Comparison of Machine Learning Algorithms for the Power Consumption Prediction

Case Study Tetouan/Morocco

Mohamed El-hassnaoui

New England Institute of Technology

**Acknowledgements**

I would like to take this opportunity to express my gratitude to the following contributors whom helped me to succeed in this research.

First, I would like to thank my professor Mr. Timothy Henry who guided me during this research project through his valuable feedback and ideas. Second, I would like to thank all staff of New England Institute of Technology, whom helped me directly or indirectly to succeed in this research project. Third, I would like to thank the Amendis company and Mr. Abdel-Aziz El-hibaoui the department head of Computer Science and Mathematics at the University of Science Tetouan City Morocco, whom provided me with data of power consumption and weather data.

Abstract

Predicting electricity power consumption is an especially important task which provides intelligence to utilities and helps them to improve their systems’ performance in terms of productivity and effectiveness. Machine learning models are the best accuracy models used in prediction. In this work I compare different types of machine learning models that recently have gained popularity: Feedforward Neural Network with Back-Propagation Algorithm, Random Forest, Decision Tree, and Support Ector Machine for Regression (SVR) with Radial Basis Function Kernel. The parameters associated with the comparative models are optimized based on Grid-search method in order to find the accurate performance. The dataset that is used in this comparative study is related to three power distribution networks of Tetouan City, which is located in north Morocco. This historical data has been taken from supervisory control and the data acquisition system (SCADA) for each 10 minutes for the period between 2017-01-01 and 2017-12-31. This study is used to predict the power consumption of the next 10 minutes and the next hour to determine which approach is most successful. The results indicate that random forest model algorithm achieved the smaller prediction errors compared to their counterpart algorithms.

Keywords— energy prediction, artificial neural networks, random forest, decision tree, support vector regression.

**Table of Contents**

1. **Introduction**
2. **Overview of Machine Learning Algorithms**
3. **Case study**
4. **Methodology**
5. **Experiments and Results**
6. **Conclusion**
7. **Environment Installation and Code-Source**
8. **References**

Comparison of Machine Learning Algorithms for the Power Consumption Prediction Case Study Tetouan City/Morocco

Over time, increased consumption of electric power and increased attention to its details, such as electrical forecasting, have been analyzed by researchers. Also electricity companies have been spending a lot of money and effort to control and manage their electric power effectively. Therefore, it becomes necessary to review all new and available methods and to choose the best services provided by energy companies. One of those things which needs to be considered is the amount of energy produced and consumed in order to balance production and consumption, to decrease the cost of production, and to control future planning.

Knowing the real production and consumption of power is the first step of making a good electrical system. To save resources and reduce cost, power utilities are required to balance between produced power and customers’ consumption. The relationship is described by the following equation:

is the energy lost in the transportation, and is real number. According to the value of , I distinguish between the three following cases:

* If is a large positive number, that means this is a quantity of energy produced but not used. In general, this exceeded energy is lost. The problem is that the extra power needs to be stored or the production needs to be reduced.
* If is a large negative number, that means the consumption is larger than production. In this case, the problem blackout can occur. So, new resources of energy are required to handle this situation.
* If is positive and is near to , that means it is a stable electrical system, and there is a harmony between the production and consumption.

The goal of power companies is to keep the value of like the third case. They have to make a balance between production and consumption. To this end, they need a strong system, which comes from accurate prediction. The electric power consumption has increased in the last decade due to the expanding economic development and population. Specifically, the annual electricity power consumption is growing in the industrial and domestic sectors. There are many factors that determine the energy consumption: I cite here weather, population, price of electricity, and consumer behavior. The variability of those factors makes energy prediction more difficult. Considering this complexity, researches have been concentrated on finding the most accurate prediction models to predict the real demand of energy consumption. So, many algorithms and models have been proposed to offer solutions to the power prediction. In general, those algorithms are classified into three categories: statistical, engineering, and artificial intelligence. In literature, several researchers use artificial intelligence algorithms to make a prediction model, especially machine learning algorithms.

Tüfekc (2012) examined machine learning to predict full load electrical power output of a base load operated combined cycle power plant. Xypolytou, Meisel, and Sauter (2017) studied short-term electricity consumption forecasts with artificial neural networks a case study of office buildings. Ahmad, Mourshed, and Rezgui (2017) compared feedforward back-propagation artificial neural network with random forest to predict power consumption of a hotel in Madrid, Spain. Günay (2016)predicted annual gross electricity demand by artificial neural networks case of Turkey. Kankal and Uzlu (2017) studied the performance of an artificial neural network for modeling electricity energy demand in Turkey. Amber, Ahmad, Aslam, Kousar, Usman, and Khan (2018) compared five artificial intelligent system techniques to predict electricity power consumption of a building located in London. Kaytez, Taplamacioglu, Cam, and Hardalac (2015) compared the regression analysis, neural networks and least squares support vector machines for predicting the electricity energy consumption of Turkey. Pombeiro, Santos, Carreira, Silva, and Sousa (2017) compared Linear regression vs. fuzzy modeling vs. neural networks models to predict electricity consumption in an institutional building. Paudel, Elmitri, Couturier, Nguyen, Kamphuis, Lacarrière, and Le-Corre (2017) predicted energy consumption of low energy building based on support vector machine. Khosravani, Castilla, Berenguel, Ruano, and Ferreira (2016) compared prediction models for energy consumption based on neural networks of a bioclimatic building.   
Those previous cited models may be appropriate for some cases but it cannot be generalized.

In this study, I will compare the well-known machine learning algorithms in order to increase the efficiency and revenues of the electrical generating and distribution network companies and to assist them planning their capacity and operations to supply all consumers with reliable, required energy. The availability of historical data allows me to use the supervised models, such as decision tree, support vector machine for regression, artificial neural network (feedforward neural network with backpropagation algorithm) and random forest. I will compare these models to the linear regression method. The comparative study is based on historical consumption energy data of Tetouan City for the period between 2017-01-01 and 2017-12-31. The historical data used are for Quads, Boussafou and Smir distribution networks. The SCADA system of the regional distribution of drinking water and electricity company, AMENDIS, provided the data. This data is exclusive and have not been used before my research. Due to the dependence of prediction models on the input variables to obtain the best output, I will include the weather data taken for the same period of power consumption. Moreover, I will use the attributes of date and time as independent variables, study the impact of those factors on the prediction, and determine the importance of each factor on power consumption.

The rest of this paper is organized as follows: Section II exposes a technical overview of different machine learning algorithms in comparison. Section III presents the case study and the description of datasets. Section IV is devoted to the methodology used. Experiments and their results are covered in Section V. And finally, Section VI concludes the paper and dresses some perspectives.

Method

**Overview of Machine Learning Algorithms**

In this section, I will provide a brief description of five different machine-learning methods: linear regression, decision tree, random forest, support vector machine for regression with radial basis function kernel, and artificial neural network. Here, I use the common notations to describe the algorithms.

**Linear regression.**

Linear regression is considered as one of the simplest approaches, and it can be used as a baseline performance measure. It is based on linear relationship between the dependent and independent variables (Fumo & Rafe-Biswas , 2015). It is defined by this equation:

(1)

Where are the available inputs and are the functional weights.

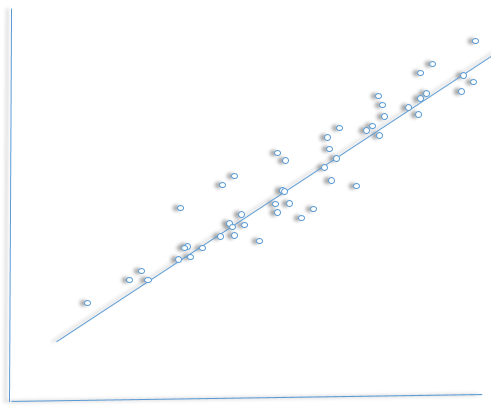


Fig 1 . Linear regression example for two variables

**Decision tree.**

The decision tree is a commonly used machine learning method (Quinlan, 1986). It is a kind of classification or regression, and it utilizes a tree structure to separate a set of data into several predefined classes, giving the characterization, generalization, and classification of given datasets (Yu, Haghighat, Fung, & Yoshino , 2010). Its goal is to predict the target variable value by learning simple decision rules deduced from the data, and it shows how the target variable can be forecasted by predictor variables set.

There are many types of decision tree generation, such as ID3 (Quinlan, 1986), C4.5 (Salzberg, 1994)and classification and regression trees (CART) (Breiman, 2017). In this work, I implemented CART, along with a Scikit-learn: Machine Learning in Python (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Cournapeau, & Passos, 2011). CART is nonparametric procedure to predict continuous dependent variables which utilizes a binary tree to divide the predictor space into subsets recursively (Razi, & Athappilly, 2005). According to my study, I found the performance of this model better than its family counterpart.

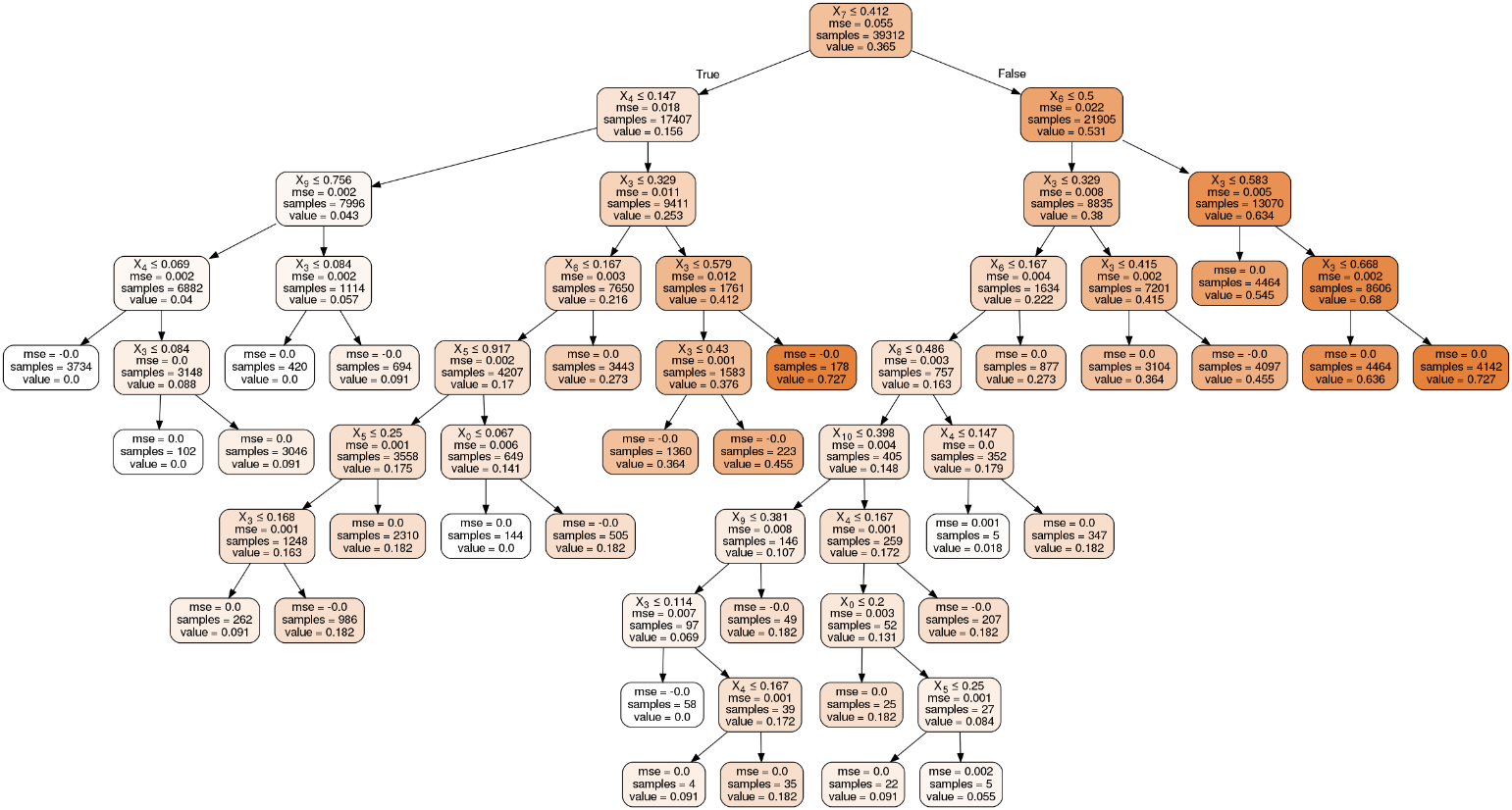


Fig 2. Drawing of Decision Tree for the output of Quads distribution network

**Random forest.**

Random forest is classified as an ensemble approach which combines the performance of numerous decision tree algorithms to predict the variable value. It was proposed to improve the accuracy of decision tree, and it has a different construction for regression and classification.

A number of regression trees are built by random forest, and the result is the average. While trees are grown, a regression predictor is defined as:

(2)

Where is p-dimensional vector of inputs and is referred to decision tree.

Randomly selecting a set of trees in the forest is accomplished to make a new training set. The set of unselecting trees is known as out of bag samples (Jiang, Tang, Wu, & Fu, 2009). Random features are selected in each split node of a decision tree, instead of all features. This process is repeated in order to create a random forest (Breiman, 2019). Aggregation of each individual prediction tree makes the prediction of the random forest, and this aggregation prediction gets better performance than the individual prediction of trees (Kane, Price, Scotch, & Rabinowitz, 2014). Moreover, random forest provides an estimation of the relevant important features and how each feature affects the prediction (Gislason, Benediktsson, & Sveinsson, 2006). In sum, simplicity, velocity, interpretability, accuracy, and ease of use are the most important properties of random forest.

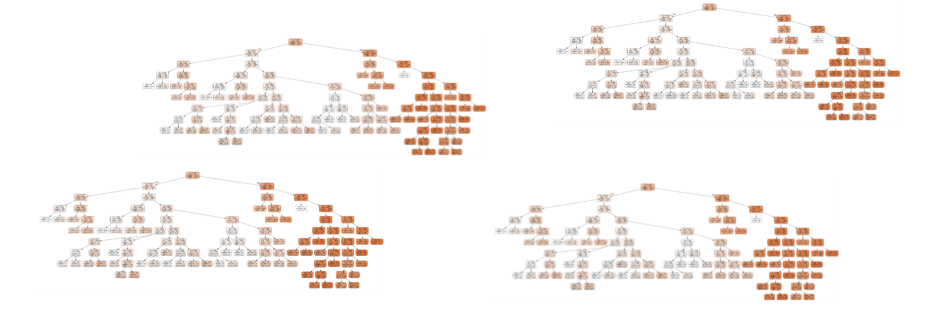


Fig 3. Drawing example for random forest of the four distribution-networks

**Support vector machine.**

The support vector machine (SVM) was introduced in the late 1960s, and it has not gotten significant consideration until recent years. SVM is a type of supervised method to achieve the classification of multidimensional, and it was originally invented as a linear classification then as a non-linear classifier. Lately, it has been used to solve regression problems (Drucker, Burges, Kaufman, Smola, & Vapnik, 1997) which are based on the concept of support vectors and is called support vector regression (SVR).

It is defined as:

(3)

Where is any nonlinear function to map input to output:

The best solution is detected by minimizing the following function:

(4)

Where is constant to control the penalty factor which is used to balance between smoothness and data fitting. and are slack variables to optimize the problems, and є is the loss function which is used to estimate the accuracy of prediction. One advantage of SVR is finding a unique solution to minimize the convex function, and it depends on providing and є (Edwards, New, & Parker, 2012) (Rodriguez-Galiano, Sanchez-Castillo, Chica-Olmo, & Chica-Rivas, 2015).

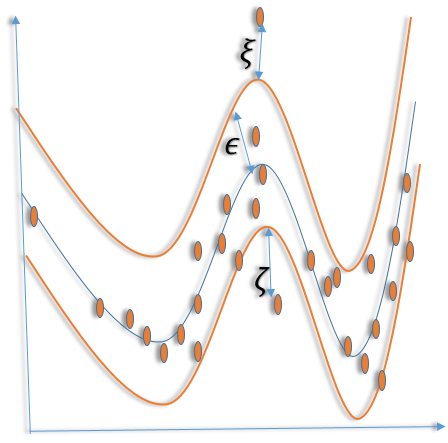


Fig 4. Drawing example for support vector regression for two variables

**Artificial neural network.**

Artificial neural network imitates the work of the brain. It is a technology that is currently widely used due to its ability to solve complex issues. Also, the artificial neural network is the most common method to develop nonlinear problems of regression and classification. Many types of networks are available in literature. Here, I concentrated on the most used one, namely the feedforward neural network. It learns through training, not through programming, and it collects the knowledge by identifying the relationships of data. Feedforward neural network basically consists of at least three layers. An input layer receives the data and processes it to the hidden layers. The hidden layers connect the input layer to the output layer through connections, and the output layer in my case (regression) combines of one output. There are no connections back from the output layer to the hidden layer or from the hidden layer to the input layer. The values are processed through transfer function from the input layer to the hidden layer and are multiplied by the connection values. Also, the values are forwarded from the hidden layers to the output layer in the same way. Mathematically, it can be described by the following equation:

(5)

Where ,, , , represent neural network output, the activation function, the number of hidden layers, the number of neurons in the hidden layer, the weight of connections, and the bias of the neuron, respectively. The transfer function is called the activation function, and there are many types. Sigmoid (Minai, & Williams, 1993), Hyperbolic Tangent Function(Tanh) (Glorot & Bengio, 2010), Rectified Linear Unit (ReLU) (Xu, Wang, Chen, & Li, 2015), Exponential linear Unit (ELU) (Clevert, Unterthiner, & Hochreiter, 2015), Scaled Exponential linear Unit (SELU) ( Klambauer, Unterthiner, Mayr, & Hochreiter, 2017) and Swish (Ramachandran, Zoph, & Le, 2017) activation functions were used in this work.

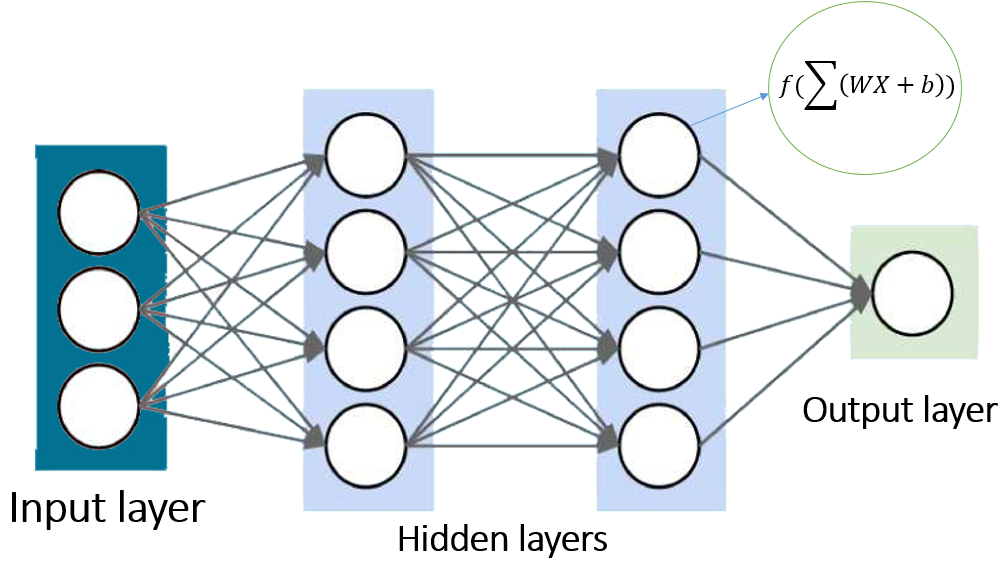


Fig 5. Feedforward neural network example for regression with two hidden layers

Feedforward neural networks learn through different types of learning rules, but backpropagation is the most used algorithm. To reduce errors, the learning rule is used with optimization algorithms to find the best parameters and compare the predicted output value with the real value and the errors’ feedback in order to adjust the weight of connections. This step is repeated until it reaches the minimum number of errors or number of epochs.

**Case study**

Tetouan is a city located in north Morocco which occupies an area of around … and its population is about … million according to last census and is increasing rapidly, approximately 2% annually. Since it is located along the Mediterranean Sea, its weather is mild and rainy in the winter, and hot and dry during the summer months. The data of power consumption was collected from supervisory control and data acquisition system (SCADA) of Amendis which is a public service operator in charge of the distribution of drinking water and electricity since 2002. The purpose of the electricity distribution network is to serve low and medium voltage consumers in Tetouan regions. For this purpose, the delivery and distribution of electrical energy from the point of delivery to the end user customer is ensured by Amendis. The energy that is distributed comes from the National Office of Electricity and Drinking Water. After transformation of the high voltage (63 kV) in medium voltage (20 kV), it is allowed to transport and distribute the energy on the whole of the delegated management scope. The distribution network is powered by 3 source stations, Qods, Smir and Boussafou.

The data in this study were the history of power consumption, collected in each 10 minutes for the period between 2017-01-01: 00:00:00 and 2017-12-31: 23:50:00. It is a unique dataset, and it does not have any missing data. It consisted of the date, time, and the consumption of the three distribution networks. Figure 6 shows the output consumption of the three distribution networks for the whole year of 2017 at each hour.

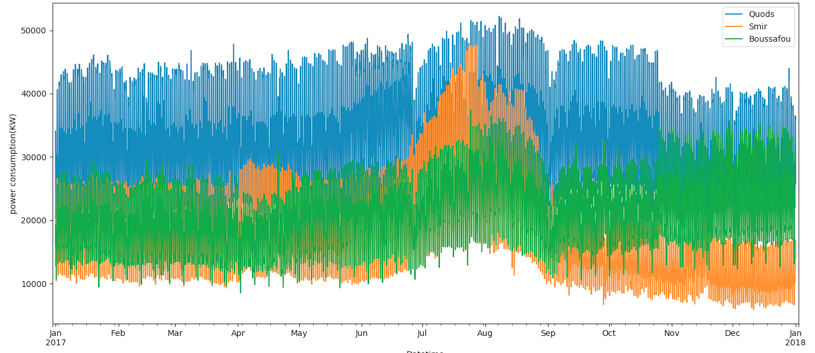


Fig 6. Power consumption for the year of 2017 at each hour for three distribution

There are similarities and differences between the three distributions: the increasing of power consumption in the summer is similar and that is because of the hot weather and vacation time. However, the difference is the reduced power consumption of Quads and Smir distribution in November and December compared to the power consumption of Boussafou distribution at the same months.

Different attributes of date and time are used as the inputs for the prediction models. Month, day of month, hour, day of year, week of year, day of week, quarter and minute are the independent variables, and their correlation to the dependent variable is shown in Figure 7.

|  |  |
| --- | --- |
| Quads distribution network | Smir distribution network |
| Boussafoudistribution network | Aggregation data |

Fig 7 Correlation relationship between power consumption of three distribution networks and calendar variables

Due to the people’s behavior, the consumption of power is changed in working days compared to weekends. Usually on working days the consumption is decreased in households and increased in factories, commercial, and public establishments, and the opposite happens on weekends. The dataset is aggregated data, and does not include whether the buildings were “green” or not. Figure 8 shows the consumption of on the weekdays, and it shows slight electricity power consumption is used on Sunday compared to the other days.

|  |  |
| --- | --- |
| daysplotboxQuads distribution network | Smir distribution network |
| Boussafoudistribution network | Aggregation data |

Fig 8. Box plot comparison of electricity consumption among week days.

There is no doubt that there are many factors which affect the power consumption, such as weather, income, population, electricity price, etc. The data from one of the factors, weather, were gathered from sensors that are located in the airport at the center of the city. This data were collected every 5 minutes in the period between 2017-01-01 and 2017-12-31. I reformed the data to be in every 10 minutes like the power consumption data by resampling the data and taking the average of two readings. The factor of weather that were considered were temperature, humidity, wind speed, (flux diffusion) and general flux diffusion. Table I shows the weather properties and the correlation between the power consumption of Quads distribution and the corresponding weather over the full period of the dataset.

Table i COEFFICIENT OF CORRELATION BETWEEN THE INPUT VARIABLES AND THE OUTPUT VARIABLE

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Count | Mean | STD | Min | Max | Correlation |
| Quods | 52560 | 32330 | 7133.05 | 13895.7 | 52204.4 | 1 |
| Temperature | 52560 | 18.81 | 5.82 | 3.247 | 40.010 | 0.440221 |
| Humidity | 52560 | 68.26 | 15.55 | 11.34 | 94.8 | -0.287421 |
| Wind Speed | 52560 | 1.96 | 2.34 | 0.05 | 6.483 | 0.167444 |
| deffus | 52560 | 75.03 | 124.21 | 0.011 | 936.00 | 0.080274 |
| Global | 52560 | 182.67 | 264.41 | 0.004 | 1163.00 | 0.187965 |

\*STD: Standard derivation

In Table I and Figure 7, I showed the power consumption correlation to the calendar and weather attributes. Also, I applied feature selection to data in order to determine the importance of predictive variables and deleted the unimportant features. There are many ways to perform that; one of them is random forest which identifies the true predictor of a large number of candidates (Genuer, Poggi, & Tuleau-Malot, 2010). It is shown in Figure 9 that all variables are important, but hour and temperature are the most valuable.

|  |  |
| --- | --- |
| importance  Quads Distribution Network | Smir Distribution Network |
| Boussafou Distribution Network | Aggregation Dataset |

Fig 9 Variable importance for Quads, Smir,Boussafou distribution dataset for minutely consumption

**Methodology**

In this study, I used five known types of algorithms that were used in prediction. Despite their advantages and the accuracy of the algorithms, these models require an accurate selection of the arrangement parameters in order to achieve best performance. I utilized the grid search method to find the best parameter for the models. It is classified as an exhaustive method for the best parameter values that have to be explored for each by setting the sort of values at first. Then, the method will show the score for each parameter value to be considered and which one will be selected. This method is applicable in case the required maximum of parameter is known (Ataei, & Osanloo, 2004). Grid search method is recommended to be used along with cross-validation in order to obtain best values (Lin, 2003). The calculations of this work have been implemented using Python 3.6, with base algorithm from Keras (Pedregosa et al., 2011).

As mentioned in the case study section, I got the data from different resources. After extracting, transforming and loading data from the resources, I normalized the data as a result of depending some models’ performance on normalization such as neural network. I transformed the data into the values between zero and one by using Min-Max Normalization which is one of the best used techniques and it is achieved by:

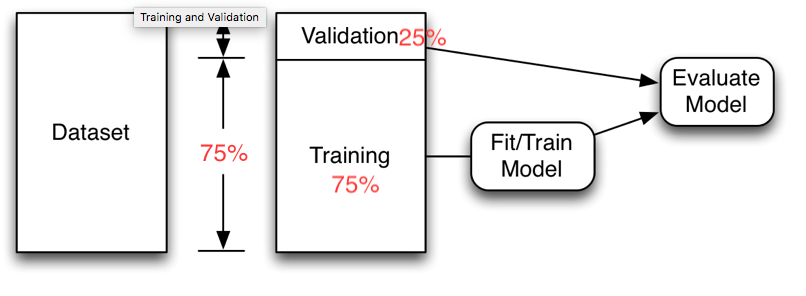
(6)

I used grid search to find the best parameters for algorithms. Random Forest algorithm depends on several hyper parameters. The most important of these are number of trees in the forest, number of features to consider at each split, max depth of each tree in the forest, the required minimum number of samples to split and the minimum number of samples demanded to be at a leaf node. Selecting the appropriate values of these parameters is an important step to get the most accurate result and there is no rule to be determined and followed. The parameters are optimized by cross-validated and grid search method. A range of parameter values were selected and I trained them. The No of Trees parameter was tested on the set of values {10, 20, 30, 50, 100, 200, 300}, No of features was tested on {1, 2, 3, 4, 5, 6, 7, 8,9} etc. The grid search method shows the score for every parameter value to be considered as chosen values. The best values which were gotten by the grid search method were 30, 7, none, 2 and 1 for the 0 of Trees and No of features, max depth of the tree, min. samples split and min. sample leaf parameters respectively

In decision tree, different parameters need to be set and examined to compare the result with other algorithms. The most important parameter of decision tree to be selected were tested in sets by grid search were: the depth of the tree, the minimum number of samples required to split an internal node, the minimum number of samples required to be at a leaf node and the number of features to consider when looking for the best split. The best result of these parameters which were gotten by grid search were none, 10, 10 and 9 for the parameters above respectively

Support vector regression is characterized by usage of kernel functions. Radial basis function is used in this work according to its lower error compared to polynomial, linear and sigmoid kernels as it is reported by (Zuo, & Carranza, 2011). Cost and gamma are kernel parameters and are required to be optimized. To assess the accuracy of support vector regression, I tested the model on different parameter companions. The values of cost parameter were in set {1, 10, 100, 1000} and the values of gamma were in set {0.01, 0.001, 0.0001}. The best performance results were when the cost equal to 10 and when gamma equal to 0.01.

Before facing The most challenge in feedforward neural network, which is how to define the number of hidden layers and the number of neurons in each hidden layer. I had split data according to some ratio between training and validation (testing) set, I used 75% of data for training and 25% for validation.



Also, choosing the suitable activation function is another challenge. A lot of studies tried to figure it out but no real role can define that. In this work, number of factors are optimized by grid-search. I set the number of hidden layers and neurons to be chosen by grid search. For one hidden layer, I specified the number of neurons by functions (Sheela, & Deepa, 2013):

𝑁ℎ = 𝑁 + 1 (12) (7)

Where N is the number of input data.

In the one hidden layer, I also tested the neural network on {4,6,8,9,10,11,12,13,16,18,20,25,30} neurons (Karsoliya, 2012).

Moreover, the grid search tested accuracy of another number of neurons in a deep network with two hidden layers and each hidden layer consists of 30 and 20 layers respectively (Karsoliya, 2012). The model of nine hidden layers which consist of 200, 160, 120, 80, 60, 40, 30, 20 neurons in each layer was also examined by grid. Another factor to be optimized by grid search is activation function. Sigmoid, Tanh, ELU, ReLU, SELU and Swish activation functions have been considered in the grid as a result of their variety and popularity. The optimization algorithms which were used in the grid search to be optimized are Stochastic gradient descent (SGD) (Zhang & Tong, 2004) and Adam (Kingma, & Ba (2014). One hidden layer with 10 neurons is selected by grid search, SELU activation is also chosen and the Adam is implemented as the preferable optimizer. I manually set the number of epochs to 100. The initial learning rate is set to 0.001. The initialization of training is Glorot uniform initialization (Glorot, & Bengio, 2010). I used no dropout and 0.9 momentum.

**Experiments and Results**

Original dataset was collected over 10 minutes and my study is examined for the prediction of 10 minutes and one-hour power consumption periods to give the utilities the ability of decision making. All independent inputs are used in the experiment according to their effect which is explained and showed in the analysis of parameters.

I applied Performance criterion to evaluate the models. I utilized two different measures: Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) that are defined as:

(8)

(9)

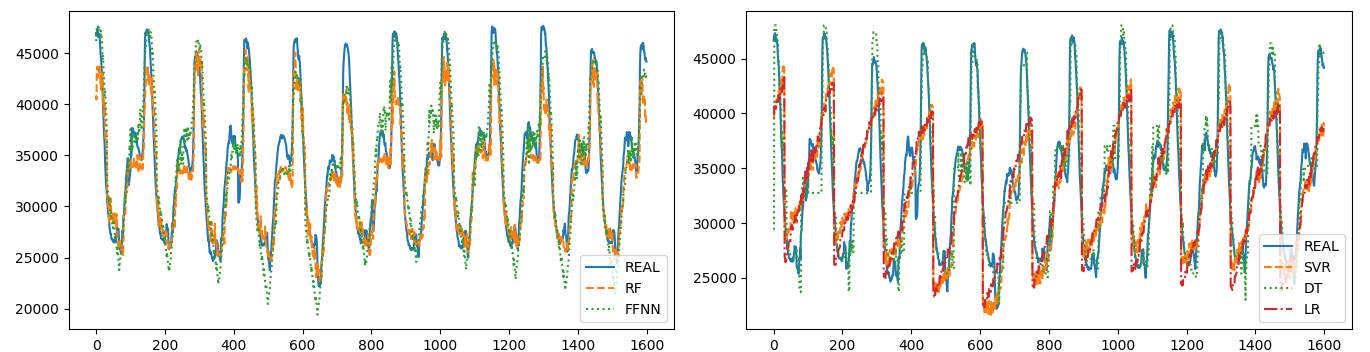
Where is predicted values, is actual values and is the average.

In order to evaluate the five models on datasets, 10-minute consumption and an hour consumption, datasets are divided into a train set and test set. Each algorithm is trained by using 75% of the data and 25% for testing. The test set is usually used to judge the models, but I also used the training set to show the ability of learning. For comparison purposes, I compared the median of 9 implementations in all models of dataset. All parameters that have been optimized by search grid before was for the comparison of dataset of 10-minute power consumption.

The experimental results for the prediction of the 10-minute period are presented in Table II. From the two performance criterions, it is shown that random forest model achieved the best results for the four examinations. Also, it is noticed that feedforward neural network achieved close results to random forest in Quads distribution network.

Table ii. RSME and MAE COMPARISON OF ALGORITHMS IN 3 DISTRIBUTIONS FOR THE 10 MINUTES POWER CONSUMPTION

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ALGORITHMS | Quads Distribution | | | | Smir Distribution | | | | Boussafou Distribution | | | | Aggregated Distribution | | | |
| RSME | | MAE | | RSME | | MAE | | RSME | | MAE | | RSME | | MAE | |
| *Train* | *Test* | *Train* | *Test* | *Train* | *Test* | *Train* | *Test* | *Train* | *Test* | *Train* | *Test* | *Train* | *Test* | *Train* | *Test* |
| RF | 671.7 | **3174.7** | 472.8 | 2663.5 | 214.1 | 2336.9 | 135.6 | 1939.6 | 594.5 | 3227.8 | 420.5 | 2475.9 | 482.3 | 4481.1 | 318.5 | 3595.3 |
| DT | 840.2 | 4613.9 | 550.7 | 3962.3 | 306.7 | 2849.8 | 179.3 | 2396.3 | 611.7 | 3543.7 | 405.9 | 2759.5 | 790.6 | 5957.3 | 490.0 | 4835.5 |
| SVR | 4092.3 | 3898.7 | 3192.3 | 3046.0 | 4205.5 | 5584.3 | 3298.6 | 4680.8 | 3334.4 | 3981.3 | 2671.6 | 3066.6 | 10821.8 | 9647.2 | 8505.6 | 7692.8 |
| FFNN | 2562.2 | 3203.6 | 1945.7 | 2601.2 | 3815.8 | 4877.8 | 2976.9 | 4007.1 | 2731.5 | 3745.6 | 2119.5 | 2965.2 | 6487.6 | 7045.9 | 4985.02 | 5583.9 |
| LR | 4404.2 | 3925.5 | 3522.1 | 3112.2 | 4068.4 | 4949.9 | 3213.4 | 4033.4 | 3142.8 | 5785.7 | 2504.1 | 4647.2 | 10687.4 | 10152.5 | 8450.9 | 8110.3 |

Fig 10. Actual vs predicted electricity power consumption of comparative models for the Quads distribution of every 10 minutes

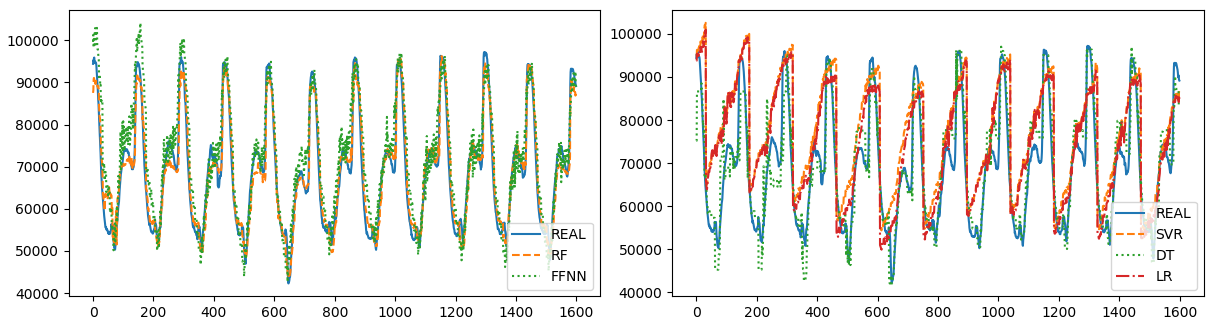


Fig 11. Actual vs predicted forecast of comparative models for the aggregation power consumption every 10 minutes

Power utilities need to have the prediction of different time periods such as hours, days, weeks, months and sometimes years for decision making and plans. In this work, I used the prediction of 10 minutes and one hour periods and it can be applied to different time periods. For the hourly prediction, I reduced the number of parameters that became ineffective such as minutes. As the value of power consumption and the parameters were changed, I needed

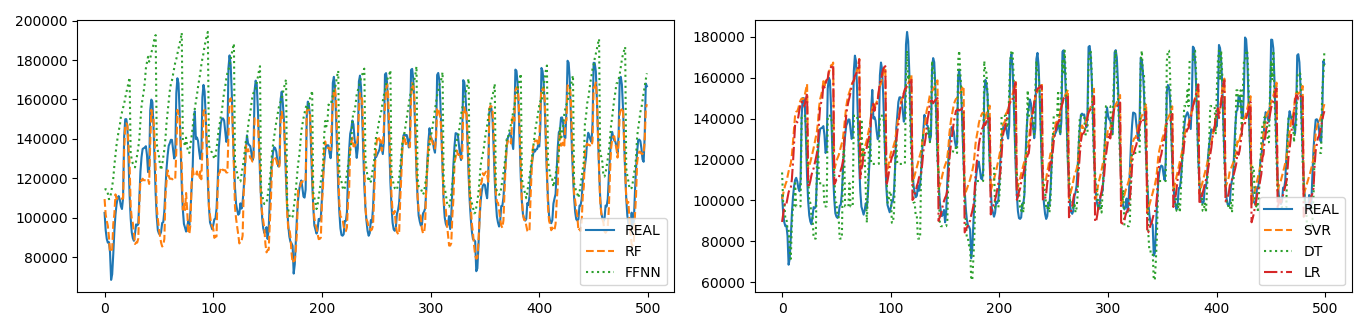
to optimize comparative algorithms again by the same optimizer method (grid search) for the same sets of parameters. Table III shows the optimized parameters of the comparative models and it is obviously different from distribution to another. Table IV shows the results of the models in each distribution and the aggregation of the three distributions. The results present that also random forest still performs the best achievement.

Table iii optimizining comparitive model parameters for every hour power consumption by using grid search

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Quads Distribution Parameter | Smir Distribution Parameter | Boussafou Distribution Parameter | Aggregated Distribution Parameter |
| RF | No of features = 3, min samples split = 2, No of Trees = 50, max depth of the tree = None, min sample leaf = 1 | No of features = 7, min samples split = 3, No of Trees = 10, max depth of the tree = None, min sample leaf = 1 | No of features = 7, min samples split = 3, No of Trees = 10, max depth of the tree = None, min sample leaf = 10 | No of features = 5, min samples split = 2, No of Trees = 100, max depth of the tree = None, min sample leaf = 1 |
| DT | No of features = 5, min samples split = 2, max depth of the tree = None, min sample leaf = 10 | No of features = 7, min samples split = 3, max depth of the tree = None, min sample leaf = 10 | No of features = 9, min samples split = 2, max depth of the tree = None, min sample leaf = 10 | No of features = 9, min samples split = 3, max depth of the tree = None, min sample leaf = 3 |
| SVR | C= 10, gamma= 0.01 | 'C': 1, 'gamma': 0.01 | 'C': 1000, 'gamma': 0.01 | C= 1, gamma= 0.01 |
| FFNN | Activation = ReLU, optimizer = SGD, batch size = 100, layers=one, neurons=25, number of epochs = 100, learning rate = 0.001 | Activation = SELU, optimizer = Adam, batch size = 350, layers=one, neurons=4, number of epochs = 100, learning rate = 0.001 | Activation = SELU, optimizer = Adam, batch size = 250, layers=one, neurons=8, number of epochs = 100, learning rate = 0.001 | Activation = SELU, optimizer = Adam, batch size = 250, layers=one, neurons=4, number of epochs = 100, learning rate = 0.001 |

Table iv. RSME and MAE COMPARISON OF ALGORITHMS IN 3 DISTRIBUTION networks FOR THE ONE HOURLY POWER CONSUMPTION

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Quads Distribution | | | | Smir Distribution | | | | Boussafou Distribution | | | | Aggregated Distribution | | | |
| **RSME** | | **MAE** | | **RSME** | | **MAE** | | **RSME** | | **MAE** | | RSME | | MAE | |
| ***Train*** | ***Test*** | ***Train*** | ***Test*** | *Train* | ***Test*** | *Train* | *Test* | *Train* | *Test* | *Train* | ***Test*** | *Train* | *Test* | *Train* | ***Test*** |
| RF | 3185.8 | 21109.7 | 2286.7 | 15442.0 | 3602.2 | 14700.9 | 2342.3 | 11955.2 | 5669.8 | 19504.1 | 4079.2 | 15777.6 | 4960.9 | 28769.3 | 3493.7 | 24033.3 |
| DT | 8218.56 | 26706.6 | 5879.6 | 23216.4 | 7364.0 | 16301.5 | 4716.7 | 13392.9 | 5766.6 | 20272.6 | 4091.4 | 16332.3 | 7487.5 | 38016.7 | 4724.6 | 30354.9 |
| SVR | 24954.1 | 26746.2 | 19707.4 | 21291.8 | 24094.4 | 29986.9 | 18886.2 | 24206.8 | 19320.6 | 23827.4 | 15435.9 | 18262.8 | 62601.5 | 56235.9 | 49188.3 | 44758.6 |
| FFNN | 19166.6 | 21127.3 | 14511.1 | 15622.3 | 20115.4 | 19845.6 | 15182.1 | 15235.3 | 20679.2 | 21873.3 | 17081.5 | 17161.9 | 46393.3 | 49175.1 | 36238.2 | 38693.8 |
| LR | 25961.7 | 23455.2 | 20798.6 | 18643.9 | 24018.5 | 29528.9 | 18985.4 | 24096.9 | 18528.9 | 34486.8 | 14767.1 | 27776.1 | 62840.1 | 59939.8 | 49712.8 | 47962.1 |

Fig 12. Actual vs predicted electricity power consumption of comparative models for the Boussafou distribution network every 10 minutes

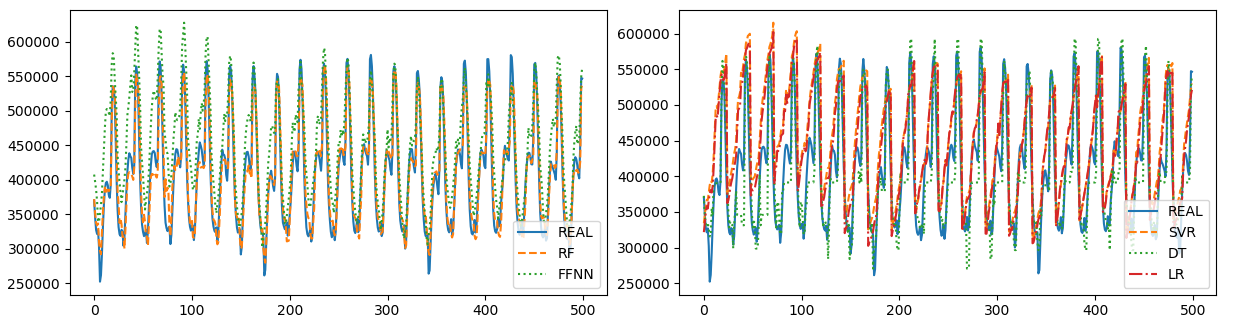


Fig 13. Actual vs predicted electricity power consumption of comparative models for the hourly aggregation of the three distributions

**Conclusion**

Accurate prediction of power consumption represents a necessary part of electricity management for sustainable, productive, and effective systems. In this paper, linear regression, decision tree, random forest, feedforward neural network, and support vector machine for regression were compared to predict the power consumption of three distribution networks in Tetouan City. The dataset that I utilized in this work is exclusive and has not been used before and is used to predict the power consumption of 10 minutes, and one hour periods. Calendar and weather predictive variables were included and it was shown that hour and temperature were the most predictive prominent variable. I optimized the comparative models by grid search to figure out the best parameters of the models. The results indicate that the Random Forest Model outperformed other models for the prediction of electricity power consumption of Tetouan City. This work can be applied efficiently for Morocco’s different power suppliers and distribution companies. However, when thinking about the data retrieved, there is a question as to whether the same results will be prevalent if taken from other countries other than Morocco.

**Environment Installation and Code-Source**

* **Environment Installation**

After installing anaconda python from <https://www.anaconda.com>, Please open Unix terminal in any Unix/IOS Environment and run the following commands.

conda install jupyter

Once Jupyter is installed, it is started with the following command:

jupyter notebook

The following packages are needed for this course:

conda install scipy

pip install --upgrade sklearn

pip install --upgrade pandas

pip install --upgrade pandas-datareader

pip install --upgrade matplotlib

pip install --upgrade pillow

pip install --upgrade requests

pip install --upgrade h5py

pip install –U scikit-learn

* **Code-Source**

###############

**libraries**

###############

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import GridSearchCV

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import Pipeline

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import Ridge

from sklearn.linear\_model import Lasso

from sklearn.linear\_model import ElasticNet

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

from sklearn.metrics import mean\_squared\_error,mean\_absolute\_error

import numpy as np

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn.preprocessing import MinMaxScaler

from time import time

from scipy.stats import randint as sp\_randint

from sklearn.model\_selection import GridSearchCV

from sklearn import tree

import pandas as pd

from keras import backend as K

from keras.layers.advanced\_activations import LeakyReLU, PReLU

from keras.models import Sequential

from keras.layers import Dense, Activation

from keras import optimizers

import keras

import math

from sklearn.metrics import r2\_score

###############

**dataset preparation**

###############

df\_train1 = pd.read\_csv('finaltetuan22.csv',parse\_dates={'dt' : ['date', 'time']} ,sep=";",infer\_datetime\_format=True,low\_memory=False, index\_col='dt')

df\_traintemp = pd.read\_csv('temp complete.csv',parse\_dates={'dt' : ['date', 'time']} ,sep=";",infer\_datetime\_format=True,low\_memory=False, index\_col='dt')

#print(df\_traintemp.head())

siz=len(df\_traintemp)

df\_train=df\_train1["Quods"].resample('10T').sum()[0:siz]

temp=df\_traintemp["Tc"].resample("10T").median()

hr=df\_traintemp["HR"].resample("10T").median()

vent=df\_traintemp["Vent"].resample("10T").median()

eg=df\_traintemp["EG"].resample("10T").median()

ediffus=df\_traintemp["Ediffus"].resample("10T").median()

#print(len(temp))

dff=pd.DataFrame({

"month": df\_train.index.month,

"day": df\_train.index.day,

"hour": df\_train.index.hour,

"minute": df\_train.index.minute,

"dayofyear": df\_train.index.dayofyear,

"weekofyear": df\_train.index.weekofyear,

"dayofweek": df\_train.index.dayofweek,

"quarter": df\_train.index.quarter,

#"Quods1":df\_train.shift(1),#["Quods"]

"temp":temp,

"hr":hr,

"vent":vent,

"eg":eg,

"ediffus":ediffus,

"Quods":df\_train #["Quods"]

})

#print(dff.head())

fig\_size = plt.rcParams["figure.figsize"]

fig\_size[0] = 20

fig\_size[1] = 6

dff.fillna(0, inplace=True)

dataset = dff.values

maxv=dataset[:,0].max()

minv=dataset[:,0].min()

print(maxv)

print(minv)

size=len(dataset)

xt=int(size\*0.75)

print(xt)

xy=size-xt

scaler = MinMaxScaler(feature\_range=(0, 1))

scaler = scaler.fit(dataset)

normalized = scaler.transform(dataset)

X\_train = normalized[0:xt, 1:]

y\_train = normalized[0:xt, 0]

X\_test = normalized[xt:, 1:]

y\_test = normalized[xt:, 0]

###############

**neural network model**

###############

def schedule(x):

if x < 40:

return 0.05

elif x < 80:

return 0.1\*0.2

elif x < 120:

return 0.1\*0.2\*0.2

elif x < 160:

return 0.1\*0.2\*0.2\*0.2

else:

return 0.1\*0.2\*0.2\*0.2\*0.5

from keras import activations

from keras.layers import BatchNormalization

def mymodel(input,layers,act):

model = Sequential()

i=0;

for l in layers:

if i==0:

model.add(Dense(l, input\_dim=input, kernel\_initializer='uniform',activation=act))

#model.add(act)

model.add(BatchNormalization())

i=i+1

else:

model.add(Dense(l, kernel\_initializer='uniform',activation=act))

#model.add(act)

model.add(BatchNormalization())

model.add(Dense(1, kernel\_initializer='uniform', activation="linear"))

return model

from keras import backend as K

def softplus(x):

return K.softplus(x)

def selu(x):

return activations.selu(x)

def softsign(x):

return K.softsign(x)

def swish(x):

return x\*K.sigmoid(x)

def eswish(x):

return 1.5\*x\*K.sigmoid(x)

def eswishb(x):

return 1.5\*x\*K.sigmoid(1.5\*x)

neron=[14]

batch=[100]

actfunc=['selu']#,'selu',swish,'elu','tanh']

opt=["sgd"]#"RMSprop","Adagrad","Adadelta","Adamax","Nadam","adam","sgd"]

for ac in actfunc:

for i in range(1,10,1):

for bi in batch:

for ne in neron:

for op in opt:

np.random.seed(7)

#print("neroun:"+str(ne)+" batch: "+str(bi)+" optimizer: "+str(op))

model = mymodel(13, [ 14,10,8], ac)

sgd = optimizers.SGD(lr=0.001, decay=1e-6, momentum=0.9, nesterov=True)

lr\_1 = keras.callbacks.LearningRateScheduler(schedule)

model.compile(loss='mean\_squared\_error', optimizer='sgd', metrics=['accuracy'])

history = model.fit(X\_train, y\_train, epochs=50, batch\_size=bi, verbose=0)

y\_train\_scaled\_fit = model.predict(X\_train)

y\_test\_scaled\_fit = model.predict(X\_test)

realvalue = y\_train \* (maxv - minv) + minv

predicvalues = y\_train\_scaled\_fit \* (maxv - minv) + minv

realvaluet = y\_test \* (maxv - minv) + minv

predicvaluest = y\_test\_scaled\_fit \* (maxv - minv) + minv

print("rmse train, "+ str(np.round(np.sqrt(mean\_squared\_error(realvalue,predicvalues)), 4))+" ,rmse test, "+str(np.round(np.sqrt(mean\_squared\_error(realvaluet,predicvaluest)), 4))+" ,mae train, "+ str(np.round(mean\_absolute\_error(realvalue,predicvalues), 4))+" ,mae test, "+str(np.round(mean\_absolute\_error(realvaluet,predicvaluest), 4))+",r2 train, "+ str(np.round(r2\_score(realvalue,predicvalues), 4))+" ,r2 test, "+str(np.round(r2\_score(realvaluet,predicvaluest), 4)))

predictnn=[x for x in predicvaluest.reshape(1,-1)[-1]]

print(predictnn)

plt.plot(realvaluet[1:500])

plt.plot(predicvaluest[1:500],linestyle='dashed',color='red')

plt.show()

################

**Randomforest**

################

for i in range(1,10,1):

linear\_reg = RandomForestRegressor(max\_features= 5, min\_samples\_split= 10, n\_estimators= 30, max\_depth= None, min\_samples\_leaf= 10)#n\_estimators= 200) #tree.DecisionTreeRegressor() #SVR(kernel='rbf', C=1e3, gamma=0.1)

reg\_scaled = linear\_reg.fit(X\_train, y\_train)

y\_train\_scaled\_fit = reg\_scaled.predict(X\_train)

realvalue = y\_train \* (maxv - minv) + minv

predicvalues = y\_train\_scaled\_fit \* (maxv - minv) + minv

#print(realvalue)

y\_test\_scaled\_fit = reg\_scaled.predict(X\_test)

realvaluet = y\_test \* (maxv - minv) + minv

predicvaluestf = y\_test\_scaled\_fit \* (maxv - minv) + minv

print("rmse train, "+ str(np.round(np.sqrt(mean\_squared\_error(realvalue,predicvalues)), 4))+" ,rmse test, "+str(np.round(np.sqrt(mean\_squared\_error(realvaluet,predicvaluestf)), 4))+" ,mae train, "+ str(np.round(mean\_absolute\_error(realvalue,predicvalues), 4))+" ,mae test, "+str(np.round(mean\_absolute\_error(realvaluet,predicvaluestf), 4))+",r2 train, "+ str(np.round(r2\_score(realvalue,predicvalues), 4))+" ,r2 test, "+str(np.round(r2\_score(realvaluet,predicvaluestf), 4)))

predict=[x for x in predicvaluestf]

print("predict")

print(predict)

real=[x for x in realvaluet]

plt.plot(realvaluet[1:500])

plt.plot(predicvaluest[1:500],linestyle='dashed',color='red')

plt.show()

print('---realvalue----------------')

print(real)

################

**Decision Tree**

################

for i in range(1,10,1):

linear\_reg = DecisionTreeRegressor(max\_features= 5, min\_samples\_split=3, max\_depth= None, min\_samples\_leaf= 3)#n\_estimators= 200) #tree.DecisionTreeRegressor() #SVR(kernel='rbf', C=1e3, gamma=0.1)

reg\_scaled = linear\_reg.fit(X\_train, y\_train)

y\_train\_scaled\_fit = reg\_scaled.predict(X\_train)

realvalue = y\_train \* (maxv - minv) + minv

predicvalues = y\_train\_scaled\_fit \* (maxv - minv) + minv

#print(realvalue)

y\_test\_scaled\_fit = reg\_scaled.predict(X\_test)

realvaluet = y\_test \* (maxv - minv) + minv

predicvaluestf = y\_test\_scaled\_fit \* (maxv - minv) + minv

print("rmse train, "+ str(np.round(np.sqrt(mean\_squared\_error(realvalue,predicvalues)), 4))+" ,rmse test, "+str(np.round(np.sqrt(mean\_squared\_error(realvaluet,predicvaluestf)), 4))+" ,mae train, "+ str(np.round(mean\_absolute\_error(realvalue,predicvalues), 4))+" ,mae test, "+str(np.round(mean\_absolute\_error(realvaluet,predicvaluestf), 4))+",r2 train, "+ str(np.round(r2\_score(realvalue,predicvalues), 4))+" ,r2 test, "+str(np.round(r2\_score(realvaluet,predicvaluestf), 4)))

predict=[x for x in predicvaluestf]

print("predict")

print(predict)

real=[x for x in realvaluet]

plt.plot(realvaluet[1:3500])

plt.plot(predicvaluestf[1:3500],linestyle='dashed',color='red')

plt.show()

################

**SVR**

################

for i in range(1,10,1):

linear\_reg = SVR(kernel='rbf', C=1, gamma=0.001)

reg\_scaled = linear\_reg.fit(X\_train, y\_train)

y\_train\_scaled\_fit = reg\_scaled.predict(X\_train)

realvalue = y\_train \* (maxv - minv) + minv

predicvaluesvm = y\_train\_scaled\_fit \* (maxv - minv) + minv

y\_test\_scaled\_fit = reg\_scaled.predict(X\_test)

realvaluetsvm = y\_test \* (maxv - minv) + minv

predicvaluestsvm = y\_test\_scaled\_fit \* (maxv - minv) + minv

print("rmse train, "+ str(np.round(np.sqrt(mean\_squared\_error(realvalue,predicvaluesvm)), 4))+" ,rmse test, "+str(np.round(np.sqrt(mean\_squared\_error(realvaluetsvm,predicvaluestsvm)), 4))+" ,mae train, "+ str(np.round(mean\_absolute\_error(realvalue,predicvaluesvm), 4))+" ,mae test, "+str(np.round(mean\_absolute\_error(realvaluetsvm,predicvaluestsvm), 4))+",r2 train, "+ str(np.round(r2\_score(realvalue,predicvaluesvm), 4))+" ,r2 test, "+str(np.round(r2\_score(realvaluetsvm,predicvaluestsvm), 4)))

predict=[x for x in predicvaluestsvm]

print("predict")

print(predict)

plt.plot(realvaluetsvm[1:500])

plt.plot(predicvaluestsvm[1:500],linestyle='dashed',color='red')

plt.show()

################

**Linear Regression**

################

from sklearn.linear\_model import LinearRegression

for i in range(1,2,1):

linear\_reg = LinearRegression()

reg\_scaled = linear\_reg.fit(X\_train, y\_train)

y\_train\_scaled\_fit = reg\_scaled.predict(X\_train)

realvalue = y\_train \* (maxv - minv) + minv

predicvalues = y\_train\_scaled\_fit \* (maxv - minv) + minv

#print(realvalue)

y\_test\_scaled\_fit = reg\_scaled.predict(X\_test)

realvaluet = y\_test \* (maxv - minv) + minv

predicvaluestf = y\_test\_scaled\_fit \* (maxv - minv) + minv

print("rmse train, "+ str(np.round(np.sqrt(mean\_squared\_error(realvalue,predicvalues)), 4))+" ,rmse test, "+str(np.round(np.sqrt(mean\_squared\_error(realvaluet,predicvaluestf)), 4))+" ,mae train, "+ str(np.round(mean\_absolute\_error(realvalue,predicvalues), 4))+" ,mae test, "+str(np.round(mean\_absolute\_error(realvaluet,predicvaluestf), 4))+",r2 train, "+ str(np.round(r2\_score(realvalue,predicvalues), 4))+" ,r2 test, "+str(np.round(r2\_score(realvaluet,predicvaluestf), 4)))

predict=[x for x in predicvaluestf]

print("predict")

print(predict)

real=[x for x in realvaluet]

plt.plot(realvaluet[1:3500])

plt.plot(predicvaluestf[1:3500],linestyle='dashed',color='red')

plt.show()

##########

**Grid search random forest**

##########

from sklearn.ensemble import RandomForestRegressor

def MSE(y\_true,y\_pred):

mse = mean\_squared\_error(y\_true, y\_pred)

print ('MSE: %2.3f' % mse)

return mse

def R2(y\_true,y\_pred):

r2 = r2\_score(y\_true, y\_pred)

print( 'R2: %2.3f' % r2)

return r2

def two\_score(y\_true,y\_pred):

MSE(y\_true,y\_pred) #set score here and not below if using MSE in GridCV

score = R2(y\_true,y\_pred)

return score

def two\_scorer():

return make\_scorer(two\_score, greater\_is\_better=True) # change for false if using MSE

clf = RandomForestRegressor()

# Utility function to report best scores

def report(results, n\_top=3):

for i in range(1, n\_top + 1):

candidates = np.flatnonzero(results['rank\_test\_score'] == i)

for candidate in candidates:

print("Model with rank: {0}".format(i))

print("Mean validation score: {0:.3f} (std: {1:.3f})".format(

results['mean\_test\_score'][candidate],

results['std\_test\_score'][candidate]))

print("Parameters: {0}".format(results['params'][candidate]))

print("")

# use a full grid over all parameters

param\_grid = {"max\_depth": [3, None],

"n\_estimators":[10,20,30,50,100,200,300],

"max\_features": [2, 3,5,7,9],

"min\_samples\_split": [2, 3, 10],

"min\_samples\_leaf": [1, 3, 10],

#"bootstrap": [True, False],

}

# run grid search

grid\_search = GridSearchCV(clf, param\_grid=param\_grid)

start = time()

grid\_search.fit(X\_train, y\_train)

print("GridSearchCV took %.2f seconds for %d candidate parameter settings."

% (time() - start, len(grid\_search.cv\_results\_['params'])))

report(grid\_search.cv\_results\_)

##########

**Grid search Decision Tree**

##########

#decision tree

#from sklearn.ensemble import DecisionTreeRegressor

def MSE(y\_true,y\_pred):

mse = mean\_squared\_error(y\_true, y\_pred)

print ('MSE: %2.3f' % mse)

return mse

def R2(y\_true,y\_pred):

r2 = r2\_score(y\_true, y\_pred)

print( 'R2: %2.3f' % r2)

return r2

def two\_score(y\_true,y\_pred):

MSE(y\_true,y\_pred) #set score here and not below if using MSE in GridCV

score = R2(y\_true,y\_pred)

return score

def two\_scorer():

return make\_scorer(two\_score, greater\_is\_better=True) # change for false if using MSE

clf = DecisionTreeRegressor()

# Utility function to report best scores

def report(results, n\_top=3):

for i in range(1, n\_top + 1):

candidates = np.flatnonzero(results['rank\_test\_score'] == i)

for candidate in candidates:

print("Model with rank: {0}".format(i))

print("Mean validation score: {0:.3f} (std: {1:.3f})".format(

results['mean\_test\_score'][candidate],

results['std\_test\_score'][candidate]))

print("Parameters: {0}".format(results['params'][candidate]))

print("")

# use a full grid over all parameters

param\_grid = {"max\_depth": [3, None],

"min\_samples\_split": [2, 3, 10],

"min\_samples\_leaf": [1, 3, 10],

"max\_features": [2, 3,5,7,9],

}

# run grid search

grid\_search = GridSearchCV(clf, param\_grid=param\_grid)

start = time()

grid\_search.fit(X\_train, y\_train)

print("GridSearchCV took %.2f seconds for %d candidate parameter settings."

% (time() - start, len(grid\_search.cv\_results\_['params'])))

report(grid\_search.cv\_results\_)

###################

**Grid search for SVR**

###################

#"support vector regression"

def MSE(y\_true,y\_pred):

mse = mean\_squared\_error(y\_true, y\_pred)

print ('MSE: %2.3f' % mse)

return mse

def R2(y\_true,y\_pred):

r2 = r2\_score(y\_true, y\_pred)

print( 'R2: %2.3f' % r2)

return r2

def two\_score(y\_true,y\_pred):

MSE(y\_true,y\_pred) #set score here and not below if using MSE in GridCV

score = R2(y\_true,y\_pred)

return score

def two\_scorer():

return make\_scorer(two\_score, greater\_is\_better=True) # change for false if using MSE

clf = SVR()

# Utility function to report best scores

def report(results, n\_top=3):

for i in range(1, n\_top + 1):

candidates = np.flatnonzero(results['rank\_test\_score'] == i)

for candidate in candidates:

print("Model with rank: {0}".format(i))

print("Mean validation score: {0:.3f} (std: {1:.3f})".format(

results['mean\_test\_score'][candidate],

results['std\_test\_score'][candidate]))

print("Parameters: {0}".format(results['params'][candidate]))

print("")

# use a full grid over all parameters

param\_grid = {'C': [1, 10, 100, 1000],

'gamma': [0.01,0.001, 0.0001],

'kernel': ['rbf']

}

# run grid search

grid\_search = GridSearchCV(clf, param\_grid=param\_grid)

start = time()

grid\_search.fit(X\_train, y\_train)

print("GridSearchCV took %.2f seconds for %d candidate parameter settings."

% (time() - start, len(grid\_search.cv\_results\_['params'])))

report(grid\_search.cv\_results\_)

**References**

Amber, K. P., Ahmad, M. W., Aslam, A., Kousar, M., Usman, M.,& Khan, M. S. (2018, August). Intelligent techniques for forecasting electricity consumption of buildings. *Energy, 157,* 886–893. Retrieved from <https://www.researchgate.net/publication/325365927_Intelligent_Techniques_for_Forecasting_Electricity_Consumption_of_Buildings>

Ahmad, M. W., Mourshed, M., & Rezgui, Y. (*2017*, July). Trees vs Neurons: Comparison between random forest and ANN for high-resolution prediction of building energy consumption — A case study of office buildings. *Energy Build, 147,* 77–89. Retrieved from <https://pdfs.semanticscholar.org/7054/6e0a15501366b679e9db8b232529a63b473c.pdf>

# Ataei, M., & Osanloo, M. (2004, January). Using a Combination of Genetic Algorithm

and the Grid Search Method to Determine Optimum Cutoff Grades of Multiple Metal Deposits. International Journal of Surface Mining, Reclamation and Environment  18(1), 60–78. Retrieved From <https://doi.org/10.1076/ijsm.18.1.60.23543>

Breiman, L. (2017, October). *Classification and Regression Trees*,. *Mathematics,* Retrieved from <https://books.google.com/books?id=gLs6DwAAQBAJ&pg=PT51&source=gbs_selected_pages&cad=3#v=onepage&q&f=false>

Breiman, L. (2001, January). Random Forests. *Machine learning*, *45*(1), 5–32. Retrieved from <https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf>

Chollet, F., & others. (2015). Keras. Retrieved from http://www.zhanjunlang.com/resources/tutorial/Deep%20Learning%20with%20Keras.pdf

Clevert, D.A., Unterthiner, T., & Hochreiter, S. (2015, November). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). Retrieved from <https://arxiv.org/pdf/1511.07289.pdf>

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support Vector Regression Machines. *Adv. NEURAL Inf. Process. Sytesyem. 9, 47*, 155–161. Retrieved from <http://papers.nips.cc/paper/1238-support-vector-regression-machines.pdf>

Edwards, R. E., New, J., & Parker, (2012, June). Predicting future hourly residential electrical consumption: A machine learning case study. *Energy Build, 49*, 591–603. Retrieved from <http://web.eecs.utk.edu/~leparker/publications/Energy-Buildings-2012.pdf>

Fumo, N., & Rafe Biswas, M. A. (2015, July). Regression analysis for prediction of residential energy consumption.[*Renewable and Sustainable Energy Reviews*](https://www.sciencedirect.com/science/journal/13640321)*, 47*, 332–343. Retrieved from [https://doi.org/10.1016/j.rser.2015.03.035](https://doi.org/10.1016/j.rser.2015.03.035" \t "_blank" \o "Persistent link using digital object identifier)

Genuer, R., Poggi, J.M., & Tuleau-Malot, C. (2010, October). Variable selection using random forests. *Pattern Recognition, 31*(14), 2225–2236. Retrieved from [https://doi.org/10.1016/j.rser.2015.03.035](https://doi.org/10.1016/j.rser.2015.03.035" \t "_blank" \o "Persistent link using digital object identifier)

Gislason, P. O., Benediktsson , J. A., & Sveinsson, J. R. (2006, March). Random Forests for land coverclassification. *Pattern Recognition, 27*(4), 294–300. Retrieved from [https://doi.org/10.1016/j.patrec.2005.08.011](https://doi.org/10.1016/j.patrec.2005.08.011" \t "_blank" \o "Persistent link using digital object identifier)

Glorot, X., & Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. 249–256. Retrieved from <http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf>

Günay, M. E. (2016, March). Forecasting annual gross electricity demand by artificial neural networks using predicted values of socio-economic indicators and climatic conditions: Case of Turkey. *Energy Policy, 90,* 92–101. Retrieved from [https://doi.org/10.1016/j.enpol.2015.12.019](https://doi.org/10.1016/j.enpol.2015.12.019" \o "Persistent link using digital object identifier" \t "_blank)

Jiang, R., Tang, W., Wu, X., & Fu, W. (2009, January). A random forest approach to

the detection of epistatic interactions in case-control studies. *BMC*

*Bioinformatics*, 10(11), S65. Retrieved from

<https://doi.org/10.1186/1471-2105-10-S1-S65>

Kane, M. J., Price, N., Scotch, M., & Rabinowitz, P. (2014, August). Comparison of

ARIMA and Random Forest time series models for prediction of avian

influenza H5N1 outbreaks. *BMC Bioinformatics, 15*(1), 276. Retrieved from

<https://doi.org/10.1186/1471-2105-15-276>

Kankal, M., & Uzlu, E. (2017, December). Neural network approach with teaching–learning-based optimization for modeling and forecasting long-term electric energy demand in Turkey. *Neural Comput, 28*(S1)*,* 737–747. Retrieved from <https://doi.org/10.1007/s00521-016-2409-2>

Karsoliya, S. (2012). Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture. *International Journal of Engineering Trends and Technology, 3(2),* 714–717 *2012*. Retrieved from http://ijettjournal.org/volume-3/issue-6/IJETT-V3I6P206.pdf

Kaytez, F., Taplamacioglu, M. C., Cam, E., & Hardalac, F. (2015, May). Forecasting electricity consumption: A comparison of regression analysis, neural networks and least squares support vector machines. *International Journal of Electrical Power & Energy System, 67,* 431–438. Retrieved from [https://doi.org/10.1016/j.ijepes.2014.12.036](https://doi.org/10.1016/j.ijepes.2014.12.036" \o "Persistent link using digital object identifier" \t "_blank)

Kingma, D. P., & Ba, J. (2014, December). Adam: A Method for Stochastic

Optimization. *Published as a conference paper at the 3rd International*

*Conference for Learning Representations, San Diego.* Retrieved from

https://arxiv.org/pdf/1412.6980.pdf

Khosravani, H., Castilla, M. D. M., Berenguel, M., Ruano, A. E.,& Ferreira, P. M. (2016, January). A Comparison of Energy Consumption Prediction Models Based on Neural Networks of a Bioclimatic Building. *Energies, 9*(1*),* 57. Retrieved from <https://doi.org/10.3390/en9010057>

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017, June). Self-Normalizing Neural Networks. Retrieved from <http://papers.nips.cc/paper/6698-self-normalizing-neural-networks.pdf>

Lin, C.J., M. A. (2003, July). A Practical Guide to Support Vector Classification Motivation. 1–29. Retrieved from <http://ntur.lib.ntu.edu.tw/bitstream/246246/20060927122852476378/1/freiburg.pdf>

Minai, A. A., & Williams, R. D. (1993, January). On the derivatives of the sigmoid. *Neural Networks, 6*(6), 845–853. Retrieved from <https://eecs.ceas.uc.edu/~aminai/papers/minai_sigmoids_NN93.pdf>

Paudel, S., Elmitri, M., Couturier, S., Nguyen, P. C., Kamphuis, R., Lacarrière, B.,& Le-Corre, O. (2017, March). A relevant data selection method for energy consumption prediction of low energy building based on support vector machine. *Energy Build, 138,* 240–256. Retrieved from [https://doi.org/10.1016/j.enbuild.2016.11.009](https://doi.org/10.1016/j.enbuild.2016.11.009" \o "Persistent link using digital object identifier" \t "_blank)

Pedregosa, F., Varoquaux , G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Cournapeau, P. & Passos, A. (2011, October). Scikit-learn: Machine Learning in

Python.*Journal of Machine Learning Research, 12*, 2825–2830. Retrieved from

<http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf>

Pombeiro, H., Santos, R., Carreira, P., Silva, C.,& Sousa, J. M. C. (2017, July). Comparative assessment of low-complexity models to predict electricity consumption in an institutional building: Linear regression vs. fuzzy modeling vs. neural networks. *Energy Build, 146,* 141–151. Retrieved from [https://doi.org/10.1016/j.enbuild.2017.04.032](https://doi.org/10.1016/j.enbuild.2017.04.032" \o "Persistent link using digital object identifier" \t "_blank)

Quinlan, J.R. (1986). Induction of decision trees.[*Learning*](https://www.sciencedirect.com/science/journal/13640321) *Machine.* *1,* (1). 81–106. Retrieved on November 11, 2018 from [https://doi.org/10.1016/j.rser.2015.03.035](https://doi.org/10.1016/j.rser.2015.03.035" \t "_blank" \o "Persistent link using digital object identifier)

Ramachandran, P., Zoph, B., & Le, Q. V. (2017, October). Searching for Activation Functions. *conference paper*. Retrieved from <https://openreview.net/pdf?id=SkBYYyZRZ>

Razi, M. A., & Athappilly, K. (2005, July). A comparative predictive analysis of

neuralnetworks(NNs), *nonlinear regression and classification and regression*

*tree (CART) models*.*Exper Systems with Applications, 47*, 332–343. Retrieved

from <https://doi.org/10.1016/j.eswa.2005.01.006>

Rodriguez-Galiano, V., Sanchez-Castillo, M., Chica-Olmo, M.,& Chica-Rivas, M (2015,

December). Machine learning predictive models for mineral prospectivity: An

evaluation of neural networks, random forest, regression trees and support

vector machines. *Ore Geology Reviews, 71*, 804–818. Retrieved from

[https://doi.org/10.1016/j.oregeorev.2015.01.001](https://doi.org/10.1016/j.oregeorev.2015.01.001" \t "_blank" \o "Persistent link using digital object identifier)

Salzberg, S. L. (1994, September). Programs for machine learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc.*,1*6(3),235–240. Retrieved on November 11, 2018 from <https://link.springer.com/article/10.1007/BF00993309>

Sheela, K. G., & Deepa, S. N. (2013, June). Review on Methods to Fix Number of

Hidden Neurons in Neural Networks. *Mathematical Problems in*

*Engineering.*1–11. Retrieved from

<https://www.hindawi.com/journals/mpe/2013/425740/>

Tüfekci, P. (2014, September). Prediction of full load electrical power output of a base

load operatedcombined cycle power plant using machine learning methods*, 45*, 126–140. Retrieved from [https://doi.org/10.1016/j.ijepes.2014.02.027](https://doi.org/10.1016/j.ijepes.2014.02.027" \o "Persistent link using digital object identifier" \t "_blank)

Xu, B., Wang, T., Chen. T., & Li. M. (2015, May). Empirical Evaluation of Rectified Activations in Convolutional Network. Retrieved from https://arxiv.org/pdf/1505.00853.pdf

Xypolytou, E., Meisel, M., & Sauter, T. (*2017*, June). Short-term electricity consumption forecast with artificial neural networks — A case study of office buildings. *IEEE Manchester PowerTech,* 1–6. Retrieved from <https://ieeexplore.ieee.org/document/7980874>

Yu, Z., Haghighat, F., Fung, B. C. M., & Yoshino, H. (2010, October). A decision tree method for building energy demand modeling. *Energy Build*, *42*, 10, 1637–1646. Retrieved on November 11, 2018 from <https://www.researchgate.net/publication/222695662_A_decision_tree_method_for_building_energy_demand_modeling>

Zhang, T., & Tong. (2004, April). Solving large scale linear prediction problems using stochastic gradient descent algorithms*.* 116. Retrieved from <https://dl.acm.org/citation.cfm?id=1015332>

Zuo, R., K. G., & Carranza, E. J. M. (2011, December). Support vector machine: A tool for mapping mineral prospectivity. *Computer. Geoscience, 37*(12),1967–1975. Retrieved from [https://doi.org/10.1016/j.cageo.2010.09.014](https://doi.org/10.1016/j.cageo.2010.09.014" \o "Persistent link using digital object identifier" \t "_blank)