Earlier known as
B. V. B. College of Engineering & Technology

School of Computer Science and Engineering

Data Mining and Analysis (18ECSC301)

Project Report

On

Cold Start Energy Forecasting

Submitted by

APOORVA HEGDE	**01FE16BCS040**
GANESH JADHAV	**01FE16BCS065**
GIRISH ILLANAD	**01FE16BCS071**
H HEMANTH KUMAR	**01FE16BCS073**

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

HUBLI – 580 031 (India)

Academic year 2018-19

INTRODUCTION

Building energy forecasting has gained momentum with the increase of building energy efficiency research and solution development. Indeed, forecasting the global energy consumption of a building can play a pivotal role in the operations of the building. It provides an initial check for facility managers and building automation systems to mark any discrepancy between expected and actual energy use. Accurate energy consumption forecasts are also used by facility managers, utility companies and building commissioning projects to implement energy-saving policies and optimize the operations of chillers, boilers and energy storage systems.

Usually, forecasting algorithms use historical information to compute their forecast. Most of the time, the bigger the historic dataset, the more accurate the forecast. This requirement presents a big challenge: We must make accurate predictions for new buildings, which don't have a long consumption history (cold start). This data consists of varying amounts of historical consumption and temperature data that are given for each series, ranging from 1 day to 2 weeks and basic building information such as surface area, base temperature, and on/off days are given for each series id.

PROBLEM STATEMENT

The goal of this challenge is to build an algorithm which provides an accurate forecast of energy consumption from the very start of a building's instrumentation, given a small amount of data.

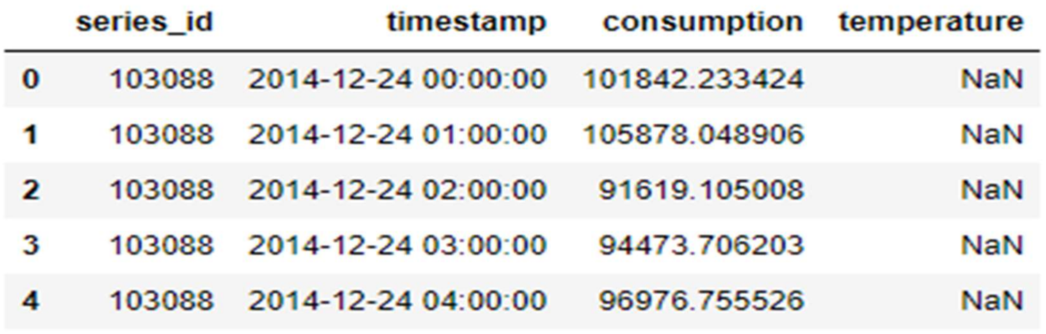
Related Work

Another competition from the same host site had challenges like ours- Power Laws: Forecasting Energy Consumption. The problem statement was as to provide an algorithm that can (i) either make a good forecast for all or some of the buildings or (ii) bring the conclusion that other data would be necessary to make relevant forecasts. The third winner of the same Will Koehrsen had applied various techniques, of which one was converting time variables to cyclic forms, i.e. splitting a variable into corresponding sine and cosine function so that the model that we build on, realizes that 23rd hour and 0th hour is rather close and not actually far. He had a final private score of .00203. This technique of converting the time variables was useful to us since our challenge had time series as well.

Dataset

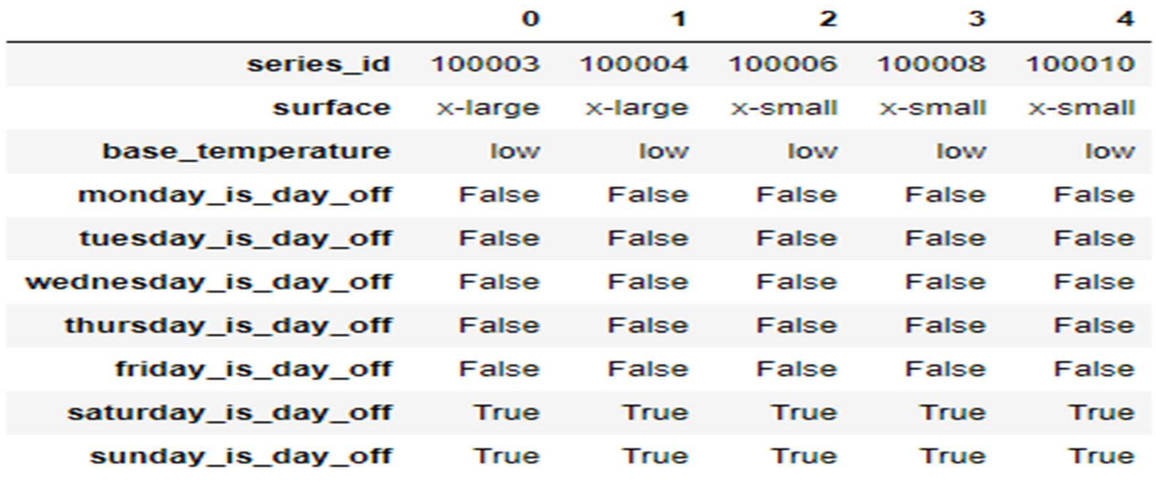
The entire data consisted of training, testing and Meta data. The train and test data had unique timestamp and series_id (combination of these attributes was unique) and other attributes were temperature, consumption and index – ‘unnamed’. Consumption was the label. The Meta data had other important information about the building i.e. base temperature, surface area of the building, on/off days and their corresponding series id.

Reviewing the dataset



	series_id	timestamp	consumption	temperature
0	103088	2014-12-24 00:00:00	101842.233424	NaN
1	103088	2014-12-24 01:00:00	105878.048906	NaN
2	103088	2014-12-24 02:00:00	91619.105008	NaN
3	103088	2014-12-24 03:00:00	94473.706203	NaN
4	103088	2014-12-24 04:00:00	96976.755526	NaN

Figure 1: Training data



	0	1	2	3	4
series_id	100003	100004	100006	100008	100010
surface	x-large	x-large	x-small	x-small	x-small
base_temperature	low	low	low	low	low
monday_is_day_off	False	False	False	False	False
tuesday_is_day_off	False	False	False	False	False
wednesday_is_day_off	False	False	False	False	False
thursday_is_day_off	False	False	False	False	False
friday_is_day_off	False	False	False	False	False
saturday_is_day_off	True	True	True	True	True
sunday_is_day_off	True	True	True	True	True

Figure 2: Meta data

METHODOLOGY

Our approach treats the problem as a standard supervised regression task. Given a set of features – the time and other building information – we want to build a model that can predict the continuous target, energy consumption. The model is trained on the past historical energy consumption using the features and the target and then can be used to make predictions for future dates where only the features are known. The KDD is an iterative process. We used various methods at first to fill in the missing values which itself was the big task. Of which, the first method was to fill in with interpolate method for all. This gave us bad results with negative accuracies. The second iteration included filling it with median after grouping them on series id and month; which left us with some missing temperature values again. So, this was followed by machine learning to fill in the rest of the missing data. This again was giving bad results. Negative scores from the two approaches were clearly indicating that our model was under fitting and we needed to change our technique of preprocessing. The fresh start from preprocess was done for our final model.

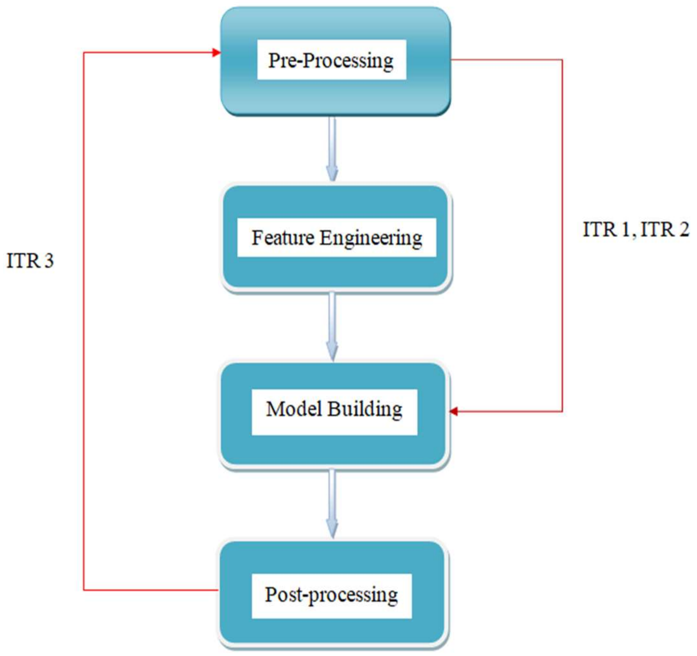


Figure 3: KDD Process

Data Preprocessing

We first merged the train and test data with the meta data. The six attributes in the Meta data for holiday was mapped to one attribute while merging (like just the reverse of one hot encoding). This attribute was crucial because the energy use differs significantly between working and non-working days. We then split the timestamp to time, day of week, day of month, and day of year for more specificity. The time variables were converted to cyclical features. This technique was adopted, as mentioned in the related works section.

Post merging and feature engineering and other conversions, the number of attributes increased to 26 from 5. The box plots didn't indicate outliers for any attributes except for consumption, we anyway didn't discard them considering those buildings maybe some industries/factories with high consumption and maybe useful for model building.

With the main aspect of the cold start challenge, i.e. most of the data was missing since the buildings are said to be new. The 44% of the temperature had missing values with some series id's no temperature at all for over the whole-time duration for which their consumption was observed. With such a challenge, we used two approaches, one being filling the missing values and the other to ignore the temperature attribute completely.

With the first approach, that is filling the missing temperature values, we first grouped the series id, month and year with the sense that for any building, the temperature variance will not be more over a span of one month and hence filled it with mean. The main reason for using mean was: for ids with no temperature at all for some span (maybe a day or a week out of total 2 weeks) will have filled with the average temperature. This made more sense. Post-filling with mean, we were still left with 34% of missing temperature. We then grouped the series Id and year to further fill the missing values, which resulted in 1% decrease of missing percentage.

Now with left 33% missing data, we further used random forest regressor to fill in the missing temperature (keeping the consumption attribute aside and passing the rest). Since the data was non-linear as shown by the graphs, using an ensemble model (like random forest regressor) rather than linear regression was appropriate.

The preprocessed data filled with random forest regressor and one hot encoding on the two categorical attributes marked the end of our preprocessing of the first approach.

In the second approach we used the timestamp and the series id attributes and ignored all the other attributes, making it a forecasting problem. This approach didn't need preprocessing except for passing the raw data to the LSTM model.

Model Building

The models used are:

Linear Regression

Multiple linear regression attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data. Every value of the independent variable *x* is associated with a value of the dependent variable *y*. The population regression line for *p* explanatory variables $x_1, x_2, \ldots, x_p$ is defined to be $\mu_y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$. This line describes how the mean response μ_y changes with the explanatory variables. The observed values for *y* vary about their means μ_y and are assumed to have the same standard deviation σ. The fitted values $b_0, b_1, \ldots, b_p$ estimate the parameters $\beta_0, \beta_1, \ldots, \beta_p$ of the population regression line.

Since the observed values for *y* vary about their means μ_y, the multiple regression model includes a term for this variation. In words, the model is expressed as DATA = FIT + RESIDUAL, where the "FIT" term represents the expression $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_p x_p$. The "RESIDUAL" term represents the deviations of the observed values *y* from their means μ_y, which are normally distributed with mean 0 and variance σ. The notation for the model deviations is ε.

Formally, the model for multiple linear regression, given *n* observations, is $y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots \beta_p x_{ip} + \varepsilon_i$ for $i = 1,2, \ldots n$.

In the least-squares model, the best-fitting line for the observed data is calculated by minimizing the sum of the squares of the vertical deviations from each data point to the line (if a point lies on the fitted line exactly, then its vertical deviation is 0). Because the deviations are first squared, then summed, there are no cancellations between positive and negative values. The least-squares estimates $b_0, b_1, \ldots b_p$ are usually computed by statistical software.

Bagging

A Bagging regressor is an ensemble meta-estimator that fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

Random Forest Regressor

The following are the basic steps involved in performing the random forest algorithm:

1. Pick N random records from the dataset.
2. Build a decision tree based on these N records.
3. Choose the number of trees you want in your algorithm and repeat steps 1 and 2.
4. In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

The random forest algorithm is not biased, since, there are multiple trees and each tree is trained on a subset of data. Basically, the random forest algorithm relies on the power of "the crowd"; therefore the overall baseness of the algorithm is reduced.
This algorithm is very stable. Even if a new data point is introduced in the dataset the overall algorithm is not affected much since new data may impact one tree, but it is very hard for it to impact all the trees.
The random forest algorithm works well when you have both categorical and numerical features.
The random forest algorithm also works well when data has missing values or it has not been scaled well (although we have performed feature scaling in this article just for the purpose of demonstration).

Extra Trees Regressor

The Extra-Tree method (standing for **ext**remely **ra**ndomized **tree**s) was proposed with the main objective of further randomizing tree building in the context of numerical input features, where the choice of the optimal cut-point is responsible for a large proportion of the variance of the induced tree.

With respect to random forests, the method drops the idea of using bootstrap copies of the learning sample, and instead of trying to find an optimal cut-point for each one of the K randomly chosen features at each node; it selects a cut-point at random.
This idea is rather productive in the context of many problems characterized by a large number of numerical features varying more or less continuously: it leads often to increased accuracy thanks to its smoothing and at the same time significantly reduces computational burdens linked to the determination of optimal cut-points in standard trees and in random forests.
From a statistical point of view, dropping the bootstrapping idea leads to an advantage in terms of bias, whereas the cut-point randomization has often an excellent variance reduction effect. This method has yielded state-of-the-art results in several high-dimensional complex problems.
From a functional point of view, the Extra-Tree method produces piece-wise multilinear approximations, rather than the piece-wise constant ones of random forests.

LSTM

Long Short Term Memory (LSTM) units are units of a recurrent neural network (RNN) well-suited to classifying, processing and making predictions based on time series data. They networks can learn patterns over long sequences and do not necessarily rely on a pre-specified window of lagged observation as input. Further, their internal states can be built up to include information about cold start data before making any number of forecast predictions we ask for. This means that a single LSTM model can handle all prediction horizons in the data.

TUNING HYPERPARAMETERS

The ensemble models have many tuning parameters, out of which the effect of max depth was the most. With increasing max_depth our model accuracy increased. We have used bagging, extra trees and random forest regressors. The important parameters that we played with are:

PARAMETERS (ENSEMBLE METHODS):
n_estimators: integer, optional (default=10)
    The number of trees in the forest.
max_depth : integer or None, optional (default=None)
    The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

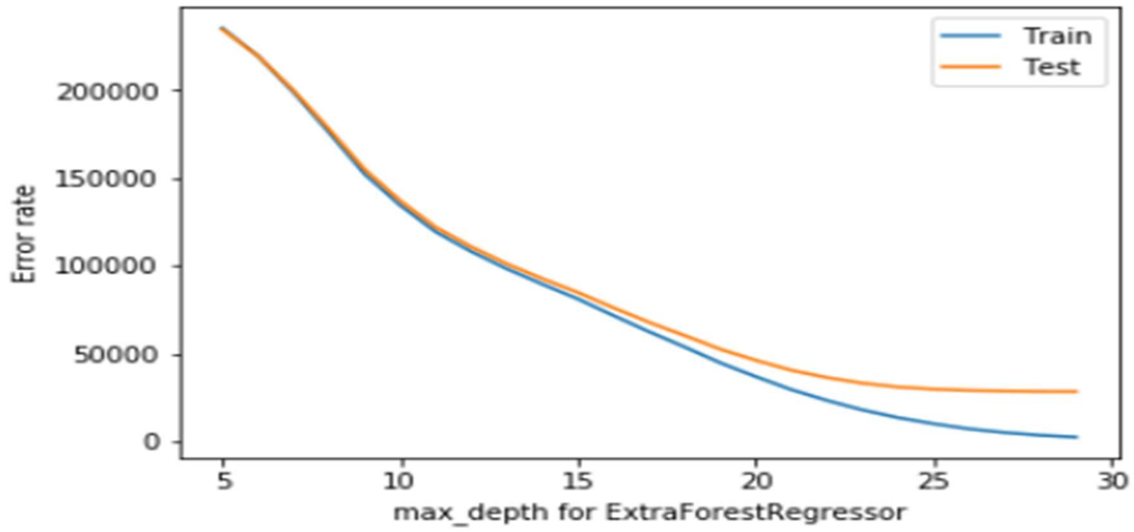


Figure 4: Error rate vs max_depth graph

The plots show that the error of the model decreases with increasing depth. At depth=30 we get min error for both training and testing data.

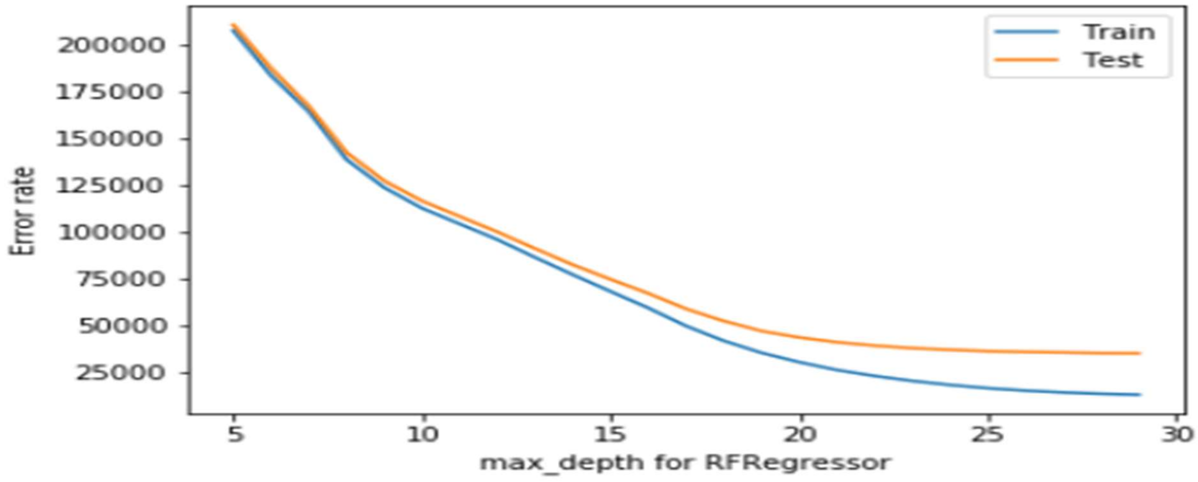


Figure 5: Error rate vs max_depth graph

The plots show that the error of the model decreases with increasing depth. At depth=30 we get min error for both training and testing data.

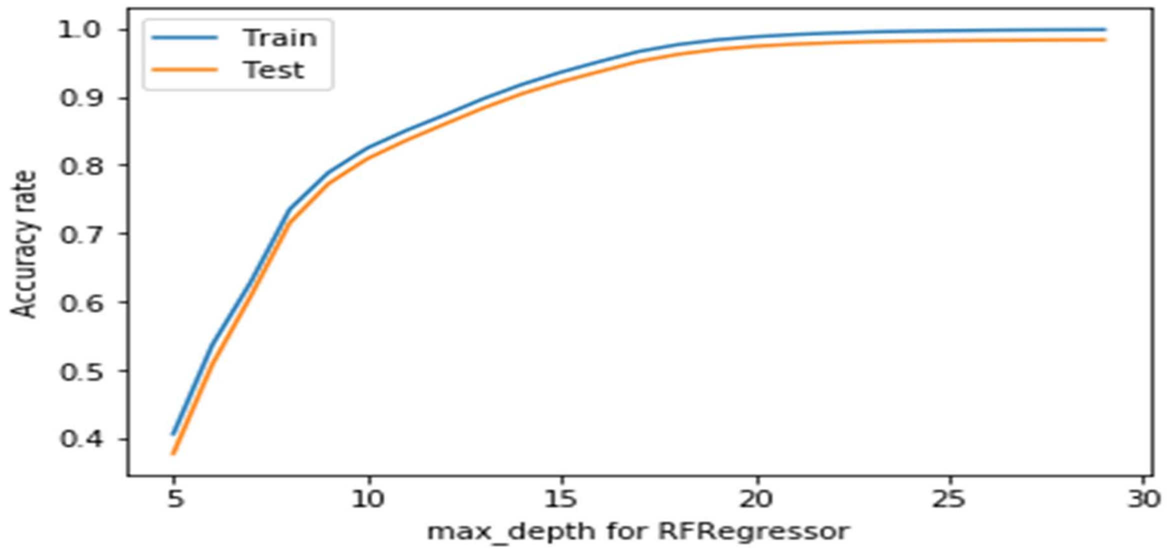


Figure 6: Accuracy vs max_depth graph

The plots show that the score of the model increases with increasing depth. At depth=30 we get max score for both training and testing data.

MODEL EVALUATION

The models that gave best results were then checked for overfit and underfit with tuned parameters.

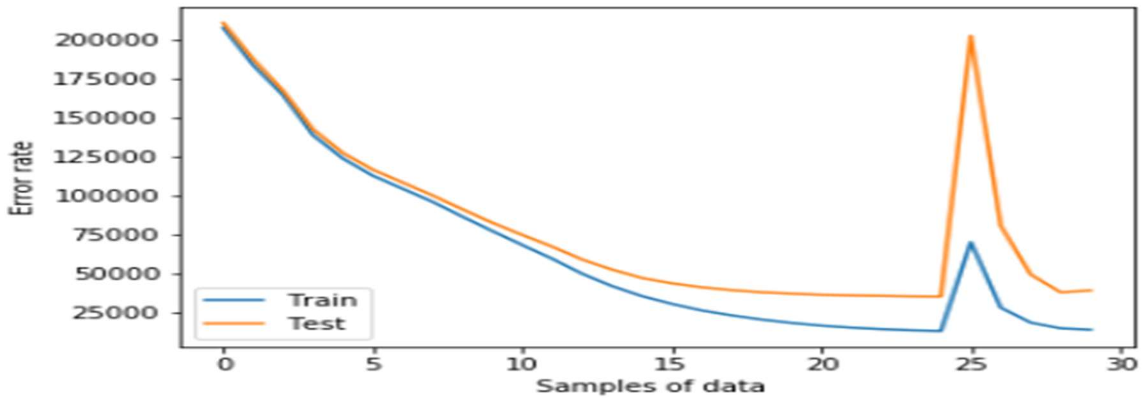
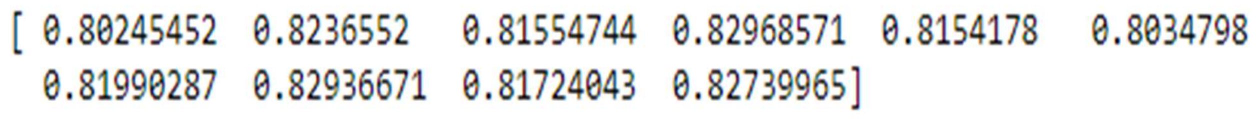


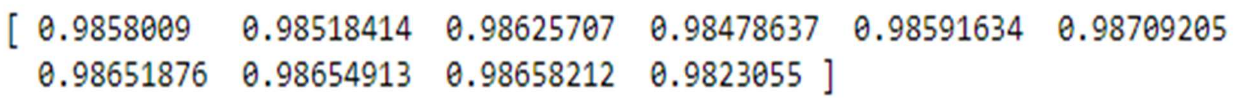
Figure 7: Error rate vs data sample graph

The Kfold Cross Validation (10 folds) results for the random forest regressor: (depth=10)



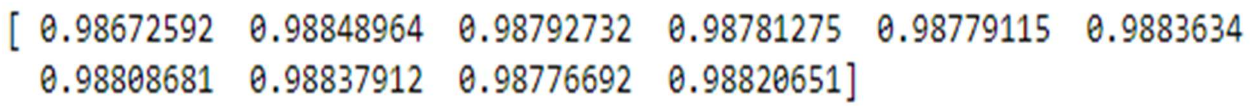
```
[ 0.80245452  0.8236552   0.81554744  0.82968571  0.8154178   0.8034798
  0.81990287  0.82936671  0.81724043  0.82739965]
```

The Kfold Cross Validation (10 folds) results for the random forest regressor: (depth=30)



```
[ 0.9858009   0.98518414  0.98625707  0.98478637  0.98591634  0.98709205
  0.98651876  0.98654913  0.98658212  0.9823055 ]
```

The Kfold Cross Validation (10 folds) results for the extra tree regressor: (depth=30)



```
[ 0.98672592  0.98848964  0.98792732  0.98781275  0.98779115  0.9883634
  0.98808681  0.98837912  0.98776692  0.98820651]
```

RESULTS AND DISCUSSIONS

Linear models like linear regression, ridge and lasso did not perform well and it was no surprise since it was already clear from the graphs that the data was non-linear. However, ensemble methods boosted the model performance and the error was optimized as well. The ensemble models that performed well were random forest regressor, extra trees regressor, bagging regressor and KNeighbor regressor. However, the random forest regressor and the extra trees regressor gave the best results comparatively. The main reason that ensemble models worked well for our data was that the Extra-Trees algorithm builds an ensemble of unpruned decision or regression trees according to the classical top-down procedure. Its two main differences with other tree-based ensemble methods are that it splits nodes by choosing cut-points fully at random and that it uses the whole learning sample (rather than a bootstrap replica) to grow the trees.

The extra randomization makes the decision boundaries smoother. The main reason they sometimes outperform RF is because of the extra randomness in the ensemble, so the base learners make mistakes which are less correlated to each other.

The model fit perfectly for both train and test data and the kfold cross validation results indicated the same. The bias variance curve also supported the same; representing low bias and low variance.

CONCLUSIONS

The meta data had attributes that played important role.

The only attribute with missing values was the temperature; it was first filled via mean and later with machine learning technique.

The ensemble methods fit best than the linear models because of the non linear property of the data.

The random forest regressor and the extra trees regressor gave accuracy of 98.2% and 98.9% respectively and were better among all other ensemble techniques. The learning curves fit perfectly (neither underfit nor overfit) for both training and testing for both models mentioned above. The extra trees regressor would be used be used for prediction because of the accuracy and the fit over the validation curve.

REFERENCES

1. https://www.drivendata.org/competitions/

2. https://scikit-learn.org/stable/modules/ensemble.html