# **IoT Temperature Analysis and Forecasting with Multiple Model comparison**

Ashutosh Kumar

Department of Engineering, University of San Diego

AAI-530: Data Analytics and the Internet of Things

Dr. Ying Lin

February 28, 2022

# **IoT Temperature Analysis and Forecasting and comparison of multiple models**

In this project, we have the temperature readings from IoT devices installed outside and inside of an anonymous room. Heat index (temperature + humidity) is one common data recorded on these IOT readers. The frequency of the upcoming data is very fast. The sensor reads hundreds to millions of data per second. There is a huge and versatile application of this data in real world. like: - Agriculture, weather forecasting, soil monitoring and treatment, enterprise maintenance, Data centers, and many more. Therefore, Wrangling, analyzing, and grasping insights from these data are equally important for multiple application sectors.

Because the device was in the testing phase, it was uninstalled or shut off several times during the entire reading period, which caused some outliers and missing values. For this project based on the details on Kaggle, assumption is the reading is taken from geographical region of India and measurements of temperature is in degree centigrade.

In this paper, I will examine the data trends using EDA and check effectiveness of applying machine learning techniques to the data. With use of EDA and machine learning algorithms we will try to get insights for below points:

* what was the max and min temperature?
* How outside temperature was related to inside temperature? any relation between the two?
* What was the variance of temperature for inside and outside room temperature?
* What is the trend in the data?
* Can you use algorithm to predict the next temp scenario?
* which was the hottest/coolest month?

A challenging aspect of this problem is we will need to use different techniques to identify the pattern of the temperature data from the given dataset. The given dataset has only the date of the data recorded with time. We have about a year's worth of data recorded from 11 January,2018 to 31st October,2018. We will have a look at the frequency of the sensor in a day i.e., how many times in a day the sensor measures the temperature, derive the season and weekdays from the given data as data cleanup activity. Thus, this seems to require more understanding than the usual. So, along with presenting our results obtained via different machine learning techniques, we also analyze the data using different techniques to gain a better under- standing of how important it is to clean the data collected from IoT devices before we use it for machine learning (ML) algorithm and getting meaningful insights from the data. In this paper we will compare experiment with multiple ML algorithms and use hyperparameter tuning to improve the results and improve the prediction capability of the models.

The source of the IoT data is Kaggle  [https://www.kaggle.com/atulanandjha/temperature-readings-iot-devices/version/1](https://www.kaggle.com/atulanandjha/temperature-readings-iot-devices/version/1!%5Bimage-7.png%5D(attachment:image-7.png)).

## Literature Review and Background of related area

The work of Korotcov, A., Tkachenko, V., Russo, D. P., & Ekins, S. (2017) is extensive study on the impact of multiple Machine learning (ML) algorithm where authors have compared multiple classic algorithms with deep learning. They have also used popular models like linear logistic regression, AdaBoost decision tree, random forest, and support vector machine. Open source Scikit-learn (http://scikit-learn.org/stable/, CPU for training and prediction) ML python library was used for building, tuning, and validating all CML models included in this pipeline. Also, the work of Pang, B., Lee, L., and Vaithyanathan, S. (2002) Validated the model accuracy for NLP using models like Naïve Bayes, Maximum Entropy, Support Vector Machine (SVM), there is already work done in past that details about Model comparison and how various hyperparameters impact the model prediction before and after tuning the hyperparameters.

## Discussion

Internet of things (IoT), as an enabler to share data through connectivity. The biggest advantage of IoT is its ability to monitor in real time. IoT also enables human-to-human, machine-to-human, and machine-to-machine communication without external intervention. We are always curious to know how climate is changing day by day, year to year. One way to understand this is by analysis and understanding of heat fluctuations and heat index. Below table show how heat index impacts human body.

|  |  |  |
| --- | --- | --- |
| Temp in Celsius | Temp in Fahrenheit | Heat Index (human discomfort index) |
| 27 – 32 °C | 80-90 °F | Caution fatigue is possible with prolonged exposure and activity. Continuing activity could result heat cramps |
| 32 – 41 °C | 90-105 °F | Extreme Caution Heat Cramps and Heat exhaustion are possible. Continuing activity could result heat stroke. |
| 41 – 54 °C | 105-130 °F | Danger Heat Cramps and Heat Exhaustion are likely; Heat stroke is probable with continued activity. |
| Over 54 °C | Over 130 °F | Extreme Danger Heat Stroke is Eminent. |

The data collected from various IoT sensor can be used with the help of machine learning models to predict the temperature and specific precautions can be taken ahead of time based on predicted heat index.

*Machine Learning methods used:*

K-Mean Clustering: - K-Means is one of the most common clustering techniques. It is a centroid-based clustering algorithm where the objective is to find K clusters / groups. The working of K-means clustering can be summarized as follows:

Step 1: Initialize the K random centroids or K points

Step 2: For each data point, calculate the Euclidean distance of it from randomly chosen K centroids and assign each point to a minimum distance cluster.

Step 3: Update the centroid by using newly assigned data points to the cluster by calculating the average of data points.

Step 4: Repeat the above process for a given no. of iterations or until the centroid allocation no longer changes

Large K produces smaller clusters and small K produces larger clusters.

Below is the output of K-Mean clustering with different values of K.

Chart, line chart

Description automatically generated

Decision Tree: - The general idea is that we will segment the space into several simple regions. The segmentation can be illustrated as a tree, The end nodes can have a category (classification) or a continuous number (regression). Algorithms for constructing decision trees usually work top- down, by choosing a variable at each step that best splits the set of items. Different algorithms use different metrics for measuring “best”. These metrics measure how similar a region, or a node is. They are said to measure the impurity of a region. Larger these impurity metrics the larger the “dissimilarity” of a nodes/regions data. Examples: Gini impurity, Entropy, Variance.

Random Forest: - A random forest is a meta estimator that fits several classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Random forest uses random sampling with replacement technique. For each subset build a decision tree. However, only use m randomly pick independent variables for each node’s branching possibilities. While predicting Random forest use each tree to make individual prediction and later it combine predictions using voting technique using means for regression or modes for classification.

AdaBoost: - Initially it assigns equal sample weights for each sample. We bootstrap the samples as per the weights assigned and build a weak learner (in our case we are using Decision Tree as base weak model on that sample Once the weak learner is built, AdaBoost measures the importance of a weak learner based on the error made by that weak learner. New sample weight according to the correct and incorrect predictions are assigned to each sample and a new bootstrapped dataset is created with the odds of each sample being chosen based on their new sample weights. Process is repeated n number of times. The final prediction is a weighted majority vote/average of all the weak learners

Diagram, calendar

Description automatically generated

Source: https: //www.researchgate.net /figure/ Training-of-an-AdaBoost-classifier-The-first-classifier-trains-on-unweighted- data-then\_fig3\_306054843

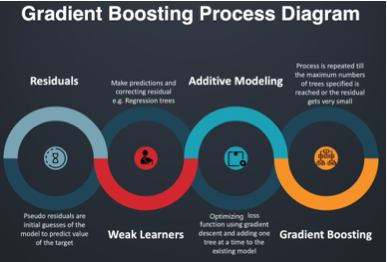
Some important hyperparameters of AdaBoost (SKlearn)

base\_estimator: The base estimator from which the boosted ensemble is built. By default the base estimator is a decision tree with max\_depth=1

n\_estimators: The maximum number of estimators at which boosting is terminated. Default value is 50.

learning\_rate: Learning rate shrinks the contribution of each classifier by learning\_rate. There is a trade-off between learning\_rate and n\_estimators.

Gradient Boost: - A weak learner is built on a subset of the original data. Errors are calculated after predictions by the weak learner by comparing the predictions and actual values. The main difference between AdaBoost and Gradient Boosting is that AdaBoost changes the weights of sample after prediction by each weak learner while Gradient Boosting fits the next weak learner to the residuals of the previous weak learner. The residuals become the target variable for the new weak learner. The main aim is to minimize the residuals i.e. the error made by the previous weak learner or regularizing the loss function The same process is repeated until there is no reduction in the residuals, or the number of estimators is reached.



Source: https://dzone.com/articles/xgboost-a-deep-dive-into-boosting

Some important hyperparameters of GradientBoost Model (SKlearn)

Init: An estimator object that is used to compute the initial predictions. If ‘zero’, the initial raw predictions are set to zero. By default, a DummyEstimator predicting the classes priors is used.

Learning\_rate: Learning rate shrinks the contribution of each tree by learning\_rate. There is a trade-off between learning\_rate and n\_estimators.default=0.1

n\_estimators:The number of boosting stages to perform. Gradient boosting is fairly robust to over-fitting so a large number usually results in better performance , default=100

subsample:The fraction of samples to be used for fitting the individual base learner,default=1.0

max\_features: The number of features to consider when looking for the best split, default=None

XgBoost: - XGBoost builds upon the idea of Gradient Boosting with some modifications to have high performance and faster implementation than gradient boosting and adaboost algorithm. It has a variety of in-built features like handling missing values and different hyperparameters to reduce overfitting and improve overall performance.

Steps involved in XGBoost-

1. XGBoost sets the initial prediction for all observations as the mean of the target values for regression and 0.5 for binary classification problems.
2. Then it calculates the residual for each observation.
3. It builds a tree to predict the residuals and
4. The main difference in terms of building a tree between xgboost and gradient boosting is how they decide the best split at each level. While gradient boosting traditional slitting criterion like gini or entropy, xgboost uses gain calculated from similarity score to find the best split which is directly
5. After a tree a built, it calculates the output value for each leaf node.
6. The same process from step 2-4 is repeated until there is no reduction in the residuals, or the number of estimators is reached.

Some important hyperparameters of AdaBoost (SKlearn)

Learning\_rate / eta: Gradient boosted decision trees are quick to learn and overfit training data. One effective way to slow down learning in the model is to use a learning rate, also called shrinkage.

gamma: A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split. Higher the gamma value, lesser the chances of overfitting

scale\_pos\_weight: Control the balance of positive and negative weights. It can have any positive value as input. It helps in imbalanced classification problems.

*Evaluation of Models and Results:*

After performing various univariate and bivariate analysis and EDA data was analyzed from multiple perspective, there was no missing value found in the data but there were some duplicated records identified that was addressed using python pre-processing methods.

After the EDA and data cleansing various machine learning algorithms used starting with K-mean clustering, Decision tree, Random Forest, Gradient Boosting, AdaBoost and XGBoost.

At the beginning the model’s performance is evaluated based on R2 Score and below is the summary.

|  |  |
| --- | --- |
| Model | R2 Score |
| Decission Tree | .98 |
| Random Forest | .99 |
| Gradient Boost | .95 |
| AdaBoost | .80 |
| XG Boost | .98 |

All the models performing well, but Random Forest model seems overfitting. AdaBoost has the lowest R2 score.

Two best and one lowest performing model selected for hyperparameter tuning and results were evaluated.

Hyperparameters are the parameters that govern the entire training process Their value are set before the learning process begins. They have a significant effect on model’s performance The process of finding optimal hyperparameters for a model is known as hyperparameter tuning. Choosing optimal hyperparameters can lead to improvements in overall model’s performance and can help in reducing both overfitting and underfitting. I have used RandomizedSearchCV as best hyperparameter search technique. Random search is very similar to grid search, the difference is that in random search we define ‘n\_iter’ - not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by n\_iter. the set of hyperparameters is not searched sequentially. We can pass a range here instead of just numbers. The reason of using Random search is because it takes lesser time than Grid search method.

Below is the model performance summary after hyperparameter tuning using RandomizedSearch method.

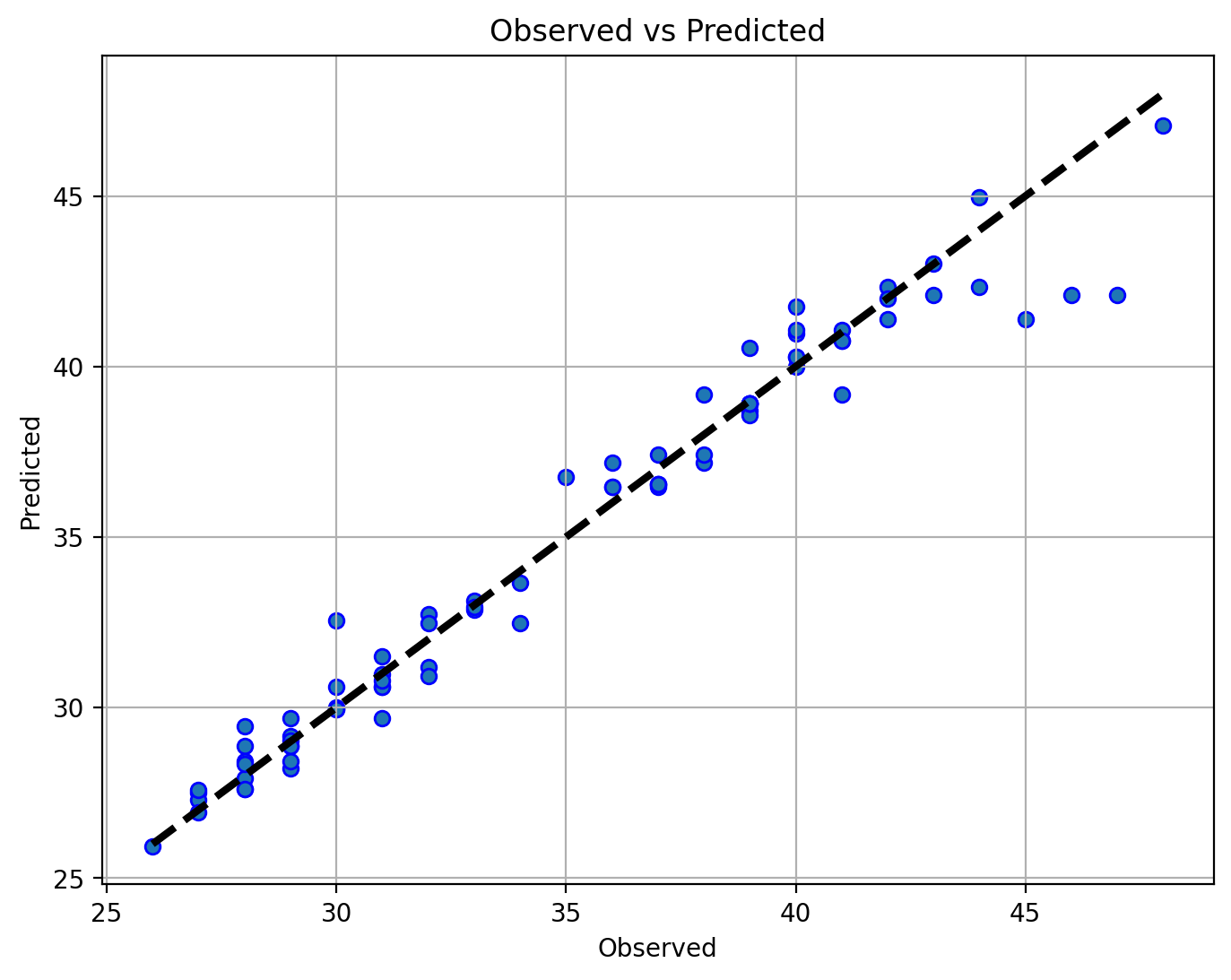
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Train R2 | Test R2 | Train RMSE | Test RMSE |
| Gradient Boost | 0.99 | 0.99 | 0.19 | 0.27 |
| XGBoost | 0.99 | 0.99 | 0.21 | 0.28 |
| AdaBoost | 0.98 | 0.97 | 0.64 | 0.67 |

## Conclusion

With analysis and performance evaluation of multiple machine learning models we have seen that performance to lowest performing model AdaBost is also improved significantly. This is observed that Gradient Boost model is the best performing model and can be utilized to predict the temperature with high accuracy and measured against the heat index. The model can be very useful to provide early temperature fluctuation and warning can be issues in the applicable geographies so that people can take precautions. Additionally same model can be deployed using a streamlit app and accessed through any desktop browser or mobile device.

Chart

Description automatically generated



## References

1. Korotcov, A., Tkachenko, V., Russo, D. P., & Ekins, S. (2017). Comparison of Deep Learning With Multiple Machine Learning Methods and Metrics Using Diverse Drug Discovery Data Sets. MOLECULAR PHARMACEUTICS, 14(12), 4462–4475. https://doi-org.sandiego.idm.oclc.org/10.1021/acs.molpharmaceut.7b00578
2. Marsh, Brendan. (2016). Multivariate Analysis of the Vector Boson Fusion Higgs Boson.
3. <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html>
4. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>
5. <https://docs.streamlit.io/library/get-started>