol. 30, pp. 271-274, 1998.

[4] R. O. Duda, P. E. Hart, D. G. Stork, "Pattern classification - 2nd ed.", New York, Wiley-Interscience, pp. 680, 2000.

[5] M. Negnevitsky, P. Mandal, A. K. Srivastava, "Machine Learning Applications for Load Price and Wind Power Prediction in Power Systems", 15th International Conference on Intelligent System Application to Power Systems, Curitiba, Brazil, 2009, pp. 1-6.

[6] S. Fan, L. Chen, W. Lee, "Machine learning based switching model for electricity load forecasting", Energy Conversion and Management, vol. 49, no. 6, pp. 1331-1344, 2008.

[7] Y. Guo , D. Niu, Y. Chen, "Support Vector Machine Model in Electricity Load Forecasting", Machine Learning and Cybernetics, 2006 International Conference, 2006, pp. 2892-2896.

[8] R. A. Swief, Y. G. Hegazy, T. S. Abdel-Salam, M.A Bader "Load-Price Forecasting Model Employing Machine Learning Techniques", The Online Journal on Power and Energy Engineering, vol. 1, no. 2, pp. 36-39, 2010.

[9] S. Russell, P. Norvig, "Artificial Intelligence: A Modern Approach", Prentice Hall, 2002.

[10] I. H. Witten, E. Frank, M. A. Hall, "Data Mining: Practical machine learning tools and techniques", 3rd Edition, Morgan Kaufmann, 2011.

[11] M. Riedmiller, "Advanced Supervised Learning in Multi-layer Perceptrons From Backpropagation to Adaptive Learning Algorithms", Computer Standards & Interfaces, vol. 16, no. 3, pp. 265-278, 1994.

[12] H. Zhang, "The Optimality of Naive Bayes", Proceedings of the 17th International FLAIRS conference (FLAIRS2004), 2004, pp. 562-567.

[13] G. H. John, P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers", Eleventh Conference on Uncertainty in Artificial Intelligence, San Mateo, 1995, pp. 338-345.

[14] E. Frank, M. Hall, B. Pfahringer, "Locally Weighted Naive Bayes", In: 19th Conference in Uncertainty in Artificial Intelligence, 2003, pp. 249-256.

[15] C. G. Atkeson, A. W. Moore, S. Schaal, "Locally Weighted Learning", Artificial Intelligence, vol. 11, pp. 11-73, 1997.

[16] U. M. Fayyad, K. B. Irani, "Multi interval discretization of continuous valued attributes for classification learning", In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, 1993, pp. 1022-1027.

[17] R. J. Quinlan, "Learning with Continuous Classes", In: 5th Australian Joint Conference on Artificial Intelligence, Singapore, 1992, pp. 343-348.

[18] Y. Wang, I. H. Witten, "Induction of model trees for predicting continuous classes", In: Poster papers of the 9th European Conference on Machine Learning, 1997.

[19] J. G. Zhang, H. W. Deng, "Gene selection for classification of microarray data based on the Bayes error", BMC Bioinformatics, vol. 8, pp. 370, 2007.

[20] I.H. Witten, E. Frank, "Datamining: practical machine learning tools and techniques - 2nd ed.", the United States of America, Morgan Kaufmann series in data management systems, 2005.

[21] J. Cohen, "Statistical power analysis for the behavioral sciences (2nd ed.)", Lawrence Erlbaum Associates, pp. 567, 1988.

[22] E. L. Lehmann, G. Casella, "Theory of Point Estimation (2nd ed.)", New York: Springer, 1998.

[23] P. Domingos, "A Unifed Bias-Variance Decomposition and its Applications", In Proc. 17th International Conf. on Machine Learning, 2000, pp. 231-238.

[24] S. Geman, E. Bienenstock, R. Doursat, "Neural networks and the bias/variance dilemma", Neural Computation, vol. 4, no. 1, pp. 1-58, 1992.

[25] E. Alpaydın, "Introduction to Machine Learning", the United States of America, The MIT Press, 2004.

[26] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection", Proc. of the 14th Int. Joint Conf. on A.I., Canada, 1995.

[27] A. Lendasse, V. Wertz, M. Verleysen, "Model Selection with Cross-Validations and Bootstraps — Application to Time Series Prediction with RBFN Models", Artificial Neural Networks and Neural Information Processing – ICANN/ICONIP 2003, O. Kaynak, E. Alpaydin, E. Oja, L. Xu eds, Springer-Verlag, Lecture Notes in Computer Science Vol. 2714, pp. 573-580, 2003.

[28] T. M. Mitchell, "Machine Learning", McGraw-Hill Science/Engineering/Math, pp. 432, 1997.

[29] N. J. Nilsson, "Introduction to Machine Learning: An Early Draft of a Proposed Textbook", Robotics Laboratory, Department of Computer Science, Stanford University, 1996.

[30] R. R. Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald, D. Seuse, "WEKA Manual for Version 3-6-0", The University of Waikato, 2008.

[31] D. H. Wolpert, W. G. Macready, "No Free Lunch Theorems for Optimization", IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, 1997.

[32] D. H. Wolpert, W. G. Macready, "Coevolutionary free lunches", IEEE Transactions on Evolutionary Computation, vol. 9, no. 6, pp. 721-735, 2005.

[33] C. Luque, D. Quintana, P. Isasi, "Predicting IPO underpricing with genetic algorithms", International Journal of Artificial Intelligence, vol. 8, no. S12, pp. 133-146, 2012.

[34] G. Prakash, M. Kulkarni, U. S. Acharya, M. N. Kalyanpur, "Classification of FSO Channel Models Using Radial Basis Function Neural Networks and Their BER Performance with Luby Transform Codes", International Journal of Artificial Intelligence, vol. 9, no. A12, 2012.

[35] I. Martišius, K. Šidlauskas, R. Damaševicius, "Real-Time Training of Voted Perceptron for Classification of EEG Data", International Journal of Artificial Intelligence, vol. 10, no. S13, 2013.

[36] A. Ismail, D.-S. Jeng, L. L. Zhang, J.-S. Zhang, "Predictions of bridge scour: Application of a feed-forward neural network with an adaptive activation function", Engineering Applications of Artificial Intelligence, vol. 26, no. 5–6, pp. 1540-1549, 2013.