Zhecheng Li, 35688296 Shuo Wu, 21687050
Peng Hao, 53041238 Hao Chen, 34085220

Random Forests

1. Reason for choosing the algorithm

Random forest is an ensemble algorithm based on decision tree. It combines many "weak" learners (high bias) to a "strong" predictor and returns the average predictions as the final result. So, it reduces complexity of a model class prone to overfit hence has low bias and low variance.

Besides, the number of its parameters are relatively small and it is easy to find a bunch of appropriate parameters.

2. Problem

There are two main parameters for decision tree, which are max depth and minimum leaf number. As for random forest, it has number of learners and each learner's number of features besides the two parameters from decision tree.

So, what we need to do is to tune these parameters to find the best random forest which has the best prediction on dataset.

3. Tuning

The measure method of learner performance is MSE (mean squared error).

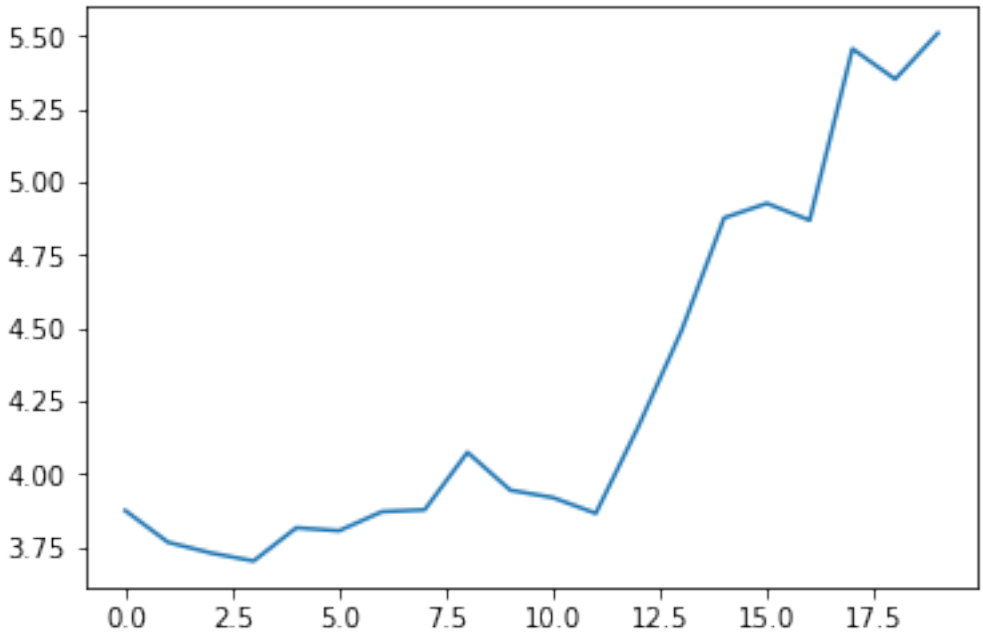
The tuning process is as follow:

The trains dataset has 100,000 samples with 14 features. And We use 90% to train the model and 10% for validation. The test dataset has 100,000 samples, too.

For decision tree:

I. minLeaf parameter is fixed, find best maxDepth

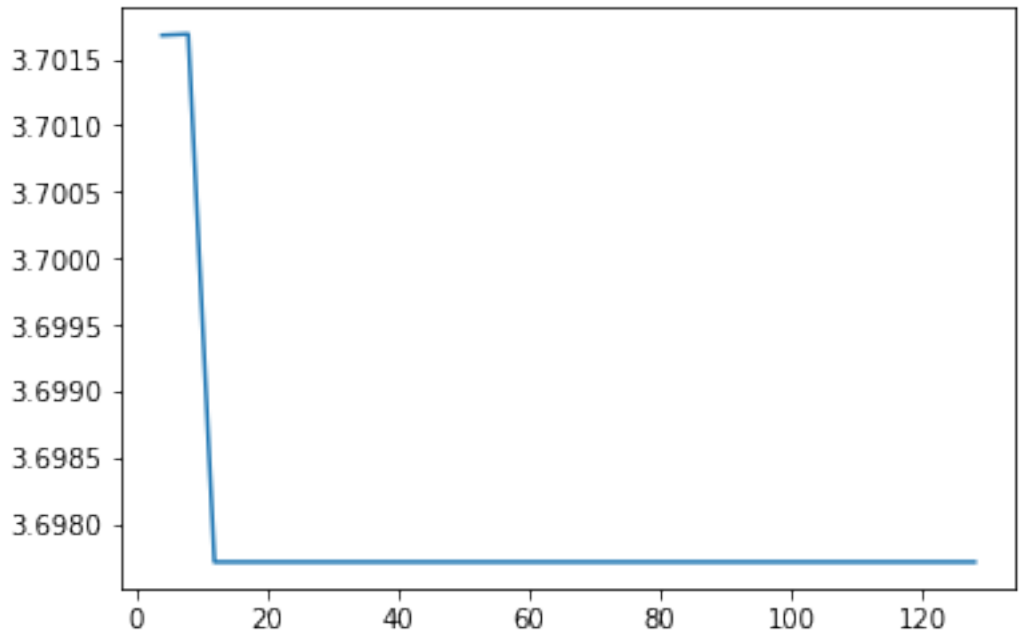
We set the range of maxDepth from 0 to 19, 20 different values in total. The plot is as follow:



According to the plot, we can see that when maxDepth is 3, the learner achieves the best performance, whose MSE is below 3.75.

II. maxDepth parameter is fixed, find best minLeaf

Here we choose 3 as the maxDepth according to results of I.
We set the range of minLeaf from 4, 8 … to 128, 33 different values in total. The plot is as follow:



According to the plot, we can see that when minLeaf is 12, the learner is much better than when minLeaf is 8. While if minLeaf becomes larger, the performance is not growing. So, we choose minLeaf to be 12.

We have two learners' predictions tested on Kaggle, here are the results:

Learner	maxDepth	minLeaf	MSE on Kaggle
1	3	12	3.52232
2	14	5	3.95651

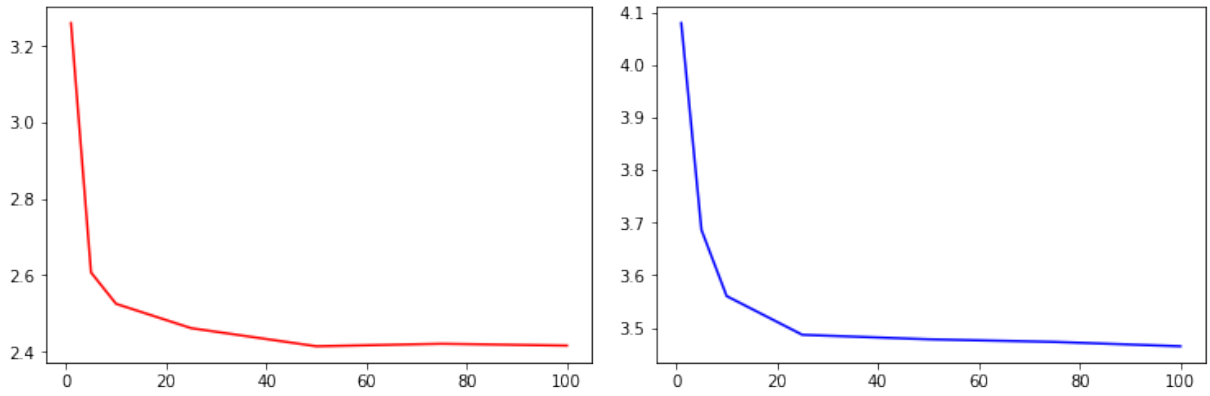
For random forest:

Here we have learner predicting on training dataset and validation dataset. The training dataset is the bagging of first 90,000 data points from X_test, Y_test and the validation dataset

is the last 10,000 data points from X_test, Y_test. Curve of training is red and that of validation is blue.

I. other parameters fixed, choose best number of learners

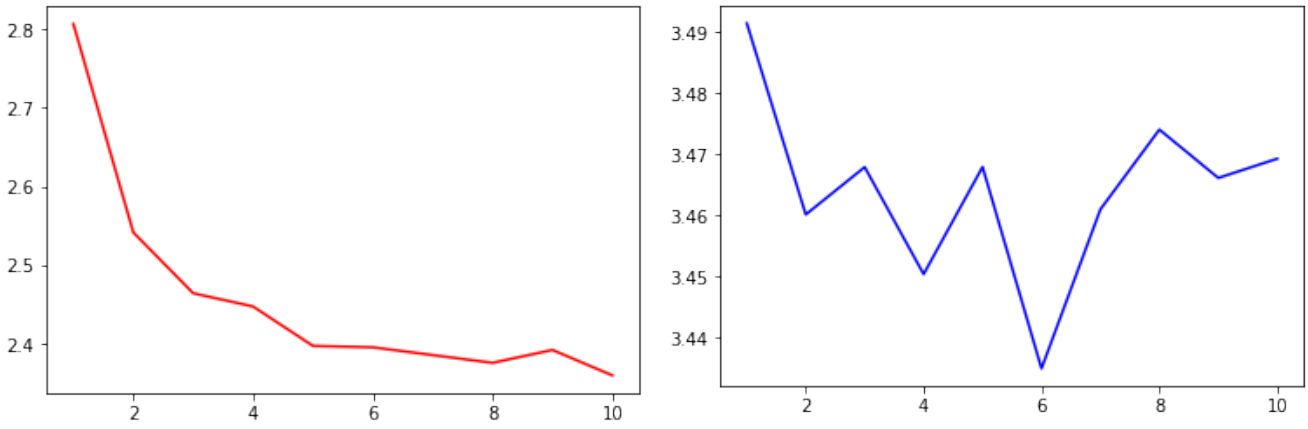
We set number of learners to be range in [1, 5, 10, 25, 50, 75, 100], maxDepth is 14, minLeaf is 5, number of features is 6. The plot is as follow:



From the plots, we can see that predictions become better with number of learners growing. However, after nBag becomes larger than 100, the improvement almost stops to grow. So we choose 100 for the final model.

II. other parameters fixed, choose best number of features

We set number of features to be range in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], maxDepth is 14, minLeaf is 5, number of learners is 100. The plot is as follow:

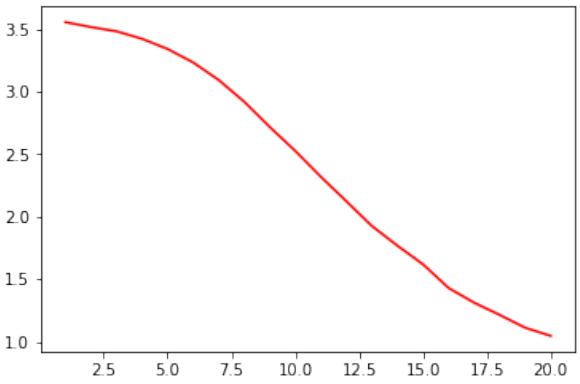


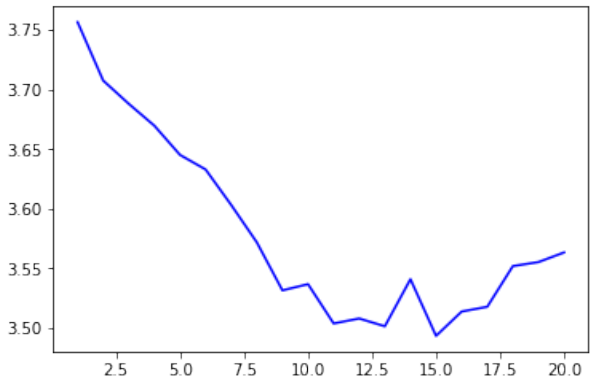
From the plots, we can see that predictions of training dataset become better with number of features growing. However, for the predictions of validation dataset, when number of features is 6, the random forest achieves the best performance. So, we choose number of features to be 6.

Noticing that increasing this feature will hurt the diversity of trained forest. Hence we tend to choose a smaller value if the performance is not degraded.

III. other parameters fixed, choose best maxDepth

We set maxDepth to be from 1 to 20, 20 different values in total, number of learners is 100, number of features is 6. The plot is as follow:



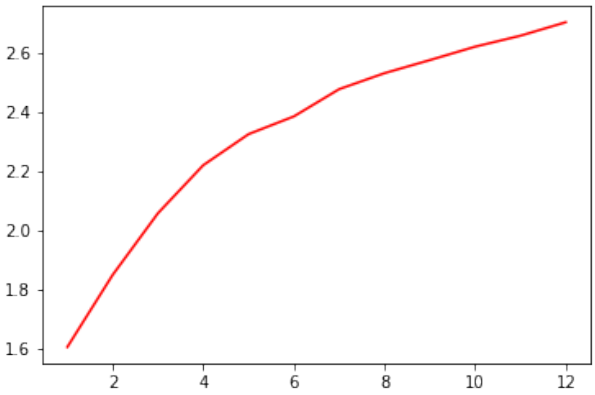


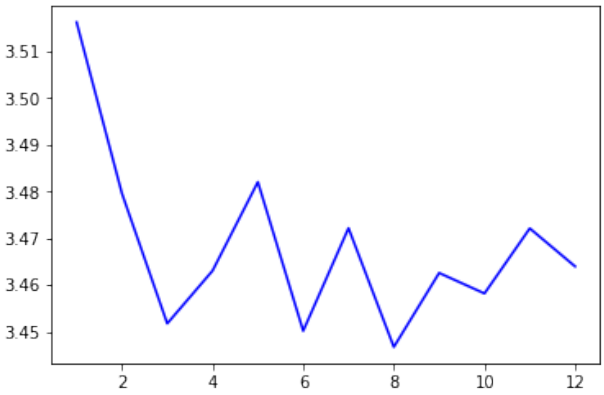
From the plots, we can see that predictions of training dataset become better with maxDepth growing. However, for the predictions of validation dataset, when maxDepth is 15, the random forest achieves the best performance. So, we choose maxDepth to be 15.

This parameter should be given and limited, otherwise the cost of computation is unacceptable.

IV. other parameters fixed, choose best minLeaf

We set minLeaf to be from 1 to 12, 12 different values in total, number of learners is 100, number of features is 6, maxDepth is 15. The plot is as follow:





From the plots, we can see that predictions of training dataset become better with minLeaf decreasing. However, for the predictions of validation dataset, when minLeaf is 8, the random forest achieves the best performance. So, we choose minLeaf to be 8.

Since the size of train dataset is not large, the value of this parameter will not be dramatically high.

4. Testing on Kaggle

According to the test above, now all the parameters for a random forest are determined.
We test different random forest on Kaggle, here are the result

Paramters	random forest 1	random forest 2	random forest 3
number of learners	25	50	100
number of features	8	6	6
maxDepth	15	14	15
minLeaf	6	5	8
MSE on Kaggle	3.42	3.39/3.40	3.36344

k-Nearest Neighbors

1. Reason for choosing the algorithm

There are several reasons for us to choose k-Nearest Neighbors method. Firstly, k-Nearest Neighbors method has clear theory background and it is relatively easy to implement, comparing to some complex methods. Secondly, this method is able to solve the non-linear problem. What is more, it requires less rely on the parameters and it shows less sensitive on outlier. Therefore, it is very suitable to us to implement the primary demonstration.

2. Problem

The main parameter in the k-Nearest Neighbors method is the value k. Hence, the problem of this section is to determine the relatively best k value by changing k value and calculate the Mean Square Error for each k value. And then compare the Mean Squared Errors to find the k value with the smallest Mean Squared Error.

What is more, due to the big amount of dataset, we also set a parameter r to represent the number of dataset to use in training, and compare the Mean Squared Errors for different r to select a suitable r value.

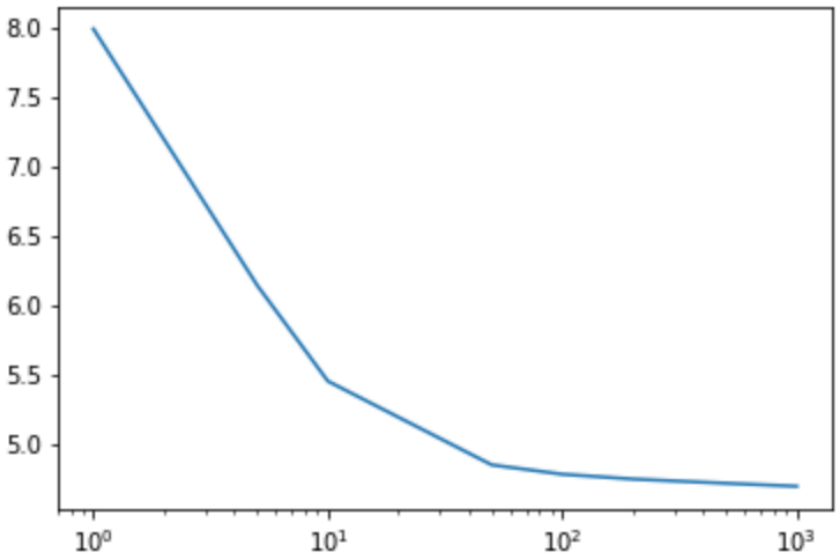
3. Tuning

Firstly, the measure method of learner performance is MSE(Mean Squared Error).

Secondly, we split the dataset into 2 parts, we utilize 90% of data to train the learners and utilize 10% of data for validation.

I. The size of dataset r is fixed to 10000, to find the best k value

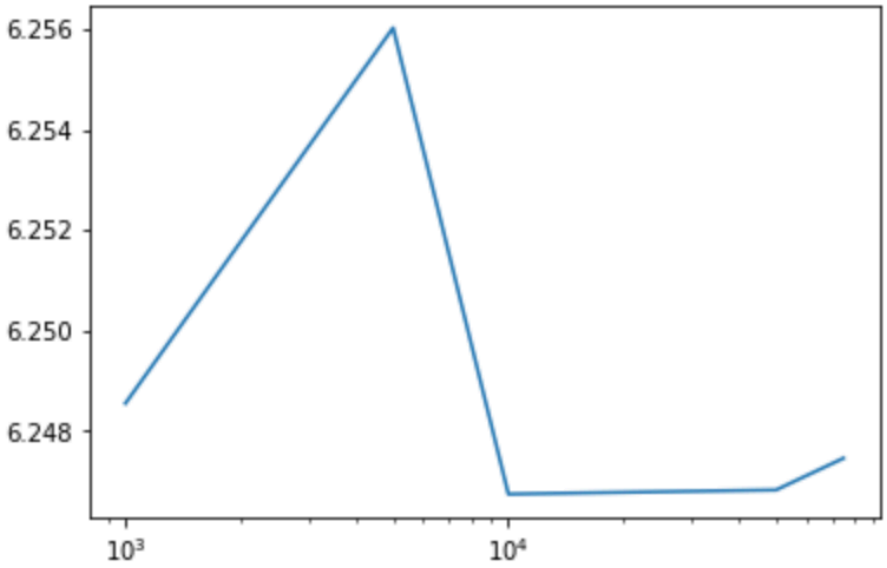
We set the k value in a range [1, 5, 10, 50, 100, 200, 500, 1000], and set the size of dataset to 10000 for training and 1000 for validation. And then we use each k value to train one learner and use the validation dataset to calculate the MSE. We choose the best k value with the lowest MSE.



As the MSE-k result shows, to achieve the lowest MSE, k should be as large as possible. However, taking the time cost considered, we select k=100 in the later medals because when k>100, the MSE starts to decrease slowly.

II. The k value is fixed to 100, to find the best size of dataset r value

We set the r value in a range [1000, 5000, 10000, 50000, 75000], and set the k value to 100 as discussed above. And then we use each r value to train one learner and use the validation dataset to calculate the MSE. We choose the best r value with the lowest MSE.



As the MSE-r plot shows, when r > 10000, MSE becomes nearly constant. Hence, we can choose r to 10000.

4. Testing on Kaggle

According to the tests above, we choose k = 100 and r = 100000 as parameters to train the model and test on Kaggle, and the result is shown below.

Learner	K value	Size of dataset	MSE on Kaggle
k-Nearest Neighbors	100	100000	3.55863

Neural Networks

1. Reason for choosing the algorithm

Comparing with other machine learning algorithm, the Neural Networks method provides more complex and powerful ability on the non-linear problems.

2. Problem

For Neural Networks, firstly, we are supposed to determine the number of hidden layer and then we need to determine the number of nodes of the hidden layer. Moreover, we are required to select an appropriate learning rate.

3. Tuning

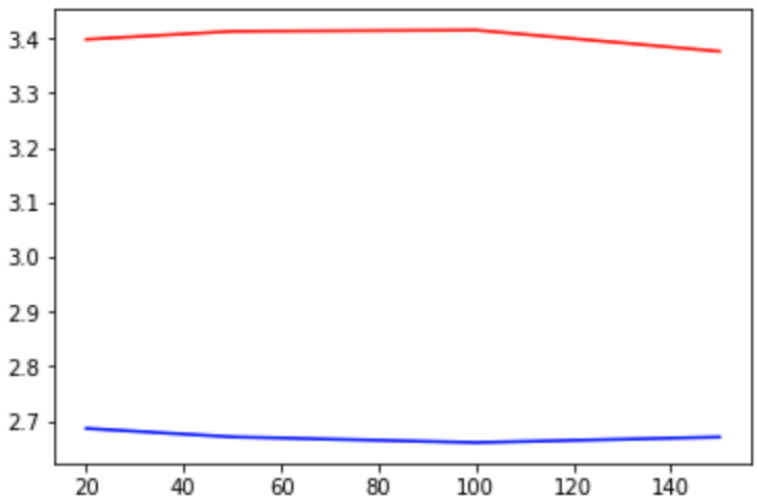
Firstly, the measure method we choose is the same as the methods before, the MSE(Mean Squared Error).

Secondly, we select 1 as the number of hidden layer.

Thirdly, we use 75% of data for training and 25% of data for validation.

I. Select number of nodes in the hidden layer

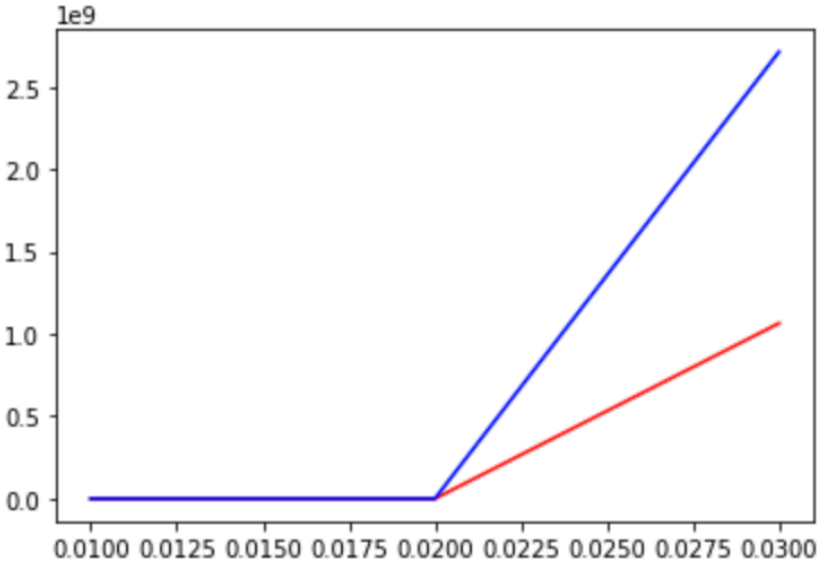
We set the number of nodes in the hidden layer in a range [20, 50, 100, 150], and set the size of dataset to 5000 for training and 1000 for validation. And then we use each value to train one learner and use the validation dataset to calculate the MSE. We choose the best value with the lowest MSE.



As the MSE-number of nodes plot (training: blue, validation: red) shows, number of nodes in hidden layers has little effect on prediction accuracy. Hence, we choose it to 100. Because we have 14 features as input and 1 output, our model is a [14-100-1] neural network.

II. Select learning rate

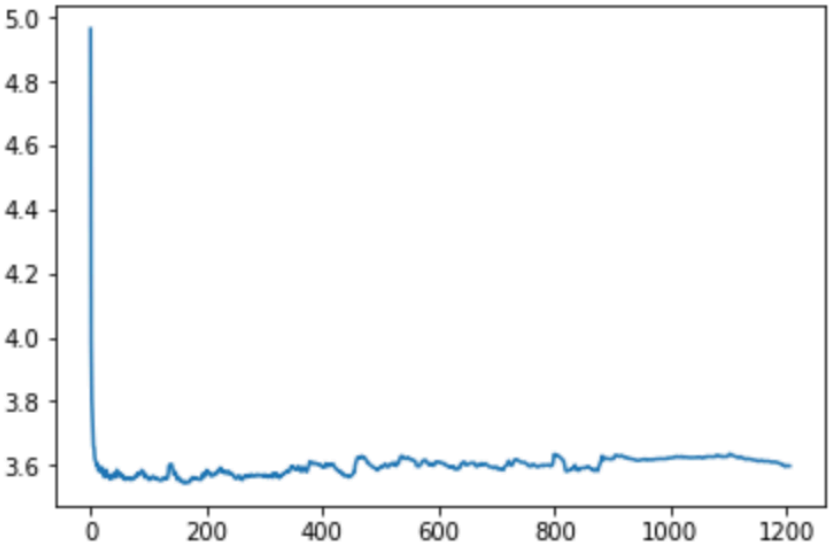
We set the learning rate in a range [0.01, 0.02, 0.03], and set the size of dataset to 5000 for training and 1000 for validation. And then we use each value to train one learner and use the validation dataset to calculate the MSE. We choose the best value with the lowest MSE.



As the MSE-learning rate plot shows, to learning rate, 0.01 and 0.02 work in similarly degrees, while MSE decreases more quickly with 0.02. the result here shows 0.03 leads to a large MSE. Hence, we choose learning rate to be 0.01.

III. Try a 4-layer Neural Networks

We choose a 4-layer neural networks model [14:50:50:1] with 0.01 as the learning rate.



As the plot shows above, we find the 4-layer model works similarly with 3-layer model, therefore, we still choose 3-layer model.

4. Testing on Kaggle

According to the tests above, we choose 3-layer model, 100 nodes in hidden layer and 0.01 as learning rate. Then train the model and test on Kaggle, and the result is shown below.

Learner	Nodes in hidden layer	Learning rate	Layers	MSE on Kaggle
Neural Networks	100	0.01	3	3.58476

Libraries used in the project

algorithms	Library
Random forest	mltools
kNN	mltools
Neural Network	mltools, tensorflow, matlab_DeepLearnToolbox

Distribution of Responsibilities

Random forest part is completed by Shuo Wu and Zhecheng Li. Shuo Wu is mostly responsible for coding and Zhecheng Li is mostly responsible for writing up the final report. They discussed together and uploaded results on Kaggle to see the performances.

K-Nearest Neighbors and Neural Networks parts are completed by Peng Hao and Hao Chen. Peng Hao is mostly responsible for coding and Hao Chen is mostly responsible for writing up the final report. They discussed together and uploaded results on Kaggle to see the performances.