Creating a Whole-building Energy Model

A Comparison of Machine Learning Methods for Predicting Building Energy Consumption

Bradley Joseph
Building Technology, MIT
Cambridge, Massachusetts, United States

I. Introduction

Buildings account for approximately 40% of all U.S. primary energy consumption, half of which is attributable to heating, ventilation, and air-conditioning (HVAC). With rapid urbanization and the need to reduce global energy consumption and emissions, governments, cities, and building owners have increased responsibility to decrease energy use. Continuous commissioning programs are an established means of improving energy efficiency, and they involve the identification and remediation of HVAC system inefficiencies, often referred to as fault detection. To measure the effectiveness of improvements to HVAC systems, energy professionals use energy baselines. These baselines are often linear-based regression models that predict facility energy consumption as a function of weather data.

While fault detection and HVAC system optimization usually save ~10% of total building energy consumption, they often require significant manual analysis and the use of rudimentary analytics tools built on rule-based detection systems. Furthermore, these analytics systems are limited in their ability to minimize the overall energy consumption of the facility because they optimize individual HVAC equipment in isolation—not the overall set of systems. Given the complex, non-linear relationship between HVAC parameter settings and energy consumption, a whole-building model that uses more than weather and energy data as an input is needed to understand and predict energy consumption. This understanding will enable the application of optimization techniques that will allow building operators to simulate the performance of their facilities and determine the optimal HVAC parameters to minimize electricity consumption.

This manuscript details two primary efforts: (1) Build several models that predict the electrical energy consumption of a building given weather and multiple HVAC system parameter inputs, and (2) Adapt those models to use only weather and energy data as inputs and compare their performance to a simple, linear regression-based baseline. For this investigation, we selected one of our clients, a large hospital located in the Bay Area, just south of San Francisco. This ~650,000 square-foot hospital has a connected medical office building and most of the facility is operational 24x7.

The hospital has complex HVAC equipment and systems that remove particulates and maintain the indoor environment at a comfortable temperature, relative humidity, and pressurization. See Figure 1 for a photo we took of the cooling towers at this hospital. These cooling towers, as well as the other HVAC systems, produce a tremendous amount of data at a 1-minute frequency. We collected this data and then applied several machine learning techniques to build models to predict the energy consumption of the facility given the weather and HVAC system configuration.

Figure 1: Image of Cooling Towers from the Hospital from which the Data Originates

II. Literature Review

A. Google Data Centers

Google has applied their machine learning expertise to model the performance of their data centers and predict power usage effectiveness (PUE) [1]. In addition to creating a model that predicts PUE, Google has developed a method to simulate their data centers under varying load conditions to determine which combination of HVAC setpoints minimizes the data center energy consumption.

Google used 19 features, several of which were identical to the ones we used in this manuscript. Some of the shared features were outside air wet bulb temperature, number of chillers running, and the number of cooling towers running. Their final machine learning model was a deep neural network composed of five hidden layers, 50 nodes/layer, and a regularization parameter of 0.001.

The 184,435 samples Google used were ~6.5x larger than the number of samples we used for this paper. 70% of the data was used for training, and the remaining 30% was used for cross validation and testing. Gao reported a test mean absolute error of 0.004 with a standard deviation of 0.005. While the published results of Google's foray into building optimization seem impressive, they have not yet commercialized their techniques.

B. Current Market Offerings

Several different companies offer services that employ machine learning to predict energy consumption and provide actionable recommendations. BuildingIQ is one such company. Based on the information available on their website, BuildingIQ's product predicts energy consumption and temperature of spaces given weather information and HVAC inputs. The product then controls the HVAC equipment by adjusting set points in a manner that minimizes energy consumption while maintaining temperatures within certain ranges [2].

Prescriptive Data is another company that uses machine learning to reduce buildings' energy consumption [3]. While the company website is lean on specifics, their tool seems to predict the thermal response of the building to weather and HVAC systems, and then it prescribes specific ramp up and down sequences at the start and end of the day, as well as during the lunch period to reduce energy consumption.

III. METHODOLOGY

This section of the text outlines the main activities we performed, which included data collection, cleaning and merging the data, feature selection, and then the implementation of the ML models. Python 3.5 was used in conjunction with several different libraries, including Pandas, Numpy, Sklearn, and Keras.

A. Data Collection

The selected hospital has a large number of equipment. 17 air handling units, 478 constant air volume boxes, 421 variable air volume boxes, 3 centrifugal chillers, 3 hydronic boilers, 6 cooling towers, 3 primary chilled water pumps, 5 secondary chilled water pumps, 3 condenser water pumps, 3 primary hot water pumps, 5 secondary hot water pumps, 8 steam boilers, and 10 fan coil units. Each of these devices has several sensors ranging in number from 15 to 250, each producing a reading every minute, or about 525,000 readings per year.

Downloading data from all systems would be time prohibitive, so we decided to focus on retrieving data from the major pieces of equipment that we anticipated are responsible for the majority of the HVAC energy consumption. Additionally, most of the systems which we omitted are downstream of the ones for which we collected data. As a result, we are optimistic that this decision will not dramatically influence the final model's ability to predict electrical energy consumption.

We collected 15 months of interval data associated to 15 major systems at the hospital: chilled water system, 3 chillers, 6 cooling towers, hot water steam system, hot water system, and 3 hydronic boilers. Next, we collected 1-minute interval data from the local weather station, located within ~2 miles of the hospital. Specifically, we gathered the following data: drybulb temperatures, relative humidity, dew point, wind speed, wind direction, pressure, and enthalpy. The combination of the HVAC and weather data is the foundation for the set of features that we used throughout this project. After collecting the large set of data for the features, we collected the electrical energy consumption data which the local utility provides at a 15-minute interval. We used this consumption data as the labels for the machine learning models.

It would be interesting to be able to incorporate natural gas consumption so that the facility's total energy consumption could be used as the labels, but unfortunately, the natural gas data was only available at a daily frequency. We were skeptical that this frequency would be sufficient for desired level of prediction capability, and we were most interested in the electricity consumption since it is the primary driver of energy costs at this facility

B. Cleaning and Merging Data

Now that we had collected the initial set of data for the features and labels, we focused on cleaning and merging the data. Given the high number of features (~500), we decided to first manually delete features we were confident should have no causal impact on facility energy consumption. This reduced the number of features to 122 across weather values and all the HVAC systems. Before investigating the potential for a more refined approach to feature selection, we wanted to clean the data. This was performed using a combination of Python 3.5 and Pandas with Microsoft Excel.

First, we removed duplicate values and non-numerical values that resulted from an error with the data collectors at the facility. Next, we pulled all features into a single .csv file and lined up the timestamps. Because the labels data is at a 15-minute interval, we resampled the features so that their 1-minute intervals were averaged to a matching 15-minute interval.

C. Data Preparation and Feature Selection

1) Data Preparation

Now that the initial set of data was cleaned and the features and labels saved into separate .csv files, we used Pandas to import the data. Next, we applied Sklearn's standard scalar to scale each feature to unit variance. Finally, we split the data into training, validation, and test sets using Sklearn's train_test_split. In the first split, 80% of the data is assigned to training and 20% test. The second application of train_test_split further subdivides the training data into a smaller training subset and a validation set which is 10% of the original 80% training set.

2) Feature Selection

We used the least absolute shrinkage and selection operator (Lasso) model from Sklearn to get a better sense for what features are most important to predicting the labels. This is a type of general linear model that uses the L1 regularization [4]. It is similar

to ridge regression, however, unlike ridge regression, lasso can set parameter values to zero. This is makes it arguably more useful for feature selection [5].

For the first implementation of Lasso, we applied it to the entire data set for three different alpha values: 0.1, 1, and 10 just to get a sense for the relative importance of the features.

D. Implementation of ML Models

We wanted to compare several different methods of predicting the facility energy consumption, and we decided to implement three different initial types of models that are well-documented for regression-type problems like ours. The three different Machine Learning models we implemented and compared were the following:

- LassoCV
- Deep Neural Net
- Random Forest Regressor

1) LassoCV

Lasso with cross validation was the first approach we implemented. As noted in the previous section, LassoCV also can be used to select features. But for this portion of the investigation, we were interested in identifying the best hyper parameter, alpha, that minimized the error on the validation set. Lasso's objective is shown in Equation 1 [4].

Equation 1: Lasso Objective Function

$$\min_{w} \frac{1}{2n} \|Xw - y\|_2^2 + \alpha \|w\|_1$$

Where

n is the number of samples

X is the set of features

w are the weights or parameters

y is the label

α is a constant hyper parameter that serves as the coefficient for the regularization component

$\|w\|_1$ is the L1 norm of the parameter vector

We implemented the LassoCV model with 20 folds, and a maximum number of iterations of 5,000. The model was first trained on the training set which represents 80% of the total data set. This training set was randomly generated using Sklearn's train_test_split function. Once the model identified the best alpha, we used the model to predict the labels (energy consumption) for the test data set. The root mean square error and CVRMSE were computed and recorded. This error is calculated by summing the squared differences between the predicted values and the actual values, normalizing by the number of samples, and then taking the square root as show in Equation 2.

Equation 2: Root Mean Square Error

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y} - y)^2}{n}}$$

Where

n is the number of samples

$\hat{y}$ is the predicted label (energy consumption)

y is the actual label

i is the index

Equation 3: Coefficient of Variation for RMSE

$$CVRMSE = \frac{RMSE}{\bar{y}} * 100\%$$

where $\bar{y}$ is the average value of the labels.

2) Deep Neural Net

Neural networks are a class of machine learning models that can be used for both regression and classification problems [6]. See Figure 2 for a generic representation of the network with the input, hidden, and output layer neurons. Together, the neurons determine how the input vector is transformed using an activation function, such as Sigmoid, ReLU, and Softmax.

The input feature vector is passed into the input layer neurons whose outputs are subsequently fed forward to the output layer of the neural network where the predicted label is given. The process of transforming the input features is called forward propagation, and simply, it entails computing the weighted sums of the inputs and applying activation functions. See Equation 4 for an example on how the inputs are transformed by a neuron.

Equation 4: Output of Neuron

$$f(z_j) = f\left(\sum_{i}^{d} x_i W_{ij} + W_{0j}\right)$$

Where

z_j represents the weighted sum of the inputs to the neuron

x_i is a feature vector

W_{ij} represents the weights (parameters)

W_{0j} is the bias or offset

$f(z_j)$ is the output of the neuron based on the selected activation function

To update the weights or parameters, stochastic gradient descent is often employed and called backpropagation. The key steps are initialize the parameters/weights, randomly take a training example, calculate the gradient of the loss function and nudge the parameters in the opposite direction of the gradient of the loss for that particular example.

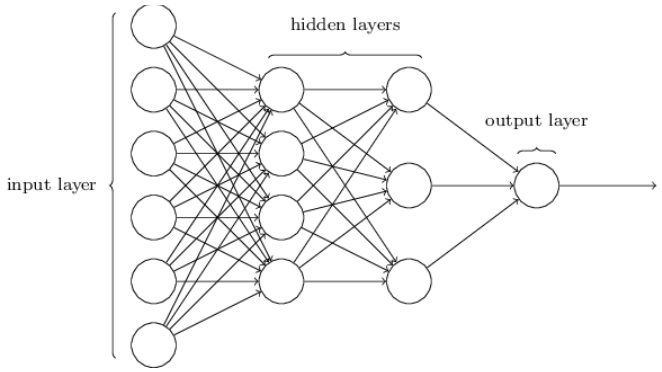

Figure 2: Generic Neural Network Structure [7]

We implemented a deep neural network using Keras 2 with TensorFlow as the backend in Python 3.5. The baseline model was inspired by the network architecture that Google cited for their deep neural network. We created the initial baseline model with 4 hidden layers, 50 neurons per layer, ReLU activation functions, 500 epochs, a batch size of 100, no dropout, and the adam optimizer.

We used the mean square error for the loss function, and returned the root mean square error. We varied different portions of the architecture, including the number of hidden layers, neurons per layer, epochs, optimizer, and dropout rate, and then noted the resulting validation error.

3) Random Forest Regressor

We implemented Sklearn's RandomForestRegressor model using Python 3.5. This is an ensemble learning method that fits classifying decision tress on sub-samples of the dataset. Each of the trees in the ensemble are built using a sample that is drawn with replacement from the training data set. The method uses averaging to reduce overfitting and improve predicative capabilities [8].

As before, we trained the model on the randomized training set representing 80% of the total data. Next, we used the model to predict the labels on the validation set and computed the root mean square error.

Before testing the model on the test set, we noted the RMSE for different numbers of trees: 1, 5, 10, 20, 25, 30, 50, 75, and 100. We then selected the model that performed the best on the validation set for use on the test set.

E. Creation of Simple Energy Baseline

As mentioned in the introduction, we also sought to adapt the three machine learning models to predicting facility energy consumption with just weather information, and then, to compare their performance to the simple, linear regression-based model their colleagues built. The general approach for doing this follows the same framework outlined in this section, with several notable changes.

For data collection, instead of collecting hundreds of thousands of samples from HVAC equipment as the features, we used the cooling degree day (CDD) and heating degree day (HDD) values that our colleagues used to create the energy baseline. Heating and cooling degree days indicate the number of hours per day when a facility is expected to require heating or cooling. While we used a 15-minute frequency for data in creating the previously described models, the CDD and HDD values were only available once per day. This led to a considerably smaller data set with only 284 training samples.

Largely, we retained the same model architecture for use with these simpler data sets. However, given how much quicker the neural network and random forest ran, we increased the number of neurons, epochs, batch size, and number of trees. As usual, we performed the tuning on the validation data set, and then used our best performing model on the test set. See Table 1 for a summary of the network architecture used.

Table 1: Summary of Network Architecture for the Deep Neural Nework

#	# HLs	Neurons/L	# Epochs	Batch Size	Optimizer	Dropout
1	4	500	15,000	100	adam	0.25

Instead of only using 75 trees for the random forest regressor, we used 500 trees, which performed slightly better on the validation set. Additionally, given the small data set, we chose to only use 15% of the data for validation and testing, as opposed to 20% before.

IV. RESULTS AND DISCUSSION

A. Feature Selection

The following table summarizes the results of the three different alpha values we used.

Alpha = 0.1	**Alpha =1.0**	**Alpha = 10**
97 features with non-zero coefficients	60 features remaining with non-zero coefficients	20 features remaining with non-zero coefficients

As expected, larger coefficients of alpha for the L1 regularization term pushed additional feature coefficients to zero. While inspecting the features with coefficients assigned to zero with a small alpha of 0.1, we noticed that the primary chilled water supply temperature set point was listed. This was initially counterintuitive because we knew that this set point is a major driver of energy consumption in buildings. However, we quickly realized that this value likely did not change over the course of the 15-month period—indeed, upon closer inspection, it had not.

The table below includes the 20 features with non-zero coefficients for a given alpha of 10.

Initial Set of Most Important Features (Variable Names)	**Description**
OARH	Outside Air Relative Humidity
ws	Wind Speed
HWR_T_Sec	Secondary Hot Water Return Temperature
PRIHWS-T	Primary Hot Water Supply Temperature
PRIHW-DP Primary Hot Water Diff Pressure	Primary Hot Water System Differential Pressure
HWS_Tspt_Sec	Secondary Hot Water System Temperature Set Point

Initial Set of Most Important Features (Variable Names)	Description
NumPrimPumpsRunning	Number of Primary Chilled Water Pumps Running
VPMP_SEC_cmd	HW System # of Secondary Pumps Commanded On
COMPRLA_x	Chiller 1 Compressor Running Load Amps
CWR_T_x	Chiller 1 Condenser Water Return Temperature
CW_FLW_x	Chiller 1 Condenser Water Flow
VTON_x	Chiller 1 Calculated Tonnage
CWR_T_y	Chiller 2 Condenser Water Return
CW_FLW_y	Chiller 2 Condenser Water Flow
CWR_T_x.1	Chiller 3 Condenser Water Return Temperature
CW_FLW	Chiller 3 Condenser Water Flow
CWS_T	Condenser Water Supply Temperature
VCHLRst	Combined Compressor Running Load Amps for Chillers
FLUE-T Flue Temperature_y	Boiler 2 Flue Temperature
HWS_T_Pri.1	Primary Hot Water System Temperature

These features with non-zero coefficients are reasonable based on our experience with buildings and HVAC systems.

B. LassoCV

The optimal alpha value to minimize the validation error was 0.28. Using this alpha value on the test data set resulted in an RMSE error of 120.65 and an R^2 of 85.86%. Given the mean value of the labels is ~2300, the CVRMSE is ~5.25%.

C. Deep Neural Net

We used 23 different configurations of a deep neural network in an attempt to minimize the validation root mean square error.

The best architecture resulted from using 3 hidden layers with 55 neurons per layer. The batch size was 100, and a dropout of 0.25 was used after each hidden layer. As with all of the scenarios, we used the ReLU activation function. This network architecture achieved an RMSE of only 88 on the validation set. More interestingly, it achieved an impressive 90.99 RMSE on the test set or a CVRMSE of 3.94%.

A few comments about the different configurations used and summarized in Table 2. Increasing the number of hidden layers over four layers had a dramatic, negative effect on the RMSE scores. We suspect that this is likely due to overfitting, despite the dropout that was used. Similarly, increasing the number of neurons per layer to over 55 had a negative effect. A small number of neurons per layer also did not perform as well as the selected architecture.

Three different optimizers were used: adam, adagrad, and adadelta. We initially selected adam from our experience with a machine learning project in MIT's 6.682 course. We wanted to try several other optimizers that automatically set the learning rate. Adagrad does this, and it is good for sparse data. Adadelta is an extension of the Adagrad optimizer, but instead of summing all previous squared gradients, this optimizer restricts the size of accumulated gradients [9].

To address overfitting, we added dropout to the neural network. 0.25 seemed to be the best value, with smaller and larger dropout rates yielding lower performance on the validation data set.

Table 2: Listing of Neural Network Architectures and the Validation RMSE

#	# HLs	Neurons/L	# Epochs	Batch Size	Optimizer	Dropout	Val RMSE
1	4	50	500	100	adam	0	99.26
2	5	50	500	100	adam	0	1387.13
3	3	50	500	100	adam	0	95.27
4	3	50	500	100	adadelta	0	105.36
5	3	50	500	100	adagrad	0	111.94
6	3	50	500	100	adam	0.25	90.68
7	3	50	500	100	adam	0.25	93.67
8	3	25	500	100	adam	0.25	92.97
9	3	15	500	100	adam	0.25	119.69
10	3	20	500	100	adam	0.25	109.1
11	3	30	500	100	adam	0.25	234.32
12	3	35	500	100	adam	0.25	94.83
13	3	40	500	100	adam	0.25	94.45
14	3	45	500	100	adam	0.25	97.42
15	3	55	500	100	adam	0.25	**88.02**
16	3	60	500	100	adam	0.25	102.05
17	3	100	500	100	adam	0.25	97.95
18	3	55	1000	100	adam	0.25	95.81
19	3	55	500	100	adam	0.3	97.51
20	3	55	500	100	adam	0.35	112.63
21	3	55	500	100	adam	0.4	95.84
22	3	55	500	100	adam	0.1	104.51
23	10	55	500	100	adam	0.25	305.75
23	7	55	500	100	adam	0.25	151.18

D. Random Forest Regressor

The validation error expectedly decreased with an increasing number of trees as illustrated in Table 3. Given our understanding that decision trees and random forests cannot overfit, this result is not surprising. We chose to use 75 trees for the model used on the test set, as this is the model that performed the best on the validation set.

It should be noted that almost all the validation errors (RMSE) for the random forest regressors were smaller than the validation errors using the deep neural network.

When applied to the test set, the Random Forest model achieved a test RMSE of only 82.66, or 3.24% CVRMSE.

Table 3: Random Forest Validation Errors

#	Model	# Trees	Val RMSE	CVRMSE
1	Baseline	1	133.46	5.78%
2	Add trees	5	85.49	3.70%
3	Add trees	10	79.86	3.44%
4	Add trees	20	79.49	3.44%
5	Add trees	25	75.40	3.27%
6	Add trees	30	77.41	3.36%
7	Add trees	50	75.25	3.26%
8	Add trees	75	73.81	3.19%
9	Add trees	100	74.77	3.24%

E. Summary of Results

Table 4: Summary of Test Error Results

Model	RMSE	CVRMSE
LassoCV	117.9	5.11%
Deep Neural Network	91.0	3.94%
Random Forest Regressor	82.7	3.58%

The deep neural network and random forest repressor performed better than the best LassoCV model with RMSE values more than 22% smaller. However, the difference in the performance of the deep neural net and the random forest regresssor was not as dramatic with ~9% difference between the test root mean square errors. Despite the lower performance of the LassoCV model, we liked the ease of interpretability of it.

F. Creation of Simple Baselines

We adapted the LassoCV, deep neural network, and random forest models to create the simple energy baselines that use CDD and HDD as the features. Given the better performance of the deep neural network and random forest approaches, we expected these to perform the best, again. We ran each model several times and selected the best performing one with the understanding that the train_test_split function uses random data sets. While the random forest regressor still performed the best, the LassoCV outperformed the deep neural net. Given the simple nature of the data set, perhaps the deep neural network and the random forest regressor do not offer as significant of performance improvements over the lasso model.

Table 5: Best Test Errors by Model for the Simple Energy Baselines

Model	RMSE	CVRMSE
Baseline Model	2042.9	3.80%
LassoCV	1639.0	3.04%
Deep Neural Network	1833.6	3.41%
Random Forest Regressor	1510.7	2.81%

Arguably more interesting is a comparison to the baseline model that our colleagues developed using a simple linear regression. Our three models' performance is not that different as can be seen in the table above, but upon further investigation, the former RMSE value of 2,042 was the error of the linear model on the actual training data set that our colleague used! So not only did our model perform better, it performed better on the actual test data set—not just the training set. We will be working with our colleagues to improve the process of creating future energy baselines as a direct result of this work.

V. Conclusions and Future Work

The original motivation for this work was to identify opportunities to use machine learning to improve the delivery of services to our client hospital through the better prediction of energy consumption. Our initial results suggest that machine learning models can be used to predict building energy consumption with the same, or lower errors than simple linear regression models. This manuscript detailed efforts to approximate the energy consumption of a large hospital given weather information and HVAC system parameters, as well as the application of this work to a simpler energy baseline.

Three different machine learning approaches were applied and the results were compared. The random forest regressor performed the best, followed by the deep neural network. However, the LassoCV model did boast the easiest interpretability. For future work, we would like to implement additional machine learning models to improve our understanding of their relative performance. Additionally, we would like to collect data from more of the hospital's HVAC systems to determine if the performance of our existing models can be improved by using additional features.

Our ultimate goal is to build a tool that will enable us to provide more nuanced recommendations to our client that will minimize their energy consumption for a given weather forecast. To do this, we first need to identify the major features that are adjustable, as well as their parameter restrictions. For example, there are operational constraints on HVAC equipment, as well as temperature, humidity, and pressurization requirements for different spaces. Given a specific weather forecast, we would expect the tool to provide recommendations on the specific adjustments that should be made to the HVAC equipment, as well as an indicator of the confidence that all building zones will remain within established temperature, humidity, and pressurization bounds. Naturally, this necessitates the creation of models that not only predict energy consumption, but temperature, humidity, and pressurization.

Acknowledgments

We would like to thank Vikas Garg who provided thoughtful guidance and machine learning expertise throughout this project. We would also like to acknowledge Michael Jermann for his help with the Python implementations and Manoj Selvakumar and Accenture Connected Buildings who helped provide the data needed for this paper.

References

[1] J. Gao, 2014. [Online]. Available: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42542.pdf.

[2] "Overview of Predictive Energy Optimization," 2017. [Online]. Available: https://buildingiq.com/peo-video/.

[3] Prescriptive Data, "Product Data Sheet," 2 May 2017. [Online]. Available: http://web.nantum.io/wp-content/uploads/2016/11/Nantum-Product-Data-Sheet.pdf.

[4] R. Tibshirani, "Modern regression2: The lasso," March 2013. [Online]. Available: http://www.stat.cmu.edu/~ryantibs/datamining/lectures/17-modr2.pdf.

[5] scikitlearn, "sklearn.linear_model.Lasso¶," 2016. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html.

[6] R. B. Tommi Jaakkola, "Introduction to Machine Learning Draft Notes by Topic," 2016.

[7] M. Nielsen, "Chapter 1: Using Neural Nets to Recognize Handwritten Digits," in *Neural Networks and Deep learning*, Determination Press, 2015.

[8] scikitlearn, "sklearn.ensemble.RandomForestRegressor," 2016. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.

[9] S. Ruder, "An overview of gradient descent optimization algorithms," January 2016. [Online]. Available: http://sebastianruder.com/optimizing-gradient-descent/index.html#adagrad.

[10] R. T. J. F. Trevor Hastie, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer, 2009.