



Text Regression Analysis for Predictive Intervals using Gradient Boosting

Alexander I. Iliev^{1,2}(✉), Ankitha Raksha¹

¹ SRH Berlin University, Charlottenburg, Germany
ankitha.raksha97@gmail.com

² Institute of Mathematics and Informatics, Bulgarian Academy
of Sciences, Sofia, Bulgaria
ailiev@berkeley.edu

Abstract. In this paper, we aim to explore text data analysis with the help of different methods to vectorize the data and carry out regression methods. At present, we have a lot of text categorization techniques, but very few algorithms are dedicated to text regression. But many regression models predict only a single estimated value. However, it's more efficient to predict a numerical range than a single estimate, since we can be more sure that the true value will be within the range rather than considering the estimated value as the true value. We aim to combine both these techniques in this paper. The goal of this paper is to collect text data, clean the data, and create a text regression model to find the best-suited algorithm that could be used in any situation, through the use of quantile regression.

Keywords: text analysis • tf-idf • word2vec • GloVe • text regression • word vectorization • quantile • gradient boosting • natural language processing

1. Introduction

With the help of Natural Language Processing (NLP) and its components, we can organize huge amounts of data. NLP is a powerful AI method for communicating with an intelligent system using natural language. We can perform numerous automated tasks and solve a wide range of problems, such as automatic summarization, machine translation, entity recognition, speech recognition, and topic segmentation. Since machine learning algorithms are not able to understand text, we need a way to convert the text into numbers so that the algorithm can understand the data that is sent to the model. After the data is pre-processed and cleaned, we perform a process called vectorization. In machine learning, most algorithms work with numerical data. Therefore, we need to find a way to convert the text data into numbers so that it can be incorporated into the machine learning models. This is called word vectorization, and the numerical form of the words is called vectors.

To gain the confidence of decision makers, it is often not necessary to present a single number as an estimate, but to provide a prediction range that reflects the uncertainty inherent in all models.

2. About the Data

The data was collected from Plentific GmbH in the last 2 years which accommodates around 170,000 records. The data consisted of various textual data that described the specificity of a home repair job that has been posted on the marketplace such as category, service, type, job description, emergency tag type which were the feature vectors for predicting the price for that repair job which is our target value.

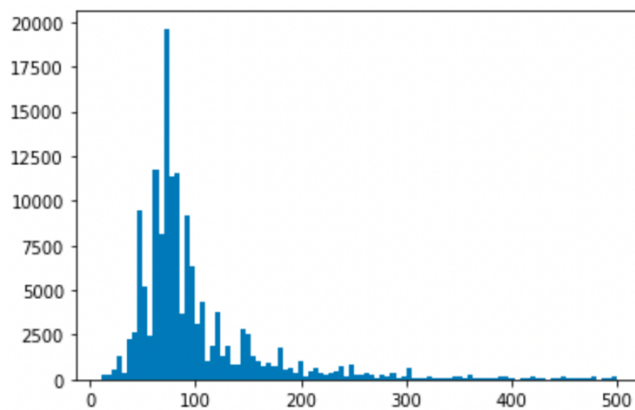


Fig. 1. Histogram of the repair job and their price value for the maximum value of 500

From the histogram shown in figure 1 of the repair job and their prices, we see that the distribution of repair job prices has been spread from a very little amount as in 10 pounds to a very large amount of 500 pounds.

2.1.Data Preparation

Data preparation involves cleaning the data, which is one of the simplest yet most important steps in the entire data modelling process. How well we clean and understand the raw input data and prepare the output data will have a big impact on the quality of the results, even more than the choice of model, model tuning, or trying different models. All text data in its original form contains a lot of unnecessary data, which we call noisy data or dirty data, and which must be cleaned. Failure to do so can result in skewed data that ultimately leads to the organization making poor decisions.

Cleaning text is particularly difficult compared to cleaning numbers because we cannot perform any of the usual statistical analysis that we can with numerical data. Steps in text cleaning:

- a. Removal of null values in the data.
- b. Convert all the words into lower case.
- c. Remove numbers.

- d. Remove newline character
- e. Remove any HTML tags
- f. Remove punctuations, limit to a certain word/s where there is an actual message.
- g. Lemmatization: bringing a word into its root format

To remove outliers from the dataset and to reduce the variance among different repair jobs, we have tried to limit the accepted price value from 45 to 125 pounds as in figure 2. Initially, we had almost 170,000 data records but after removing outliers we have 115,000 data left in our dataset, which comprises almost 70% of data.

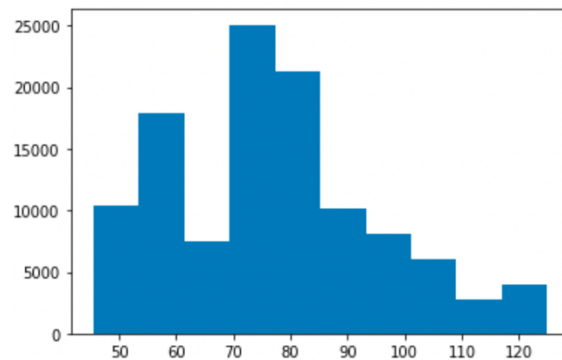


Fig. 2. Histogram of the repair job and their price value for the maximum value of 125 and minimum of 45

3. Literature Survey

For our case, we had to understand insights from previous works for text regression. One such study includes considering different text corpora to predict the age of the author that's written it using a linear model and text features [1]. There was another study that's carried out that used much more complex modeling like Convolution neural networks (CNN) for text regression for financial analysis [2]. A similar study was done for financial documents pertaining to stocks volatility to analyze different documents and predict risk using Support Vector Regression [3]. In another study, the authors have proposed a text regression model based on a conditional generative adversarial network (GAN) [4] the model works with unbalanced datasets of limited labeled data, which align with real-world scenarios [5]. Having both text data and metadata sent into the same model is a unique concept of having two different data structures being combined, this was beautifully explained in predicting revenues for movies [6]. This paper shows a non-linear method based on a deep convolutional neural network [7]. Another paper suggests the use of logistic regression and its advantages to text categorization [8].

TF-IDF has its mathematical implications that provide the notion of the

probability-weighted amount of information (PWI). To understand how TF-IDF weights can help give a relevant decision, the authors have come up with a model that combines the local relevance with document-wide relevance decision [9]. Another method for word representation is word2vec, which is originally done by the Google developers to have an efficient word representation in an N-d vector space and have illustrated two different models: Skip-gram and Continuous bag of words [10]. By extending this study, the author of the paper has discussed emerging trends that come with word2vec, with all different operations that can be carried out on words to find out relationships [11] and how it can be applied to big text contents [12]. Like word2vec we have GloVe, which is short for Global Vectors, founded by Stanford researchers who have shown that the result is a combination of global matrix factorization and local context window methods [13]. Creating a model on top of the GloVe baseline model has been done to get better results [14]. There is also a study which shows a way to combine both word2vec and GloVe model [15].

Decision trees suit better to fit nonlinear non-normal data [16], and a combination of many models together in ensembling learning increases the accuracy of the model [17]. Gradient Boosting is a greedy function where it is an inclusive model with loss functions like least square, least absolute deviation, and Huber-M [18]. This paper provides conditions that point to a quantile regression model that proposes the conversion of the data that disregards the fixed effects by assuming of location shifters [19]. There were many attempts for predicting intervals like fuzzy intervals [20], and machine learning approach for artificial and real hydrologic data sets [21].

4. Word Vectorization

Word vectorization schemes are needed in machine learning for the simple reason that the encoding scheme we're most familiar with as humans and the alphabet tell us very little about the nuances of the word it represents. The need for good embedding schemes increases because we cannot simply feed a network the binary data from ASCII or something that's been hot coded based on an extensive vocabulary because these soon balloon into unacceptably large dimensions that no neural network can easily handle. Neural networks for natural language processing are well known for the fact that their performance depends almost exactly on how good their word embedding is. The goal of word embedding, then, is to solve all these problems and provide the model with a learning method that allows it to place words in hyperspace, where their position tells it something about the word and its relationship to all the other words in its vocabulary.

The goal of word embedding is:

- To reduce dimensionality
- To use a word to predict the words around it
- Inter-word semantics must be captured

There are many word vectorization techniques out there, but we want to focus on the following ways:

4.1. Term frequency-inverse document frequency (TF-IDF)

Term frequency-inverse document frequency (TF-IDF) is a method whose main purpose is to introduce the concept of semantic meaning of words and to consider the meaning of words in a document. This is done through a statistical analysis of word frequencies.

This is represented by the following formula: $TF(w) * IDF(w)$.

Where TF stands for term frequency and IDF stands for inverse document frequency and is formulated as follows:

$TF(w) = (\text{number of occurrences of a word in the document}) / (\text{total number of words in the document})$.

$IDF(w) = \log (\text{number of documents} / \text{number of documents containing the word } w)$

4.2. Word2Vec

Word2Vec is a method for generating word embeddings. It converts words into vectors, and with vectors we can perform several operations, such as adding, subtracting, and calculating distance, so that a relationship between words is preserved. Unlike TF-IDF, word2vec and other encoding methods use a neural network model that provides an N-d vector for the words. Training the word2vec model is also very resource intensive, as it requires a large amount of RAM to store the vocabulary of the corpus. In simple terms, word2vec tries to consider words with similar contexts as similarly embedded. Word2vec, or any other vector embedding method using a neural network trained with different corpora, has "rightly" gained knowledge about words through the context of similarity with surrounding words.

There are two types:

- a. Continuous bag of words (CBOW): a word is predicted based on the surrounding words and as an example shown in figure 3, we are trying to predict the word 'Jumps' from all the other words in that sentence.

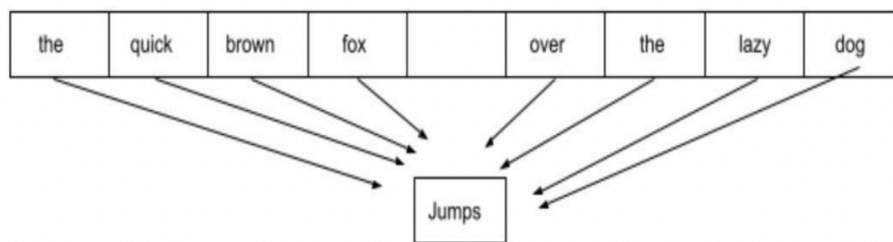


Fig. 3. Continuous bag of words example

- b. Skip-gram: one word is used to predict the surrounding words and as an example shown in figure 4, we are trying to predict all the other words in that sentence from the word 'Jumps'.

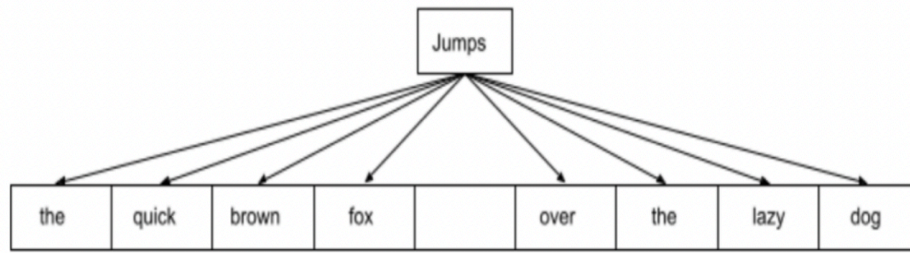


Fig. 4. Skip-gram model example

4.3. GloVe

GloVe is an abbreviation for Global Vector. The difference between Word2Vec and GloVe is that in Word2Vec we only considered the local presence of a word in the dataset. GloVe is a word representation scheme that aims to extract semantic relations between words in their embeddings. Thus, GloVe aims to capture the meaning of the word in the embedding by explicitly recognizing the probabilities of co-occurrence, and this is empirically demonstrated in the paper, as in table 1:

Table 1. GloVe co-occurrence matrix example.

	k = 'solid'	k = 'gas'	k = 'water'	k = 'fashion'
P(k 'ice')	1.90E-04	6.60E-05	3.00E-03	1.70E-05
P(k 'steam')	2.20E-05	7.80E-04	2.20E-03	1.80E-05
P(k 'ice') / P(k 'steam')	8.90E+00	8.50E-02	1.30E+00	9.60E-01

The words 'ice' and 'steam' are compared to various probe words like 'gas', 'water', 'solid', 'fashion'. Now we can see that the word 'ice' is more related to 'solid' than it is to 'gas' and the converse is true for 'steam' as seen by the ratios calculated in the last row. Both terms have similar large values with water, and on the other hand have very small values in the context of the word 'fashion' [13]. All these aims to argue the point that embedding should be built not on just the word probabilities but the co-occurrence probabilities within the context and this is what is going to be in the loss function. The essence of GloVe is to build a matrix from these probabilities and subsequently learn a vector representation of each word. And here we arrive at the loss function in its entirety

$$J = \sum f(X_{ij})(w_i^T w'_j + b_i + b'_j - \log(X_{ij}))^2$$

Where $f(X_{ij})$ = weighting function

$w_i^T w'_j$ = dot product of input/output vectors

$b_i + b'_j$ = bias term

X_{ij} = number of occurrences of j in context i

5. The intuition of text regression

We have all heard of text classification, where a given text or sentence is divided into different classes, thus predicting a class name when a new text is present. But in today's world, there is very little structured data, the rest is unstructured, and this is where text data comes in. There are cases where we need to predict a number for a given text, such as a property description and predicting the price of the property, or an article and predicting how many "likes" it can get, or an article (or story) and predicting the age of the author based on the vocabulary used [1]. These all seem to be real-world examples, and we may need to look ahead a bit to grasp the importance of text regression.

5.1. Gradient boosting regressor

Gradient Boosting Regression is a boosting algorithm with serial implementation of decision trees. We trained two models, one for predicting a lower bound and the other for the upper bound, by providing values for the quantile or "alpha" to create an interval. Predicting a numerical range by this method can be very useful in situations where we cannot rely one hundred percent on a single number, but rather on a range. This is very important in our case because with textual data such as the repair order description, there are many variations within the description that can only be properly understood by the contractor based on their experience. For this reason, we chose to use quantile regression rather than just linear regression.

Gradient boosting regressors have been shown to fit and perform better on complex data sets. Another reason for choosing this algorithm is that it provides intrinsic quantile prediction functionality through its loss function.

5.2. Quantile Regression Model

Before we dive into understanding a quantile regression model, we must first understand the meaning of quantiles. In general, quantiles are just lines that divide the data into equal groups. Percentiles are simply quantiles that divide the data into a hundred equal groups. For example, when we talk about a 75th quantile, it generally means that 75% of the data is below that point. The quantiles are only known from the distribution of the data provided to the model.

Linear regression, or so-called ordinary least squares (OLS), assumes that the relationship between the input variable X and the output label Y can be transformed into a linear function.

$$Y = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \dots + \theta_p X_p + \epsilon$$

Where Y is the dependent variable, $X_1..p$ are the independent variables, $\theta_0..p$ are the coefficient and ϵ is the bias term. The objective loss function is a squared error represented as:

$$L = (y - X\theta)^2$$

Where y is the actual value and $X\theta$ is the predicted value.

With quantile regression, we have an extra parameter τ , which talks about the τ th quantile of our target variable Y that we're interested in our model, where $\tau \in (0,1)$ and our loss function becomes:

$$L = \tau(y - y'), \text{ if } y - y' \geq 0$$

or

$$L = (\tau - 1)(y - y'), \text{ if } y - y' < 0$$

Where τ is quantile,
 L is the quantile loss function, y is the actual value,
 y' is the predicted value.

We want to penalize the loss function when the percentile is low, but the prediction is high, and when the percentile is high, but the prediction is low. What is meant by this is that we use the quantile loss to predict a percentile within which we are confident that the true estimate lies. The first condition in the quantile loss function indicates when $y - y' \geq 0$, which means that the value predicted by our model is low, which is good if we want to predict the lower percentiles, but we want to penalize the loss if the predicted value is much higher than the value. $y - y' < 0$ indicates that our prediction is high, which is good for higher percentiles, but we want to penalize the loss if the predicted value is much lower than the true value.

The quantile loss differs depending on the quantile evaluated, so negative errors are penalized more when we specify higher quantiles, and positive errors are penalized more for lower quantiles. Now that we know quantile regression, we need to see how to apply it in our scenario. After some research, we saw that gradient boosting regression has an option to use quantile regression, which simplifies the process. This is easily accomplished by specifying `loss="quantile"` with the desired quantile in the `alpha`.

5.3. Benchmarking the Model

In order to evaluate the model performance, and how well the interval range fits the true value, we have decided to consider the frequency of the true value captured in the range. In other words, the accuracy of the model is the number of times the actual priced value is in the predicted interval over the number of records of data.

5.4. Model comparisons

We have considered only three popular methods of text vectorization for now i.e. TF-IDF, Word2Vec and GloVe. In table 2 we show the statics of those methods:

Table 2. Method comparison.

Methods	Test Accuracy	Training time
TF-IDF	75.10%	3 min
Word2Vec	75.50%	10 min
GloVe	74.60%	6 min

The values given in the table were trained with gradient boosting models with quantile values 0.8 and 0.2 to understand how different methods behave. We also see that the more time required to train the model, the higher the accuracy. However, the trade-off is always between the accuracy and the computational time because if training the data in the production environment is computationally expensive, it will be questioned because it would cost a lot of money on the production side as well. There, it is more optimal that we achieve good accuracy with minimal cost.

Note that the training time may vary depending on the size of the dataset and the instances used to train the models. In our case we used a GPU instance g4dn.xlarge, but again it is all a matter of resources and optimization. To keep the training time for the different tests as low as possible, we decided to use the model with the TF-IDF vectorization method to test the models with different quantile values, as shown in table 3:

Table 3. Quantile comparisons.

Lower Quantile	Upper Quantile	Testing Accuracy	Interval Difference [in pounds]
0.2	0.8	75%	28
0.18	0.82	78%	30
0.15	0.85	82%	34
0.1	0.9	89%	44

The above table shows the data with different quantile levels and the test accuracy, as well as the interval difference (in pounds) between the lower and upper limits. From the above values, we can see that there is a clear trade-off between the accuracy achieved and the interval difference, as it is very easy to capture more of the actual value within the range by simply extending it further. However, this is not our goal, as a large range is not suitable in the real world. What we need is a decently narrow range with good detection accuracy.

5.5. Hyperparameter tuning

To understand the model even better, and since we had a large enough data set, we tried to increase the ratio of training to test as test data from 20% to 40%. The result of this experiment is quite interesting, as it yielded similar accuracy to the 85% to 20% split. Before reaching a conclusion, we had to test different values for the parameters of the gradient boosting regression. We had chosen `n_estimators` as a parameter to vary and compare the results. `n_estimators` show the number of trees formed in the model. The following table 4 shows different values for `n_estimators`:

Table 4. `N_estimators` hyperparameters.

<code>n_estimators</code>	Test Accuracy
40	82.66%
60	82.78%
80	82.86%

100	83.20%
120	83.08%
140	82.93%
160	82.92%

We can conclude that the change in accuracy is very minimal and that it increases/decreases in decimal points. The accuracy increases around 100 trees and then decreases again. A special feature of the gradient boosting trees is that they are based on the learning rate and the average price value. Since the target value is mostly the same even for different repair jobs, this also affects the overall value when we try to build the trees.

6. Result

In table 5, we see an example of the prediction of the lower and upper limits for the repair order, which are called "lower" and "upper" values in the diagram, while "price" is the actual value in the test data.

Table 5. Results.

id	price	upper	lower
1	90	58.821936	90.234450
2	85	49.699136	97.002349
3	84	59.973129	94.544094
4	80	65.785190	93.253945
5	95	60.164752	90.126343

For our model, we need to show the prediction error on the test set. Measuring the error from a prediction interval is more difficult than predicting a single number. We can easily determine the percentage of cases where the true value was captured in the interval, but these values can easily be increased if we increase the interval boundaries. Therefore, we need to show the absolute error calculations to account for this, as in table 6:

Table 6. Absolute error calculations.

	absolute_error_lower	absolute_error_upper	absolute_error_interval
count	45343.000	45343.000	45343.000
mean	18.749340	20.576137	19.662739
std	15.600487	13.186537	6.646751
min	0.000465	0.000292	2.152931
25%	6.298666	9.967990	14.898903
50%	13.917017	19.058457	18.511
75%	27.664566	29.542923	22.241346
max	75.874350	67.695534	57.763181

We see that the lower prediction has a smaller absolute error (with respect to the median) than the upper prediction. It is interesting to note that the absolute error for the lower bound in terms of the mean and standard deviation is almost the same

as the absolute error for the upper bound, which in turn shows that the upper bound and the lower bound are almost equally far from the true value.

6.1. Inference

From the tests performed above, we can conclude that the accuracy of the model is not drastically changed even when increasing/decreasing the test-train size or the `n_estimator` values, because the variance in the data set is very small. To get this right, we added the outliers back into our data set to see if there was any additional variance added to the data set. But that did not change anything, and the model returned about 78 percent on the test data. This confirms that the variance in the data set is very small and that the power changes only slightly with each change.

7. Scope of future improvements

Building a model never has an end, as there will always be new requirements, adjustments and improvements that need to be reconciled with the product development process. The main concern is to keep re-training the model as we collect more and more data over time. The reason for this is that new data brings its own variance and deviation that could be affected or incorrectly predicted by using a previous model.

We have listed down use cases of improvements for this model in the future:

- a. Try it with various other machine learning algorithms or neural networks by altering the loss function to give an interval instead of a single estimated value.
- b. Add location data in terms of GPS coordinates to see how the price changes with location

8. Conclusion

The main goal of this work was to experiment with different word vectorization methods that can be used in the gradient boosting engine to take advantage of its inherent functionality of quantile regression and to find out how text data behaves with respect to regression.

From all the experiments that have been conducted on this topic, we can conclude that:

- We have been able to successfully show that we can predict numerical intervals for arbitrary text data. When a machine learning model predicts a single number, it creates the illusion of a high degree of confidence in the entire modelling process. However, since one of the models is only a rough approximation, we need to convey the uncertainty in the estimates.
- Even though the tree models can be robust, it always depends on the problem we want to solve and the input data for the model. Comparison of different methods TF-IDF, Word2Vec, GloVe for converting words to vectors has helped to obtain important text features from the input data.

References

1. Nguyen, D., Smith, N. A., & Rose, C. P. (2011). Author Age Prediction from Text using Linear Regression. *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*.
2. Dereli, N., & Saraclar, M. (2019). Convolutional Neural Networks for Financial Text Regression. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*.
3. Kogan, S., Levin, D., Routledge, B. R., Sagi, J. S., & Smith, N. A. (2009). Predicting Risk from Financial Reports with Regression. *Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL*.
4. Aggarwal, A., Mittal, M., & Battineni, G. (2020). *Generative adversarial network: An overview of theory and applications*. Elsevier.
5. Li, T., Liu, X., & Su, S. (2018). Semi-supervised Text Regression with Conditional Generative Adversarial Networks.
6. Joshi, M., Das, D., Gimpel, K., & Smith, N. A. (n.d.). *Movie Reviews and Revenues: An Experiment in Text Regression*. Language Technologies Institute.
7. Bitvai, Z., & Cohn, T. (2015). Non-Linear Text Regression with a Deep Convolutional Neural Network. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, 180–185.
8. Genkin, A., Lewis, D. D., & Madigan, D. (n.d.). Sparse Logistic Regression for Text Categorization.
9. Chung Wu, H., Fai Wong, K., & Kui, L. K. (2008). Interpreting TF-IDF Term Weights as Making Relevance Decisions. *ACM Transactions on Information Systems*.
10. Mikolov, T., Chen, K. C., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space.
11. Church, K. W. (2016). Emerging Trends Word2Vec . *Natural Language Engineering* , 155– 162.
12. Ma, L., & Zhang, Y. (2016). Using Word2Vec to Process Big Text Data.
13. Pennington, J., Socher, R., & Manning, C. D. (n.d.). GloVe: Global Vectors for Word Representation.
14. Ibrahim, M., Gauch, S., Gerth, T., & Cox, B. (n.d.). WOVe: Incorporating Word Order in GloVe Word Embeddings.
15. Shi, T., & Liu, Z. (2014). Linking GloVe with word2vec.
16. Chowdhury, S., Lin, Y., Liaw, B., & Kerby, L. (2021). Evaluation of Tree Based Regression over Multiple Linear Regression for Non-normally Distributed Data in Battery Performance.
17. Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. *First International Workshop*, (pp. 21-23). Italy.
18. Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine.
19. Canay, I. A. (2011). A simple approach to quantile regression for panel data. *The Economist Journal*, 368-386.
20. Sáez, D., Ávila, F., Olivares, D., Cañizares, C., & Marín, L. (2015). Fuzzy Prediction Interval Models for Forecasting Renewable Resources and Loads in Microgrids. *IEEE Transactions On Smart Grid*.

21. Shrestha, D. L., & Solomatine, D. P. (2006). Machine learning approaches for estimation of prediction interval for the model output. Elsevier, 225-235.