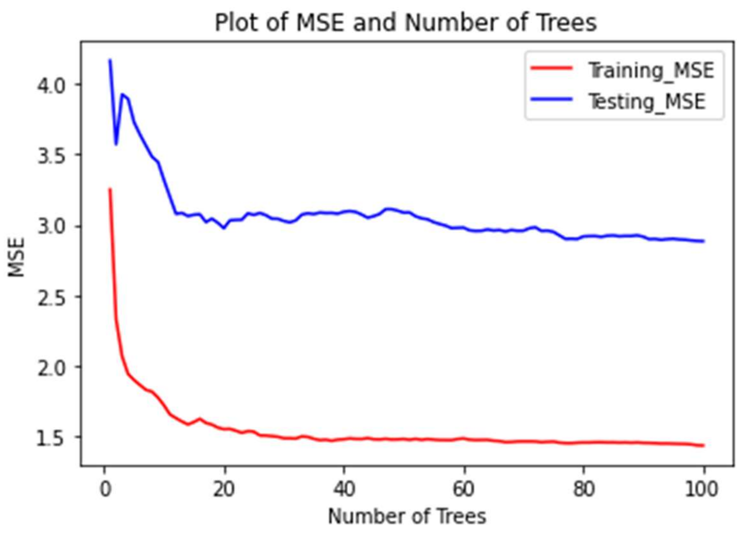
```
Best Parameter Pairs:  {'max_depth': 7, 'max_features': 7, 'min_samples_leaf': 5, 'n_estimators': 20}
```

Then, I define a function random_forests (numtrees, m), which use RandomForestRegressor in sklearn to do the task and return its training and testing mse by metrics.mean_squared_error ().

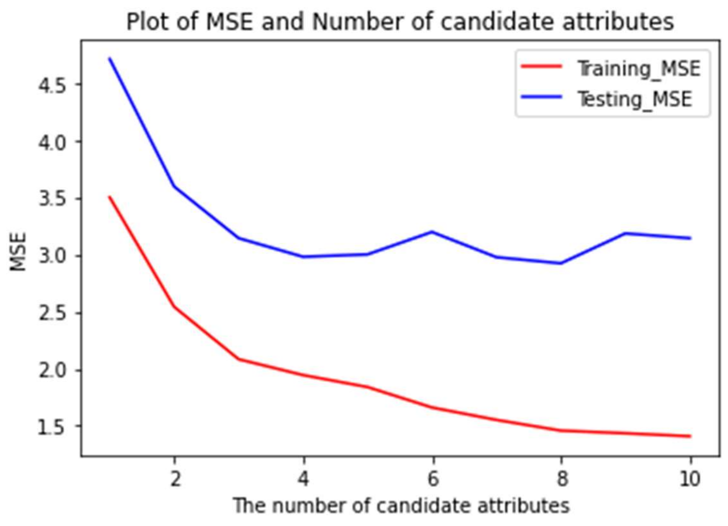
Result Analysis:

1. First, I change the number of trees from 1 to 100, while setting max_depth as 7, m as 7, min_samples_leaf as 5, according to the GridSearchCV. The result is shown below:



From this figure, we find that when the number of trees increase, the training error and testing error start to decrease at first and then be stable. The number of trees means data-level randomness, has no relationship with complexity since each tree may have roughly the same depth in the whole random forest, for example, they may have the depth of only 2 or 3.

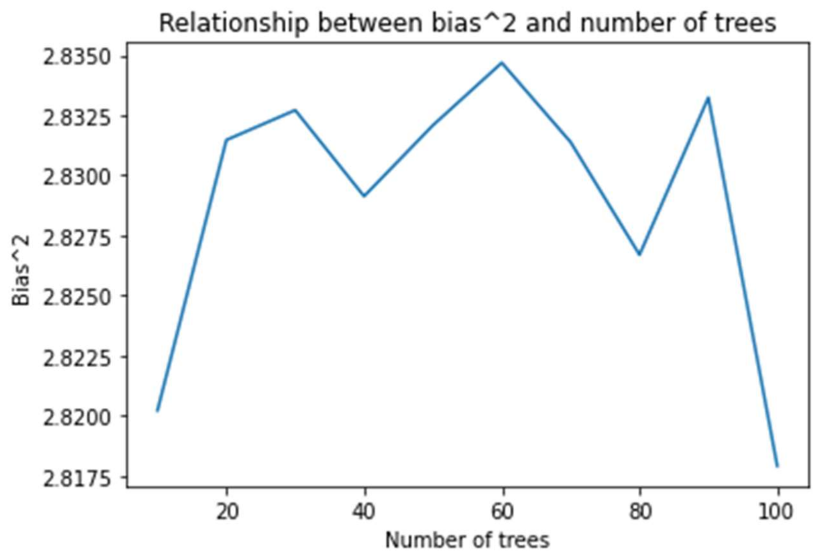
2. Then, I change value of m from 1 to 100, while setting max_depth as 7, number of trees as 20, min_samples_leaf as 5, according to the GridSearchCV. The result is shown below:



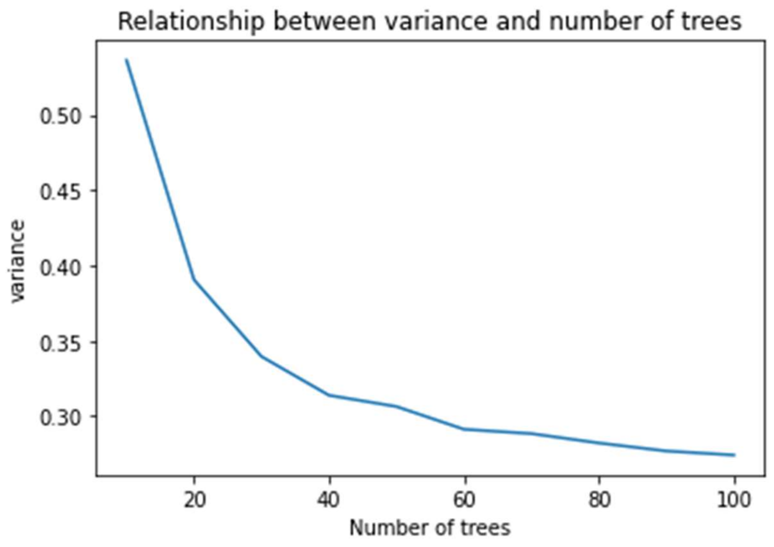
From this figure, we find that with the increase of candidate attributes, the training error is always decreasing. The testing error decreases at first and start to be stable when m = 3 or 4, which is same as we discussed in lecture slides that m = N/3 is always a good choice.

2.1.5 Bias^2 and Variance

In this question, I use bias_variance_decomp () provided by mlxtend to get the average bias and average variance throughout the whole dataset.
Below is the curve of bias w.r.t the number of trees:



From the figure, we can obtain that there's no obvious relationship between bias and the number of trees, which is consistent with the statement provided in lecture slides that different trees in the random forests are independent and the overall model complexity is not increased, so we can't expect increasing the number of trees can reduce bias.
However, for the curve of variance and the number of trees, it's different:



From the figure, we can see that increasing the number of trees can indeed decrease the variance. Since random forests introduce both data-level randomness and model-level randomness (From random feature selection), so there is a guarantee that variance will be reduced.
Mathematical Proof:
We can write a random forest as the formula below:

$$T(x) = \frac{1}{B}\sum_{i=1}^{B} T_{i,Z_i}(x)$$

where B is the number of trees, T_{i,Z_i} is the i-th decision tree, Z_i is the training sample.

We can easily obtain that for a certain x, $T_{i,Z_i}(x)$ follows the same distribution, so increasing the number of trees has no contribution to reduce the model bias.

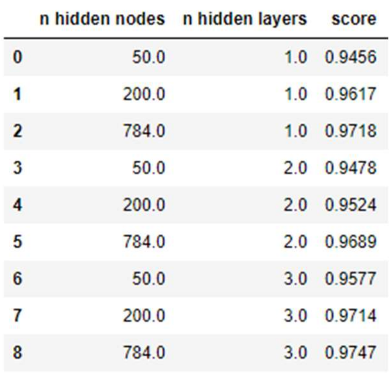
However, $T_{i,Z_i}(x)$ is dependent to each other, since the samples are drawn from the same dataset, and they use the same algorithm to get each decision tree. If we set the correlation coefficient to be ρ, and variance of each tree to be σ^2, we can get the variance of the whole random forest:

$$\begin{aligned} Var(T(x)) &= [\frac{1}{B}, \cdots, \frac{1}{B}] \begin{bmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \cdots & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{B} \\ \vdots \\ \frac{1}{B} \end{bmatrix} \sigma^2 \\ &= \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \end{aligned}$$

From the formula derived, we can see that if we increase the number of trees B, we can significantly reduce the variance of the random forest. In conclusion, our conclusions from processing the dataset and mathematical formulation are the same.

2.2 Handwritten Digit Recognition

In this question, I use load_mnist.py to load the train/test data, and use MLPClassifier from sklearn to construct a Multi-Layer Perceptron neural network, with hyperparameters: (solver = 'adam', activation = 'relu', alpha = 0.05, random_state = 3020, max_iter = 100, verbose = False, learning_rate_init = 3e-4, batch_size = 512), which are figured out by GridSearchCV. The result is shown below:



	n hidden nodes	n hidden layers	score
0	50.0	1.0	0.9456
1	200.0	1.0	0.9617
2	784.0	1.0	0.9718
3	50.0	2.0	0.9478
4	200.0	2.0	0.9524
5	784.0	2.0	0.9689
6	50.0	3.0	0.9577
7	200.0	3.0	0.9714
8	784.0	3.0	0.9747

From the chart, we can see that if we fix the number of hidden nodes, with the increase of the number of hidden layers, we may not have a higher prediction accuracy. In contrast, if we fix the number of hidden layers, with the increase of the number of hidden nodes, we would have a higher prediction accuracy.