ABSTRACT

The wind energy has emerged as one of the safest growing renewable energy to address the crisis witnessed in power generation. Wind speed is an important factor in wind power production and integration. However, the complex nature of wind speed limits the dependability and induces high fluctuation in power generation. The accurate prediction of wind speed energy with minimum accepted errors will increase to harness the energy content in a wind efficiently. In recent years, machine learning algorithms are used to analyze and predict data to make better decisions. Ensemble model is one of the supervised machine learning approaches to predict numerical data. In this project, the speed of wind is predicted through XGBoost package in a distributed programming environment of Apache Spark. XGBoost is a short form of Extreme Gradient Boosting tool of supervised machine learning, where the training data set $x_i$ is used to predict a target variable $y_i$. The results are compared with different iterations in order to minimize the uncertainties and to evaluate the efficiency and accuracy. The main findings were that Ensemble model was the most accurate method.

Keywords: Tree Ensemble, XGBoost, RMSE, Apache Spark.

1. INTRODUCTION

Wind energy is becoming a viable solution to global warming and fossil fuel problem. But the wind power industries face numerous technical issues one such issue is the prediction of wind speed. Errors in wind speed forecasts lead to errors in forecasting for both demands of electricity and economic benefits of power supply from wind farms. Therefore, small improvements in wind-speed forecasts constitute much larger improvements in wind power generation forecasts. There are two model classes for prediction tasks such as physical numerical weather simulation and machine learning algorithms. In the recent years, the machine learning algorithms are used widely. Since these models derive functional dependencies directly from the observations. They have also known as data-driven models. The project uses the ensemble model which is an accurate method used in machine learning wind prediction.

2. LITERATURE STUDY

(Pan Zhao et al, 2010), in this paper, the wind speed forecasting was carried out using the algorithm of support vector regression (SVR) techniques. The outcome is listed by comparing its performance with a back propagation neural network model through simulation. Both algorithms are applicable for prediction the wind speed time series in future; the prediction effect of support vector regression outperforms the back propagation neural network model as indicated by the mean square errors and mean absolute errors. Three different stages of the wind speed curve are analyzed, the results show that the proposed algorithm fit the original wind speed curve well at the whole process, but the back propagation neural network failed during the rise stage when the ascent rate suddenly become flatness of the original wind speed curve.
ors of 

The choice of model is based on the availability of weather adiance forecasting models using statistics and/or numerical weather forecasting. The system leverages upon multiple existing 

The steps involved in 

3. PROCESS FLOW

The steps involved in Short-term wind speed prediction is shown in Figure 1. Wind data set is read, pre-processed if any missing values. The XGBoost algorithm has been executed using Scala programming. Further data set is split into the training and testing set and used for validation and testing.
4. METHODOLOGY USED

Tree ensemble algorithms empower predictive models with high accuracy, stability and ease of interpretation. They are adaptable to solving any kind of problem. The decision trees are typically drawn upside down such that leaves are the bottom and roots are the tops. The tree formation follows both regression and classification types.

Like every other model, the tree ensemble also suffers from the bias means and variance of the split. Some of the commonly used ensemble methods are bagging and boosting. Bagging is a technique used to reduce the variance of predictions by combining the results of multiple classifiers modeled on different sub-samples of the same dataset.

(Leonel Viano) Boosting refers to a family of an algorithm which converts “weak learner” to be a “Strong learner” by using some kind of modifications. From a statistics point of view, this process was similar to creating a “good hypothesis” from a relatively “poor hypothesis”. A poor learner or a “weak hypothesis” is a model whose performance is slightly better than random chance. (Jerome 1999) Gradient boosting algorithm was developed for such high productive and capability. It builds an ensemble of trees one-by-one, then the predictions of individual trees are summed as given in equation 1 (with an assumption as ensemble has 3 trees).

$$D(x) = d_{tree\ 1}(x) + d_{tree\ 2}(x) + d_{tree\ 3}(x)$$  \hspace{1cm} (1)

The next tree (tree 4) in the ensemble should complement well the existing trees and minimize the training error of the ensemble as given in equation 2.

$$D(x) + d_{tree\ 4}(x) = f(x).$$  \hspace{1cm} (2)

To get a closer destination, the tree is trained to reconstruct the difference between the target function and the current prediction of an ensemble, which is called the residual/ errors as given in equation 3.

$$R(x) = f(x) - D(x).$$  \hspace{1cm} (3)

The losses currently supported by GBTs in MLlib are Log Loss (classification), Squared Error (Regression) and Absolute Error (Regression and outlier model). The steps to be followed in the Loss calculation

(i) Consider a whole set as the root node, Calculate Mean and Sum of Squared Error (SSE)

$$SSE = \sum_{i=1}^{n}(x_i - Mean)^2$$

(ii) Split the root node into leaf nodes, Calculate Mean and SSE for the leaf nodes

(iii) Find SSE Drop = SSE of Root Node – (Sum of SSE of Leaf nodes)

(iv) Select the Tree which has a high value of SSE Drop

Consider the following set of Wind Speed (m/s) is observed in various Wind power densities: 4.4, 5.1, 5.6, 6.0, 6.4, 6.8, 7.0, 9.4. The Tree can be formed in many ways. Consider the two trees represented in Figure 2 and 3.
Since SSE Drop is high in Tree 1, it is chosen as a fit model in the first level iteration. Further to solve over-fitting issues, the residual tree (error) will be formed and analyzed for prediction.

Developed by (Tianqui chen et al, 2016) XGBoost is an advanced implementation of Gradient boost algorithm. Some of the high performance capabilities of XGBoost are listed as

- known as ‘regularized boosting’ technique
- implements parallel processing and support faster computation
- allows users to define custom optimization objectives and evaluation criteria
- has an in-built routine to handle missing values
- Tree splitting up to the max-depth and remove splits beyond which there is no positive gain
- Allows the user to run a built-in cross-validation

The most important factor behind the success of the XGBoost is its scalability in all scenarios. The scalability of XGBoost is due to several important systems and algorithmic optimizations. These innovations include: a new tree learning algorithm for dealing with sparse data; theoretically reasonable weighted quantile sketch program can handle instance weights in approximate tree learning. Parallel and distributed computing make learning faster, enabling faster model exploration.

A precise short-term wind speed prediction is important for a safe and sustainable balancing of the electricity grid. This work focuses on short-range forecasting model using XGBoost regression in a distributed environment.

**System Design:**

The Wind Speed Prediction using machine learning algorithm depicts simple block diagram as shown in Figure 4. The Wind Speed Historical Data set is obtained, pre-processing is limited, since the data set is in the form of numerical information and there is no missing value. The data set is also divided into training and testing set.

![Wind Speed Prediction Block Diagram](image)
The prediction tool XGBoost4J performs the data processing and serve as complete data analytic pipeline on top of deployment frameworks such as Apache Spark. The combination of technologies used in the product environment is as follows:

- Apache Spark for the processing engine,
- Scala for the Programming Language
- XGBoost for the Supervised Learning Algorithm

To create a solution that can make accurate predictions, the predicted model to be designed as shown in Figure 5. The training set is a sample of data used to fit the model. The training data set is the actual data set, used to train the model. The validation data set is used to provide an unbiased evaluation of a model fit on the training data set and fine tune the model. The test data set is the sample of data used to provide an unbiased evaluation of a final model fit on the training data set. The validation set is released initially along with training set and the actual test set is released only when the model fit is derived. The split mainly depends on two things such as the total number of sample in data and actual model of training. Initially, the data set is split into training and test data set. The random choice of X \% of train data set makes remaining (100- x) \% as a validation test. However, when the model has more hyper parameters, the validation test is large, otherwise, it is small or nil. This is known as cross-validation and avoids over fitting during iterations.

**Figure 5. Iterative Flow of Data Processing in Predictive Model**

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. Scala is an acronym for “Scalable Language”. It is a general-purpose programming language designed for the programmers who want to write programs in a concise, elegant, and type-safe way. Scala can be regarded as a model for distributed computation of large amounts of data sets which are Java Virtual Machine (JVM) based and supports high scalability over other languages such as Java, Python, and R.

XGBoost is used for supervised learning problems, where one use the training data (with multiple features) $x_i$ to predict a target variable $y_i$. Like decision trees, GBTs handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture non-linearity and feature interactions. MLlib supports GBTs for binary classification and for regression, using both continuous and categorical features. MLlib implements GBTs using the existing decision tree implementation.

**The algorithm of XGBoost:**

Gradient boosting iteratively trains a sequence of decision trees. On each iteration, the algorithm uses the current ensemble to predict the label of each training instance and then compares the prediction with the true label. The dataset is re-labeled to put more emphasis on training instances with poor predictions. Thus, in the next iteration, the decision tree will help correct for previous mistakes. The specific mechanism for re-labeling instances is defined by a loss function (discussed below). With each iteration, GBTs further reduce this loss function on the training data.

GBTS train one tree at a time, which increases the likelihood of over-fitting and scalability. There are few parameters to be discussed before defining the algorithm:

- **numIterations**: This sets the number of trees in the ensemble. Each iteration produces one tree. Increasing this number makes the model more expressive, improving training data accuracy. However, test-time accuracy may suffer if this is too large.

- **learningRate**: This parameter should not need to be tuned. If the algorithm behavior seems unstable, decreasing this value may improve stability.

- **loss**: Different losses can give significantly different results, depending on the dataset.

- **algo**: The algorithm or task (classification vs. regression) is set using the tree [Strategy] parameter

However, it requires careful tuning, slow to train but fast to predict. Also, it cannot be extrapolated. The steps involved in the algorithm is listed as follows:

1. Input Dataset
2. Pre-process, do dataset cleaning and submit for training
   - The classifier looks for feature extraction and suggests best possible split
   - The optimal splitting is found through a greedy approach
3. Upon splitting, Percentile value is assigned for each split using an approximate algorithm.
4. Based on the percentile value, scores are assigned in each split.
5. Based on the scores, ranks are allotted to these splits.
6. During prediction, the scores are added to find the best fit and strengthen the weak learner.
7. The issues in real-time data such as missing values and empty space are addressed with the help of scores calculated in previous events with a separate path, while the normal decision tree faces misleading with such events.

5. SYSTEM ARCHITECTURE

XGBoost is a Machine learning library designed and optimized for tree boosting. The distributed XGBoost system runs magnitude faster than existing alternative of distributed machine learning algorithms and uses a few resources. Despite the benefits, there is a gap between the implementation of data processing frameworks and machine learning libraries, which prohibits the smooth connection. The common workflow utilizes the Spark/Flink to preprocess or clean data and creates additional overhead. To resolve the situation, a new-brewed XGBoost4J, for JVM platform is introduced as shown in Figure 6. It aims to provide the clean Java/Scala APIs and the integration with the most popular data processing system.

Figure 6. A new pipeline Architecture with XGBoost4J and Spark

Implementation

Upon completion successful completion of Apache Spark in a distributed environment, the wind speed prediction is executed using the proposed machine learning XGBoost algorithm through open source clustering framework. The study assumes that there are no missing values in the dataset for simplicity. The detailed implementation and experimental results are presented herewith.

The dataset fed into the programming is obtained through SRM Automated Weather Station (AWS). The sample dataset format is given in Figure 7. The dataset has other values such as wind direction, air temperature, etc., which helps in calculating wind power energy. They can also be used in an algorithm with multivariate analysis. XGBoost supports CSV and libsvm formats for training and inference.

Figure 7. Sample Dataset obtained from SRMAWS
XGBoost performs remarkably well in machine learning operations because it robustly handles a variety of data types, relationships and distributions, and a large number of parameters that can be tweaked and tuned for improved fits. The tree boosting parameters are set to provide optimum value. The list of parameters is eta, subsample colsample_bytree, colsample_bylevel(to prevent overfitting), max_depth of a tree and min_child_weight. The eval_metric sets as Root Mean Square Error (RMSE) for regression. The RMSE represents the sample standard deviation of the differences between predicted values and observed values. These individual differences are called residuals/loss when the calculations are performed over the data sample that was used for estimation and are called prediction errors when computed out-of-sample. The objective parameter specifies the learning task and the corresponding learning objective and set as reg: linear. The resource allocation can be plan ahead when the load requirement is known ahead of time and the process completion can be made within a given time frame using numworkers and numRound. There are no hard and fast rules which can guarantee best results in parameter tuning. One must understand, what the hyperparameters control and theoretical understanding of XGBoost algorithm. The most important parameters include depth of the trees, shrinkage (eta) and subsample. The setting of parameters in XGBoost algorithm is given in Figure 8.

Figure 8. Parameters setting along with RMSE calculation at Run time

The different terminal ids used in distributed environment for task scheduler is tracked and consolidated at the end when all the nodes complete their jobs. The warnings are also listed in case of non-issuance of allocated resources by nodes. This helps in consolidating available free resources for further processing. Figure 9 depicts the feature extraction along with prediction at various time intervals. The metrics evaluation concludes the performance efficiency of XGBoost.
6. CONCLUSION

The literature available for wind speed prediction modeling depicts that most of the models are used in wind power generation demand forecasting. There are several models ranging from linear regression, support vector model, artificial neural network and tree ensemble model etc., the efficiency of new tree ensemble model followed in XGBoost algorithm is to optimize the value of the objective function. The benefits of XGBoost constructs the boosted trees to intelligently obtain the feature scores, thereby indicating the importance of each feature to train the model.

The present study has taken the input from single machine, univariate model in automated weather station of SRM. However, the power output of a single system is not high enough to cause relevant effects on the electric grid, so the output of entire wind park to be studied and analyzed. The need of multi-machine, the multivariate model suggests including robust techniques in data pre-processing to deal missing values and inconsistency, data conversion and normalization, selection of proper input parameter, a combination of supervised machine learning algorithms with appropriate loss calculation and implementation. Our study focused on short-term wind prediction with XGBoost and with the objective function as regression: linear. The long-term prediction insists on make use of objective function as an artificial neural network for improving the results in wind forecasting. Thus the appropriateness of machine learning model in wind speed prediction is purely based on time series of wind prediction, variables used in the dataset, type of prediction model.

7. REFERENCES

Reitz Lecture,