# Algorithm Description

## 1. Validation

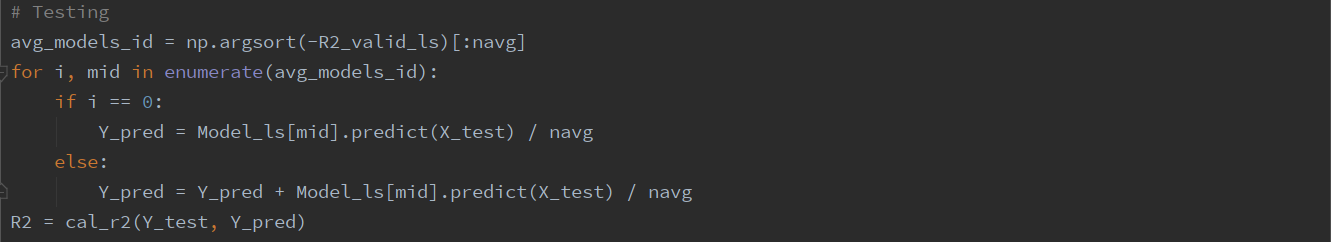
To determine the proper hyper-parameters for each machine learning method, I use the validation sample to select the top models with the highest validation, and then average their outputs. In the code, I first generate the list of all combinations of several key hyper-parameters.



For each hyper-parameters’ combination, I train a model using and , calculate its validation, and save the model in for testing.



After finishing the validation, I obtain the IDs of well performed models by sorting in a descending order and selecting the first corresponding indexes. Finally, the predicted value is the average of these models’ outputs, and the testingcan also be calculated.

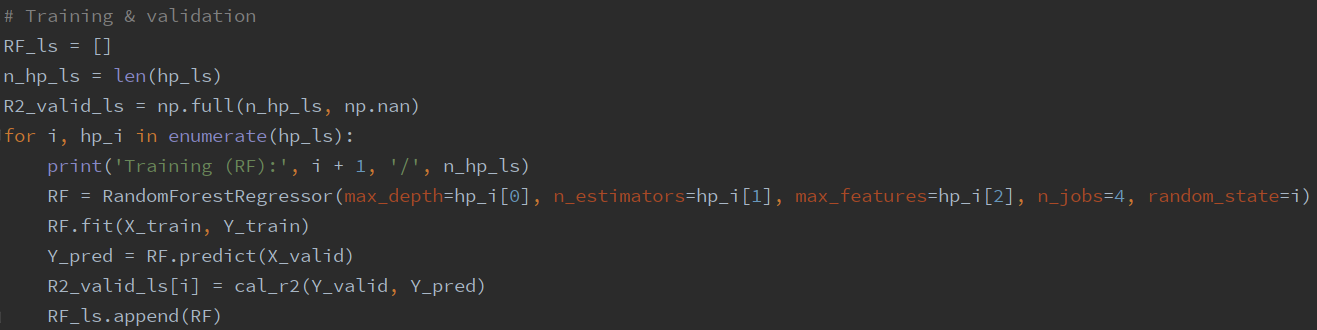


## 2. Methods

### 2.1. Random Forest

A random forest is an ensemble method that combines forecasts from many different trees. In the code, I mainly tune the following hyper-parameters for random forest.

|  |  |
| --- | --- |
| Name | Description |
| max\_depth | The maximum depth of the tree. Though a deeper tree can fit a more complex functional form, it also has a higher tendency to overfit the data. |
| n\_estimators | The number of trees in the forest. |
| max\_features | The number of features to consider when looking for the best split. This parameter is to lower the similarities between trees. |



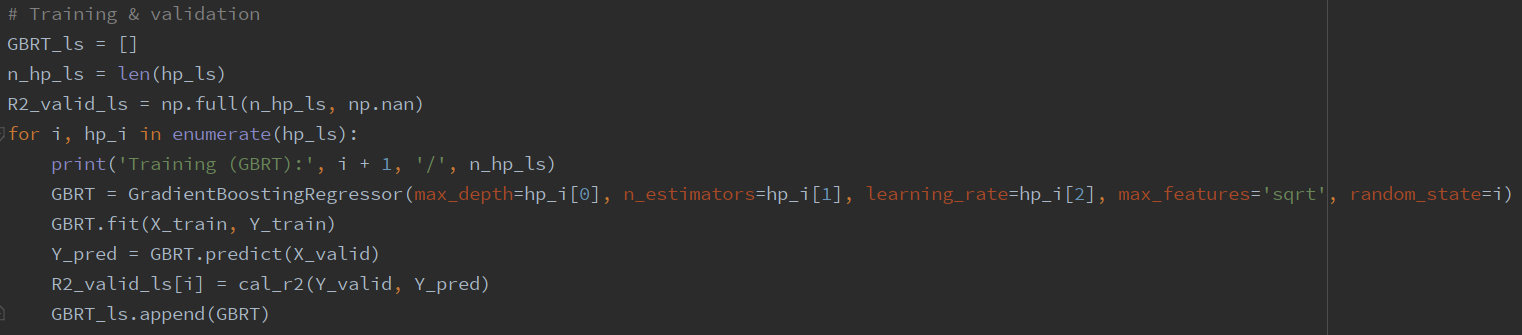
For more details, please see

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#sklearn.ensemble.RandomForestRegressor>

### 2.2. Gradient Boosting Regression Tree (GBRT)

GBRT recursively combines forecasts from many trees. The descriptions of the hyper-parameters are as follows.

|  |  |
| --- | --- |
| Name | Description |
| max\_depth | The maximum depth of the tree. This parameter is similar to the one in random forest. |
| n\_estimators | The number of trees in the forest. In GBRT, these trees are combined in series, while in random forest they are in parallel. |
| learning\_rate | Learning rate shrinks the contribution of each tree. |
| max\_features | The number of features to consider when looking for the best split. “sqrt” means . |



For more details, please see

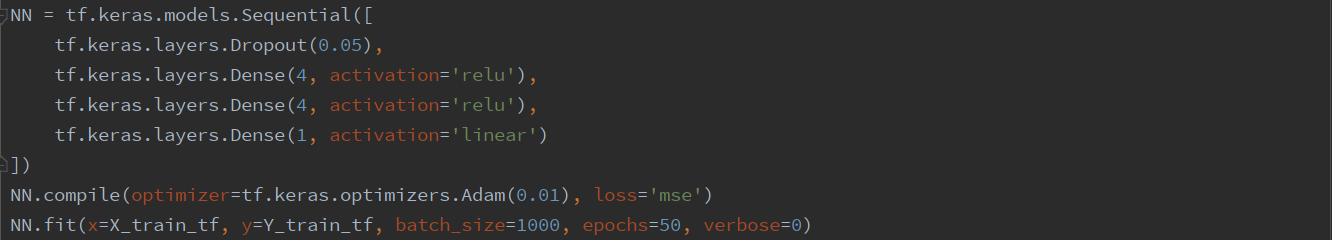
<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html#sklearn.ensemble.GradientBoostingRegressor>

### 2.3. Neural Networks

The feed-forward networks consist of an “input layer” of raw predictors, one or more “hidden layers” that interact and nonlinearly transform the predictors, and an “output layer” that aggregates hidden layers into an ultimate outcome prediction. Typically, we train neural networks using stochastic gradient descent (SGD), which evaluates the gradient from a small random subset of the data, called “batch”, rather than the entire training sample. Since neural networks have a high tendency of overfitting the data, a common regularization method called “dropout” can be used to mitigate this problem. “Dropout” refers to randomly dropping out some units with a probability during training process, preventing the model to rely on some specific neurons. Here I mainly tune the following 4 hyper-parameters for neural networks.

|  |  |
| --- | --- |
| Name | Description |
| dropout | Dropout rate. It’s the probability of randomly dropping out some units. |
| lr | Learning rate. It’s the step size of SGD. |
| batch\_size | The size of a random sample when calculating the gradients. |
| epoch | Number of training epochs. An epoch refers to one cycle through the full training dataset. Since we use SGD to train the network, an epoch consists of iterations. |

In the code, I use “tensorflow” package to build the network.



For example, the above is a network consisting of one dropout layer with 5% dropout rate and two hidden layers with 4 neurons in each layer and ReLU activation function. The learning rate, batch size, and number of epochs are set to 0.01, 1000, and 50, respectively.

For more details, please see

<https://tensorflow.google.cn/tutorials>