RUTGERS

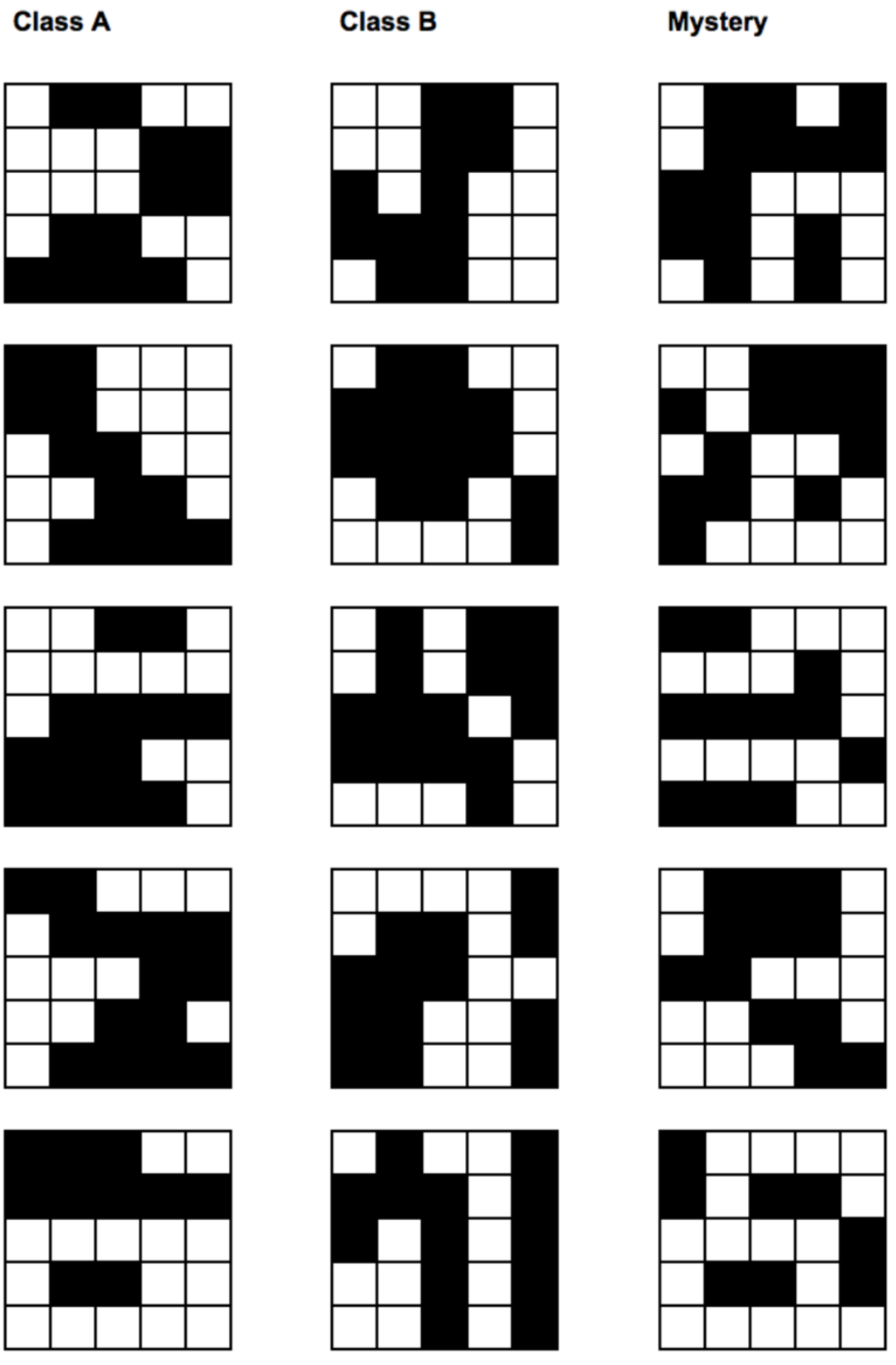
THE STATE UNIVERSITY
OF NEW JERSEY

CS520 - INTRO TO ARTIF INTEL

Final: Question 3 - Classification

Name:
Xinyu Lyu(xl422)

The following gives five instances of Class A images, five instances of Class B images, and five instances of unlabeled images. (Each image is a 5x5 grid of black or white cells.)



a) Construct a model to classify images as Class A or B, and train it on the indicated data. Specify your trained model. What does your model predict for each of the unlabeled images? Give the details of your model, its training, and the final result. Do the predictions make sense, to you?

I construct a RandomForestClassifier model to help me classify images as Class A or B. I think of the problem as a two-category problem. So, with the mystery images as input, my model will predict if each unlabeled image in mystery belongs to Class A or Class B. In other the information of the model is the vector of each image, and the output will be the label of 0(ClassA) or 1(ClassB).

Given the shortcomings of decision trees that are prone to overfitting, random forests use voting mechanisms from multiple decision trees to improve decision trees.
And the steps to generate random forests are shown followings:
1. Generate n samples from the sample set by resampling
2. Assuming that the number of sample features is an excellent k feature in a for n samples, and obtain the best segmentation point by establishing a decision tree.
3. Repeat m times to generate m decision trees
4. Most voting mechanisms to make predictions

First, for each image in Class A and Class B, I convert it to a vector with dimension as (1,25). And each element stands for a pixel in the image. And for each black pixel, I represent it as 0, and white pixel as 255. For example, for the first image in Class A, I convert it into
A1 = [255, 0, 0, 255, 255, 255, 255, 255, 0, 0, 255, 255, 255, 0, 0, 255, 0, 0, 255, 255, 0, 0, 0, 0, 255].

The most critical problem in training is the amount of training set is too small, so I use some strategy to expand the dataset. I rotate or flip the image to get more datasets after the rotate or flip, the dataset extend five times from A/B1 to A/B25. For Class A, A1-A5 means the original image data, and A6-A10 means the image data after the flip. A11-A15 means the image data after the first closes rotate. A15-A20 means the image data after second closewise rotate. A20-A25 means the image data after third closewise rotate. Before using these data for training, I use standard scaler operation to the standard the dataset. Then I separate it into train data and test data with the scale as 9:1. Commonly, the size should be larger, but the training set is too small, I can't separate more data for testing. And after the training, I use test data to make the cross-validation testing to evaluate the model. With the evaluation results, I can adjust the parameters to make it more perfect. And when we use the Mystery as the test data, we find that the final product result is [ClassB, ClassB, ClassA, ClassA, ClassB], which means classify the first two images to ClassB and the next two images as ClassA and the last one as ClassB. The result makes sense to me.

b) The data provided is quite small, and overfitting is a serious risk. What steps can you take to avoid it?

The primary reason for overfitting problem is the amount training data is small which can't satisfy the training needs. So, the similar strategy is to expand the training data, which I have

mentioned in a) about the training data.
Second, there are some methods to make up for the overfitting problem, to some extent. And I use two kinds of curves to help me reduce the model overfitting.

Validation curve is a curve to determine training and test scores for varying parameter values. It computes scores for an estimator with different amounts of a specified parameter. This is similar to grid search with one parameter. However, this will also calculate training scores and is merely a utility for plotting the results. According to the validation curve, I find the best parameters for the model for the best prediction results.

For the Random Forest Classifier model, there are three adjustable parameters to improve the prediction ability of the model.

A. max_features
The number of features to consider when looking for the best split. Increasing max_features generally improves the performance of the model because at each node we have more options to consider. And in Sklearn, we have some options for this parameter.

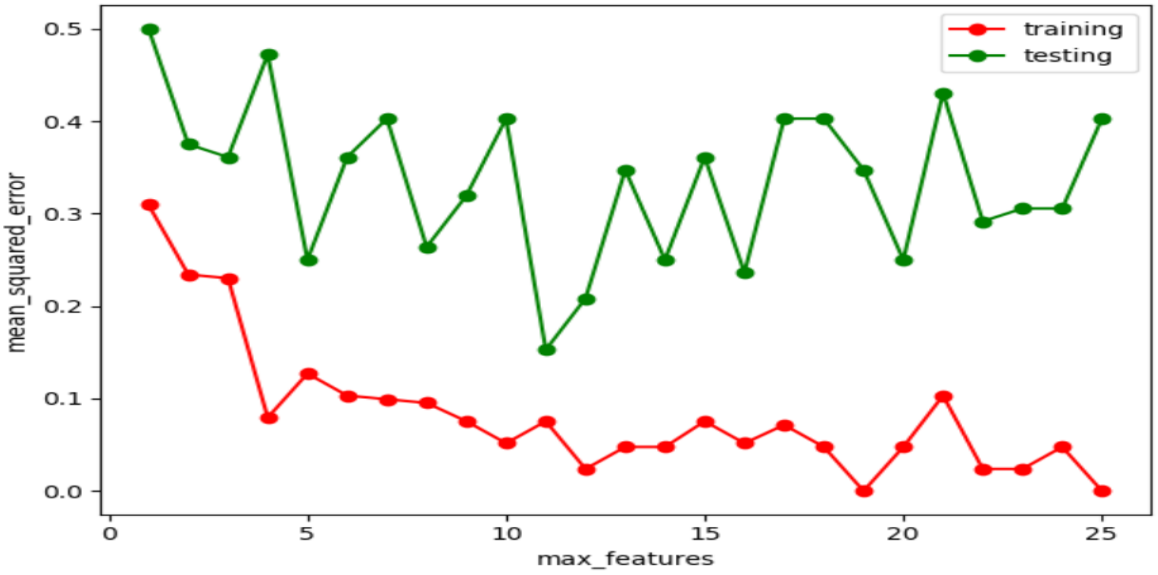
1. Auto/None： Simply select all the features and use them for each tree. In this case, there is no limit to each tree.

2. sqrt： This option is the square root of the total number of features that each subtree can take advantage. For example, if the total number of variables (functions) is 100, only 10 of them can be made from each subtree. "log2" is another similar type of option.

3. Integer number: Consider max_features features at each split.

4. Float number: Max_features is a fraction and int(max_features * n_features) features are considered at each split.

Experiments:



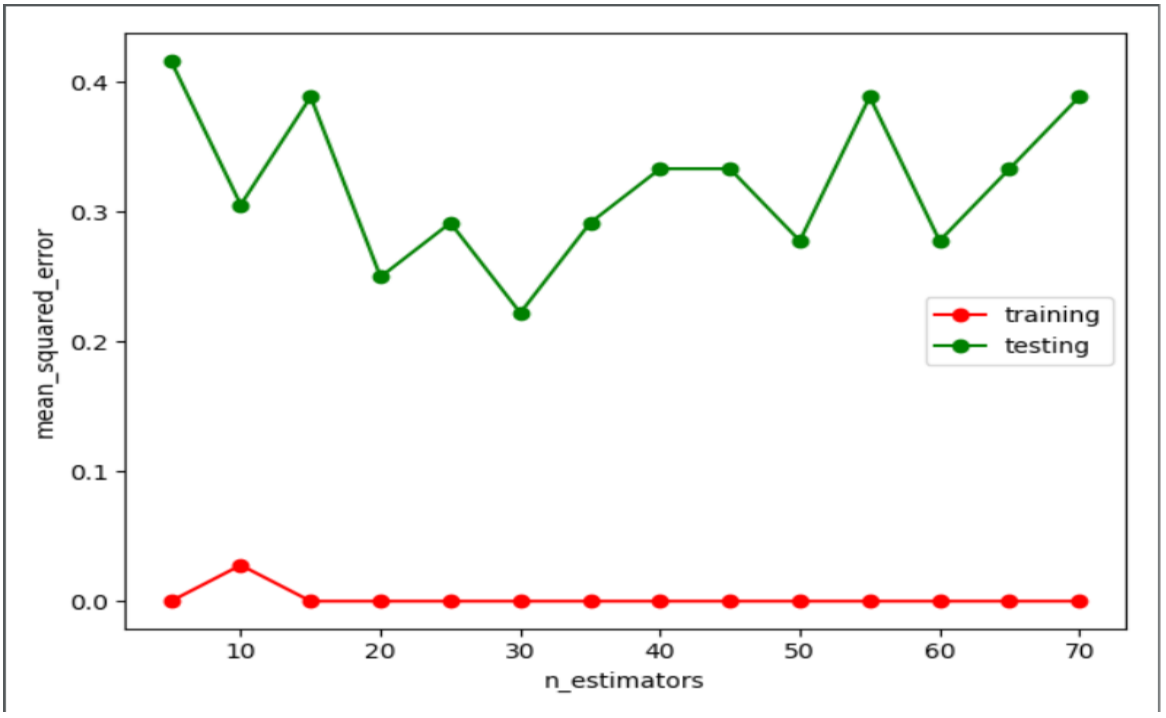
In the experiments, I set the range for max_features as [1,2,3,4,...,25], which is the X-axis of the figure above. And the Y axis is the mean squared error of the model testing with different max_features. From the result, we can find out when select 11 as max_features, the model behaves best; we set the max_features as 11.

B. n_estimators

The number of trees in the forest.

Before using the maximum number of votes or the average to predict, you need to set the number of subtrees. More subtrees can make the model perform better, but at the same time, it will make your code slower. I should choose the highest possible value, which makes my predictions better and more stable.

Experiments:

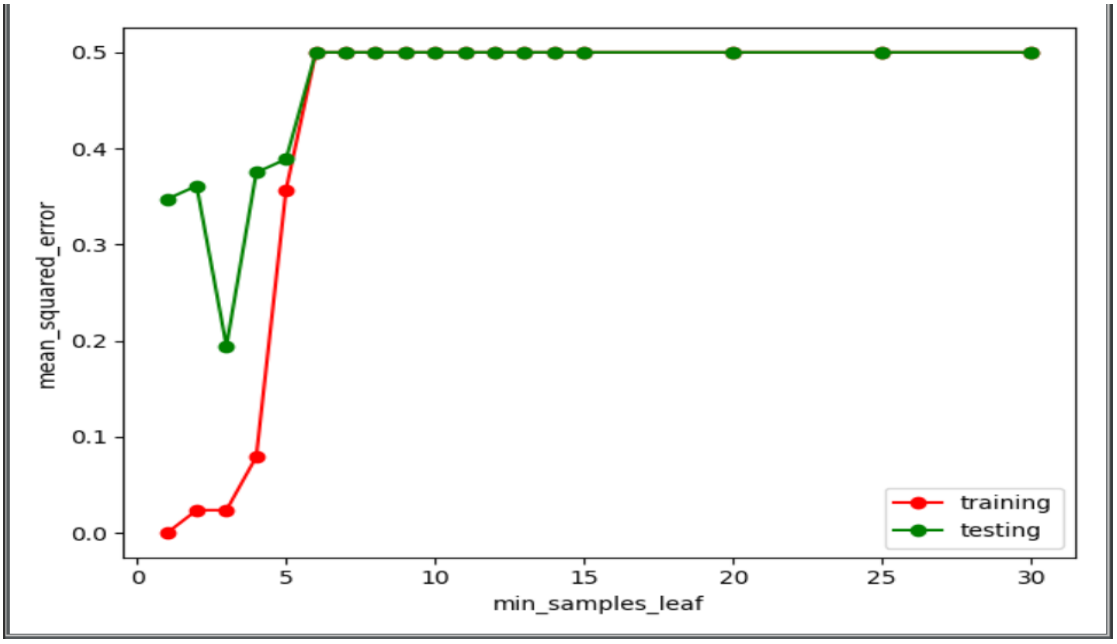


In the experiments, I set the range for n_estimators as [5,10,15,20,25,30,35,40,45,50,55,60, 65,70], which is the X-axis of the figure above. And the Y axis is the mean squared error of the model testing with different n_estimators. From the result, we can find out when select 30 as n_estimators, the model behaves best; we set the n_estimators as 30.

C. min_sample_leaf

A leaf is the end node of a decision tree. Smaller leaves make it easier for the model to capture noise in the training data. In general, I prefer to set the minimum number of leaf nodes to be higher than 50. In my case, I should try a variety of leaf sizes to find the best one.

Experiments:



In the experiments, I set the range for min_sample_leaf as [1,2,3,4,5,10,15,20,25,30], which is the X-axis of the figure above. And the Y axis is the accuracy of the model testing with different min_sample_leaf. From the result, we can find out when select four as min_sample_leaf, the model behaves best; we set the min_sample_leaf as 4.

c) Construct and train a second type of model. Specify its details. How do its predictions compare to the first model? Are there any differences, and what about the two models caused the differences?

For the second model, I train an MLP. A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of, at least, three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from direct perception. It can identify data that is not linearly separable.

Its predictions are not as good as the previous one. Because unlike the Random Forest, the MLP has too many adjustable parameters which are just like the DNN. I try my best to adjust the parameters and the test results is [ClassA,ClassA,ClassB,ClassA,ClassA]. I use the mean_squared_error scores to evaluate the model compared to the first model. And the mean_squared_error for the Random Forest is less than the MLP, so I think the first is better in predicting at present. But, I will make an effort to adjust the parameters of the MLP to make it behaves better after the exam!

These two kinds of Classifier are very different from each other. Random Forest integrated the classical Decision Tree, and MLP comes from the Neural Network. These two models are both excellent models, and they behave differently with different kinds of application scenarios. When the amount of data is enormous, MLP acts better, because of the advantage of DNN which is popular nowadays. However, in some specific application occasion, the classical models like Random Forest may behave better based on the solid

mathematical foundation.