Epochs: 10

RMSLE Train = 1.41

RMSLE Test = 1.41

Kaggle score = 2.327

Epochs: 23

RMSLE Train = 1.02

RMSLE Test = 1.02

Kaggle score = 2.327??

**Methodology**

**Lightbgm:** Lightbgm is a gradient boosting framework, using decision trees. By using gradient based one side sampling and exclusive feature bundling make the lightbgm a very lightweight and fast algorithm. It is particularly effective with large datasets, however it is prone to overfitting small datasets. Whilst a decision tree splits the tree tree-wise, the lightbgm algorithm splits the tree leaf wise.

**RNN:** A recurring neural network is a form of machine learning that returns on itself to increase accuracy. For a typical neural network, firstly, an example from a dataset is loaded (in this case, the training data supplied). The network takes that example, and applies mathematical formulae to it using random variables, yielding a predicted result. Using the validation data, this prediction can be compared, and the difference between them will give an error. Returning the error back through the same path will adjust the variables, and this is all repeated until the variables are defined with minimal error. The recurrent neural network takes this a step further, by instead of taking in one example at a time and producing one result, it takes multiple neural networks which feed information to each other. This allows for a low time complexity and makes it suitable for dealing with large datasets.

**Decision Tree:** The general idea of a decision tree is that it uses rules to make decisions. You begin with the root node, which splits of to two different regions. These regions also split into two different regions. The decision tree uses the rules supplied to decide which region to go to at a node, and this process is continued until all the rules are applied or until there are no data points left. The decision tree time complexity is of form O (n log(n) \* d), where d is dimensionality in the data. A random forest uses multiple decision trees with an element of randomness for more accuracy, however due to the multiple decision trees the computational memory cost is significantly higher than that of a single decision tree.

**KNN**: The idea of KNN is that it uses ‘feature similarity’ to predict the values of new data points. Effectively, new points are assigned based on its resemblance in comparison to the training dataset. The value of k can be adjusted, such that more of the data is included in a single prediction. This algorithm can be used for either classification or regression, however for large datasets it can be extremely costly in terms of memory in comparison to other models with a time complexity of O (n) for testing.

**Experimental Setting:**

As part of the pre-processing, one of the main factors was memory usage due to the large nature of the data. By converting the data to feather files and applying a function to reduce the memory usage (mainly by converting int64s to int8s, or float64s to float32s) this allowed us to run the programme more efficiently with very little loss of data. The data was also merged with the weather data and the building data to allow for more features to be selected, and the timestamp was broken down into hours, days, day of the week, day of the year, the month, and the year, which allows us to get the exact date in integer values. A log transformation was also applied to the meter reading and square feet, which was done to account for the large range of values and make the training of the model more effective. Furthermore, months were added along with an ‘isDayTime’ Boolean in form 1 or 0, 1 being it is between the hours of 06:00 and 18:00. The feature selection allowed us to remove various features from the data to acquire a better result from the data.

**Results (Kind of)**

**RNN**

A recurring neural network model was used to predict the meter readings data, the reasoning being that an RNN has a low time complexity. Given the extremely large nature of the data set (~41 million rows of data), this is an extremely useful property. With epochs = 10, the RMSLE was 1.41, and the prediction returned a Kaggle score of 2.33. With epochs = 23, the RMSLE was 1.02, however the prediction still returned a Kaggle score of 2.33. With epochs = 50, the RMSLE was 1.05, and the prediction returned a Kaggle score of 2.54. Whilst this model is not as accurate as other models, such as lightbgm or random forest, the low memory usage and quick running time makes it suitable for usage on extremely large datasets, where the accuracy can be within a tolerable range.