GBRT:

1. Numbers Estimator (**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.
2. Learning Rate (learning\_rate): Learning rate is a decreasing function of time. Two forms that are commonly used are a linear function of time and a function that is inversely proportional to the time *t*. learning rate shrinks the contribution of each tree. There is a trade-off between learning\_rate and n\_estimators.
3. Maximum Depth (max\_depth): maximum depth of the individual regression estimators. The maximum depth limits the number of nodes in the tree. This parameter is tuned for best performance; the best value depends on the interaction of the input variables.
4. Random State (random\_state): If int, random\_state is the seed used by the random number generator; If RandomState instance, random\_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.
5. Loss (loss): Loss function to be optimized. ‘ls’ refers to least squares regression. ‘lad’ (least absolute deviation) is a highly robust loss function solely based on order information of the input variables. There are other loss functions for example, ‘huber’ is a combination of the two. ‘quantile’ allows quantile regression (alpha is used to specify the quantile).

RF:

1. Numbers Estimator (**n\_estimators**): The number of trees in the forest to evaluate.
2. Random State (random\_state): If int, random\_state is the seed used by the random number generator; If RandomState instance, random\_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.
3. Maximum Depth (max\_depth): The maximum depth of the tree. If none, then nodes are expanded until all leaves are pure or until all leaves contain less than minimum samples split samples.

SVR:

1. Epsilon (epsilon): Epsilon parameter in the epsilon-insensitive loss function. Note that the value of this parameter depends on the scale of the target variable y.
2. Accepted Penalty(c): Penalty parameter C belongs to the error term. The penalty is a squared l2 penalty. The bigger this parameter, the less regularization is used.
3. Kernel: In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), kernel methods are a class of algorithms for [pattern analysis](https://en.wikipedia.org/wiki/Pattern_analysis), whose best known member is the [support vector machine](https://en.wikipedia.org/wiki/Support_vector_machine) (SVM). The general task of pattern analysis is to find and study general types of relations (for example [clusters](https://en.wikipedia.org/wiki/Cluster_analysis), [rankings](https://en.wikipedia.org/wiki/Ranking), [principal components](https://en.wikipedia.org/wiki/Principal_components), [correlations](https://en.wikipedia.org/wiki/Correlation), [classifications](https://en.wikipedia.org/wiki/Statistical_classification)) in datasets. In its simplest form, the kernel trick means [transforming data](https://en.wikipedia.org/wiki/Data_transformation) into another dimension that has a clear dividing margin between classes of data.

KNN:

1. Number of Neighbour (n\_neighbors): Number of neighbour determines how many boundaries we will create or how much similar data could be separate in a specific region.n\_neighbors is therefore just the number of neighbors "voting" on the test example's class.

|  |  |
| --- | --- |
| **Model** | **Error(RMSE)** |
| Linear Regression | 41163.23764482 |
| k-Nearest Neighbour | 47765.54137982 |
| Gradient Boosted Regression Trees | 44671.54111611 |
| Random Forest | 32757.47926456 |
| Support Vector Regression | 52250.7601211 |
| Neural Network | 41760.2730680 |