ECS171 Fall 2017

HW3 Report

Gabriel Yin

999885129

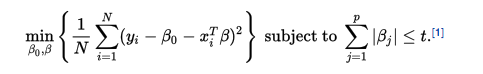
Problem 1

Optimal constrained parameter value: 0.00694

10-Fold cross-validation error: 0.03634

Number of non-zero features: 58

In Linear Regression, we aim to minimize the squared error. In Lasso method, we add a new constrain to the equation to prevent the value from going beyond a certain value. More specifically, we are trying to solve the following equation (Image taken from Wikipedia):



In this case, yi is the actual value of i-th sample, and xiTB is the predicted value of the i-th input value. We add a new value of Beta as the penalty constrain and we further define that the sum of Beta cannot exceed a certain value t. In Ridge method, however, there is no constrain for the Beta value. I choose to use Lasso method to solve this problem. To get the optimal constrained parameter value, I used bic approach (please see code for details).

Problem 2

90th Confident interval Lower: 0.8345 Upper: 0.9678

95th Confident interval Lower: 0.8123 Upper: 0.9678

First of all, as we are dealing with a large dataset, we can assume that the data is distributed normally (i.e. normal distribution). To solve this problem, I used Ridge method instead of the first one. Then I use bootstrap method to calculate 90th and 95th confidence interval of the data. To do so, I first set the iteration number to be 1000. Then for each iteration, I randomly take a index (using np.random.choice) and use this to split the training dataset x and training dataset y. Similarly, based on the previously acquired training index, I get the test index and use it to split the test set value. In each iteration I calculate the score using prediction function and append it to a list.

To calculate confidence interval, I first calculate p, a percentile indicator.

P = 100 \* ((1 – alpha\_val)/2.0) and lower can be calculated as

low = max(0.0, np.percentile(data, int(p))). Similarly, to calculate the upper bound, p = 100 \* (alpha\_val + ((1.0 – alpha\_val)/2.0)) and upper = min(1.0, np.percentile(data, int(p))).

Problem 3

Based on my code, the result is 0.3936.

Problem 4

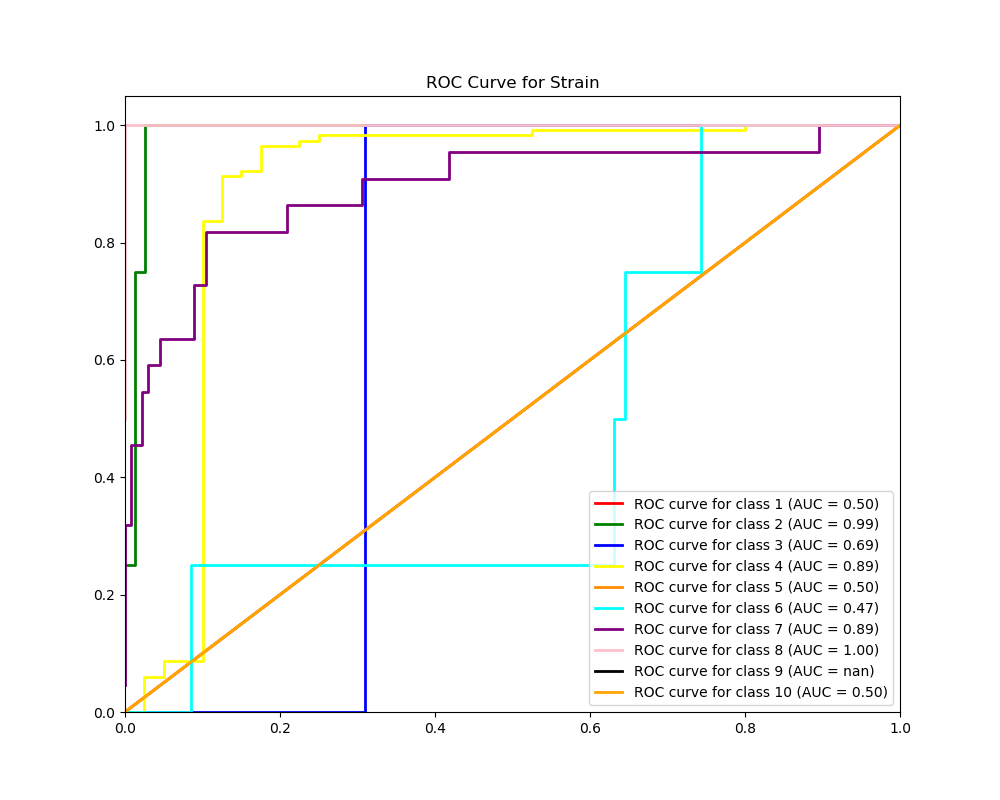
Strain: Best AUC: 1 AUPRC for the best AUC: 0.9342

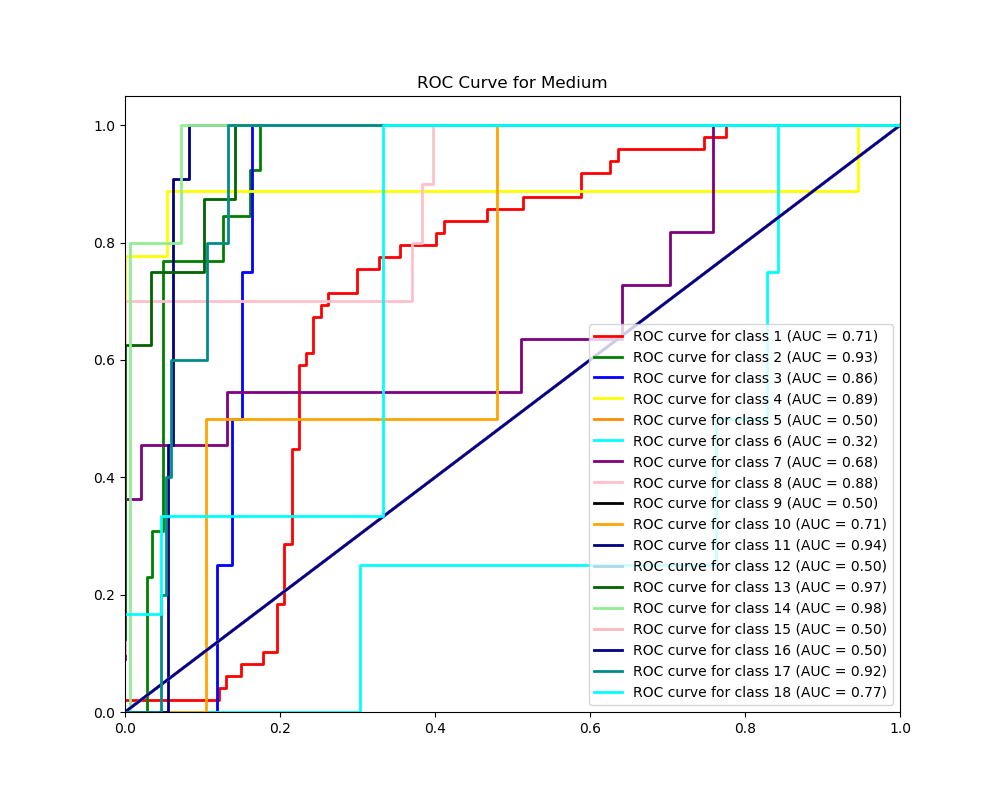
Medium: Best AUC: 1 AUPRC for the best AUC: 0.8637

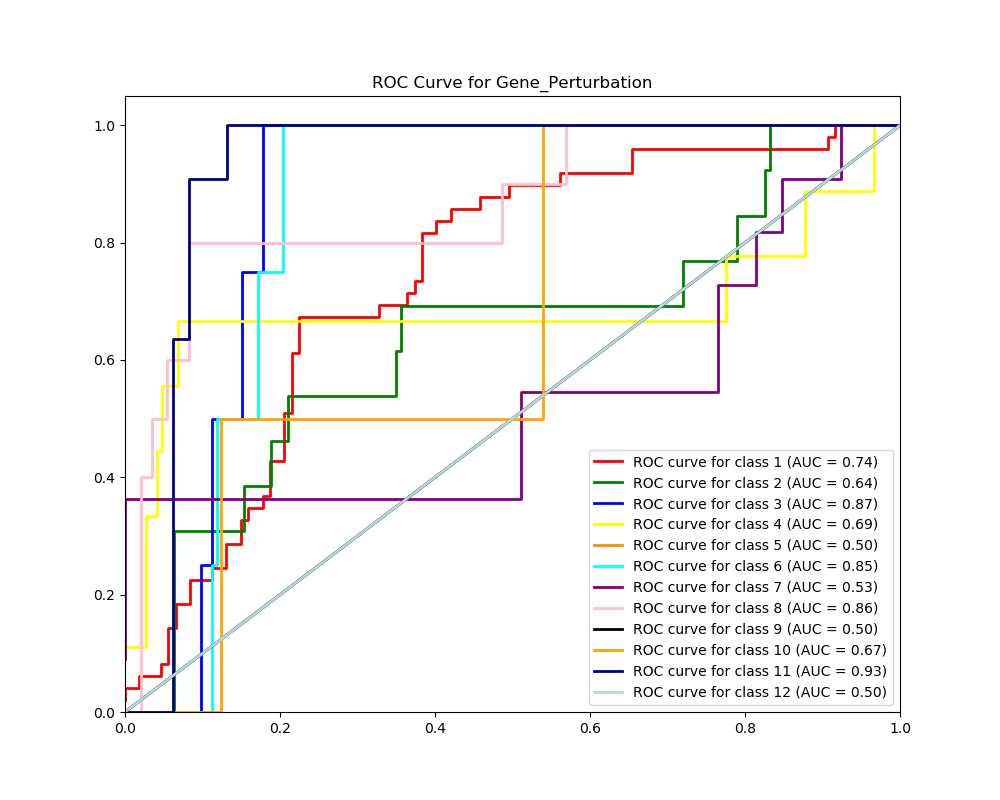
Environmental Perturbation: Best AUC: 1 AUPRC for the best AUC: 0.9134

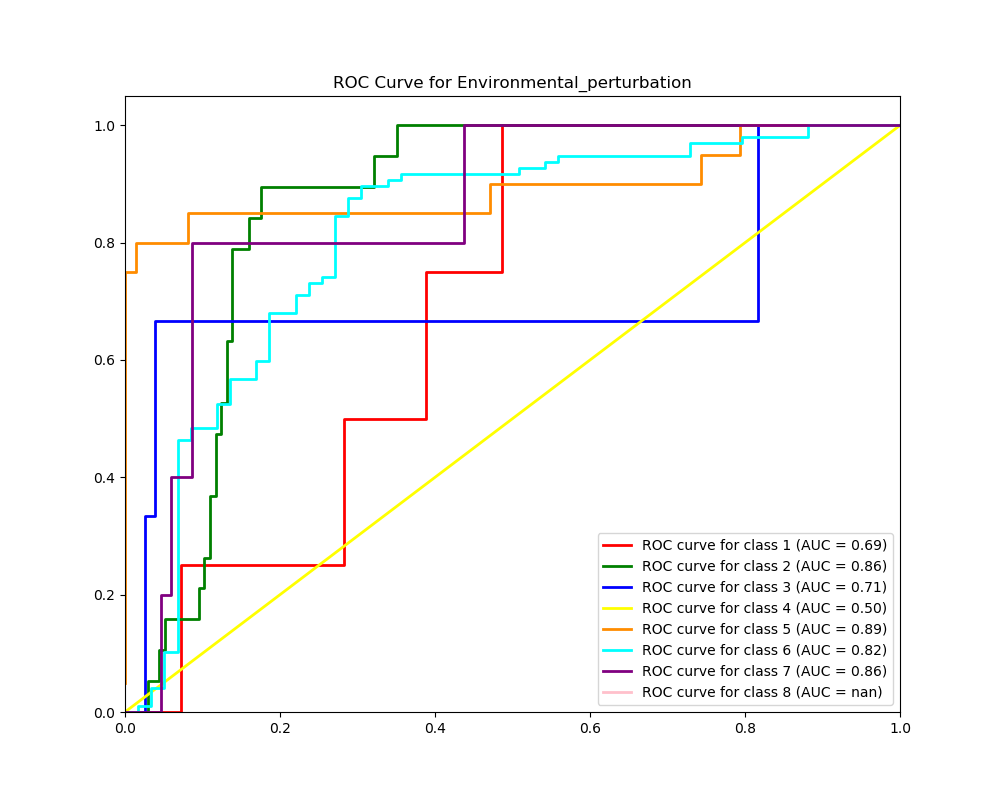
Genetic Perturbation: Best AUC: 1 AUPRC for the best AUC: 0.9231

ROC Curve Graphs:

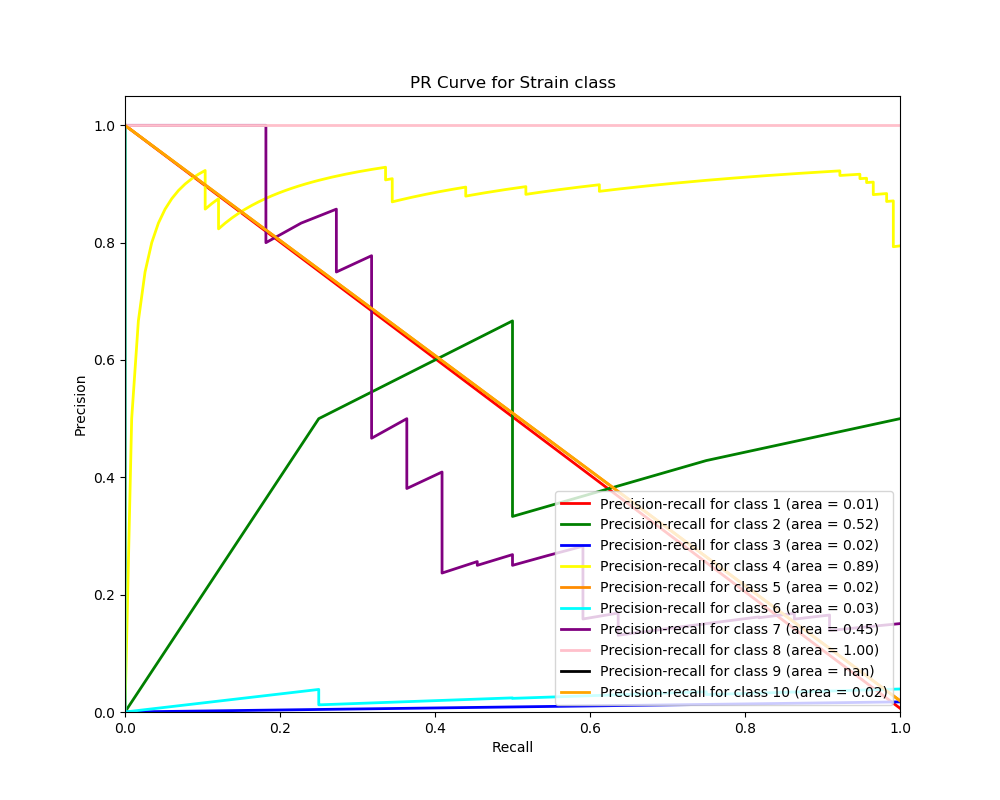


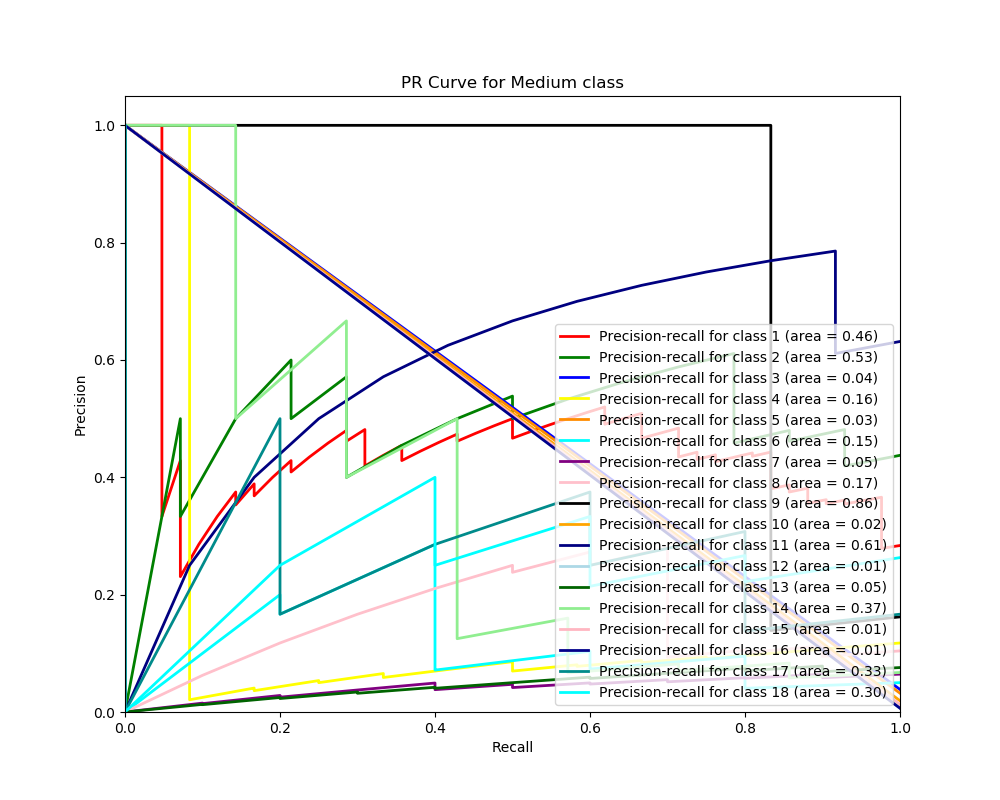


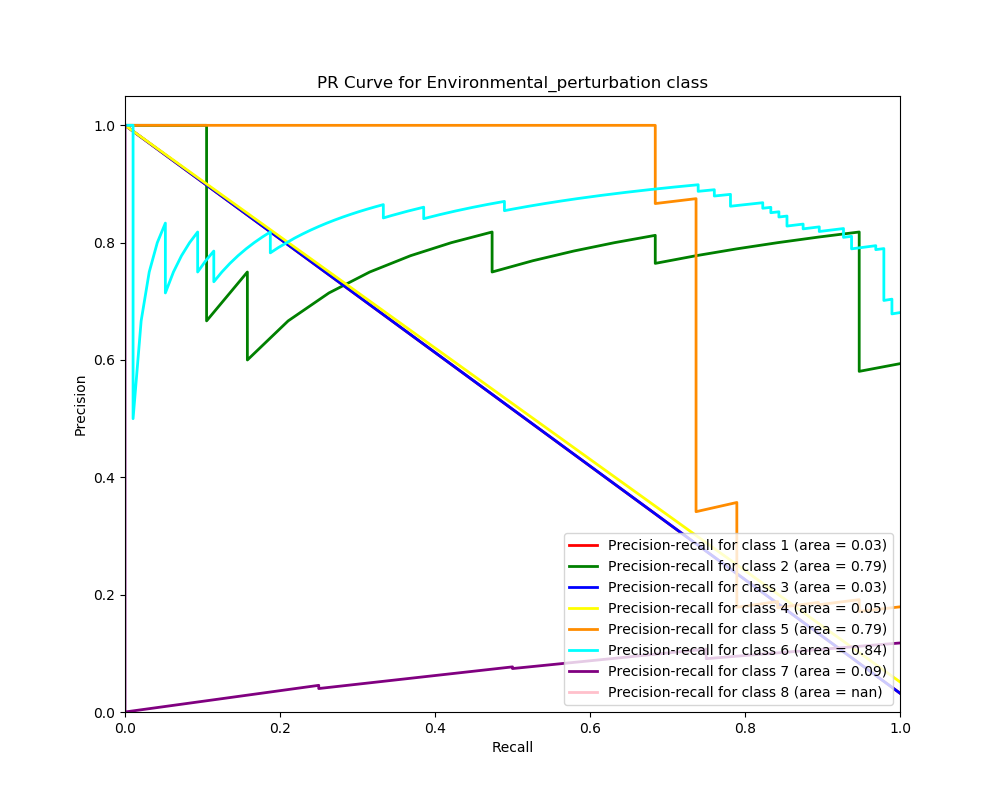


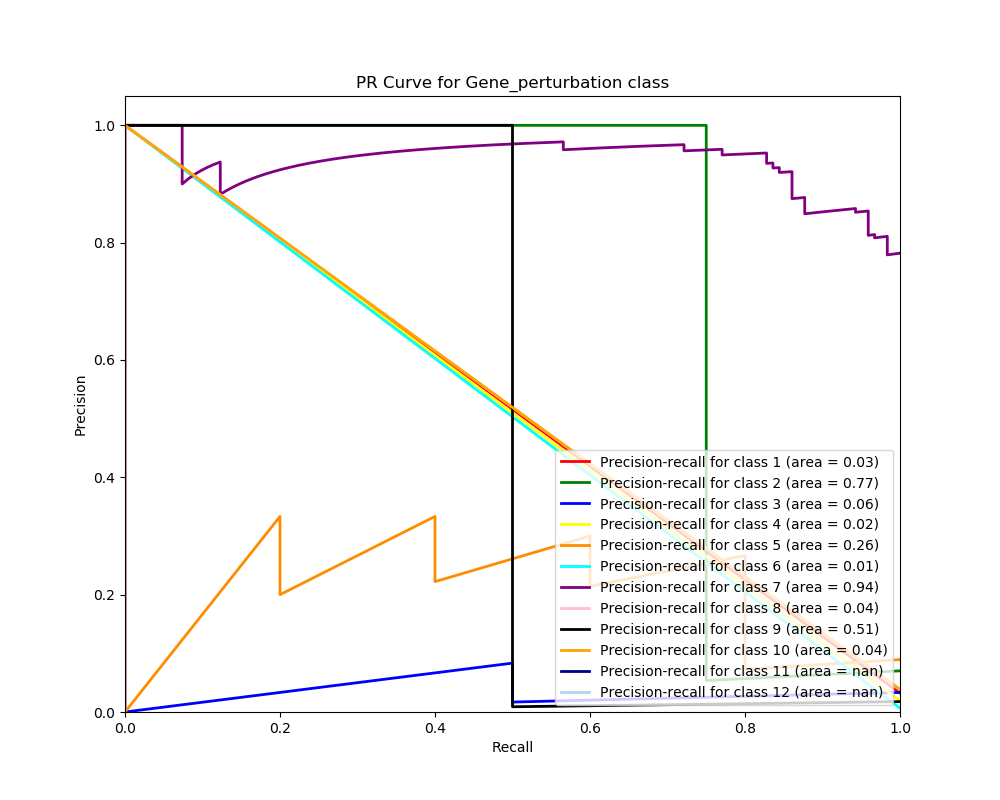


PR Curve Graphs:





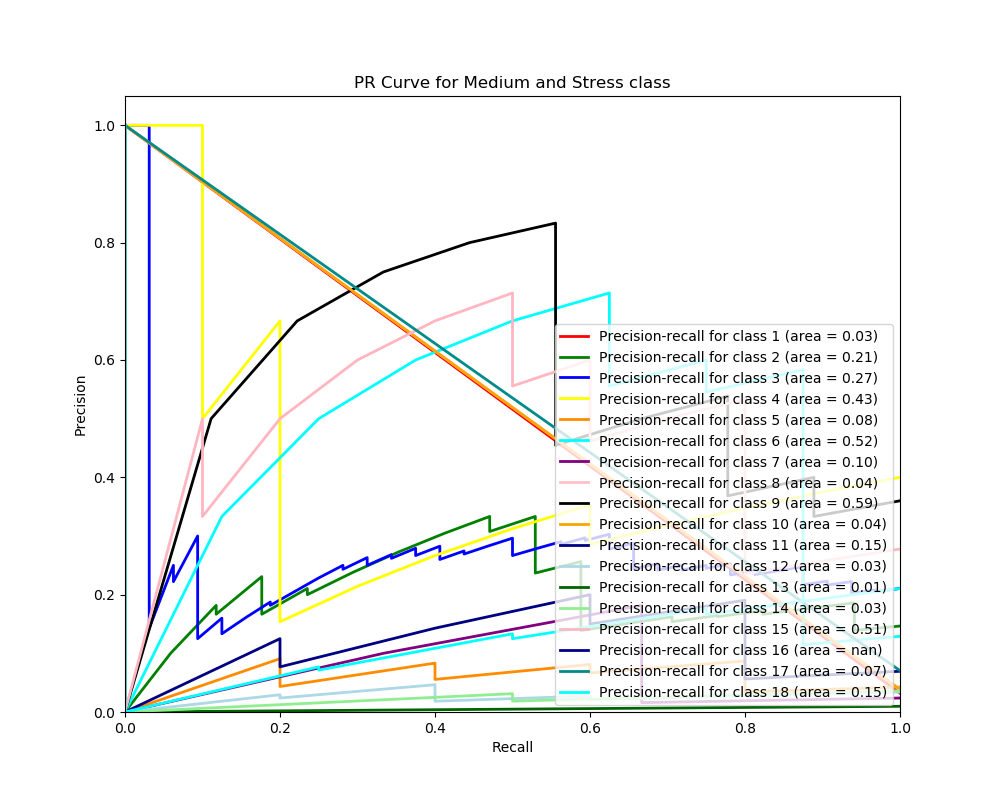


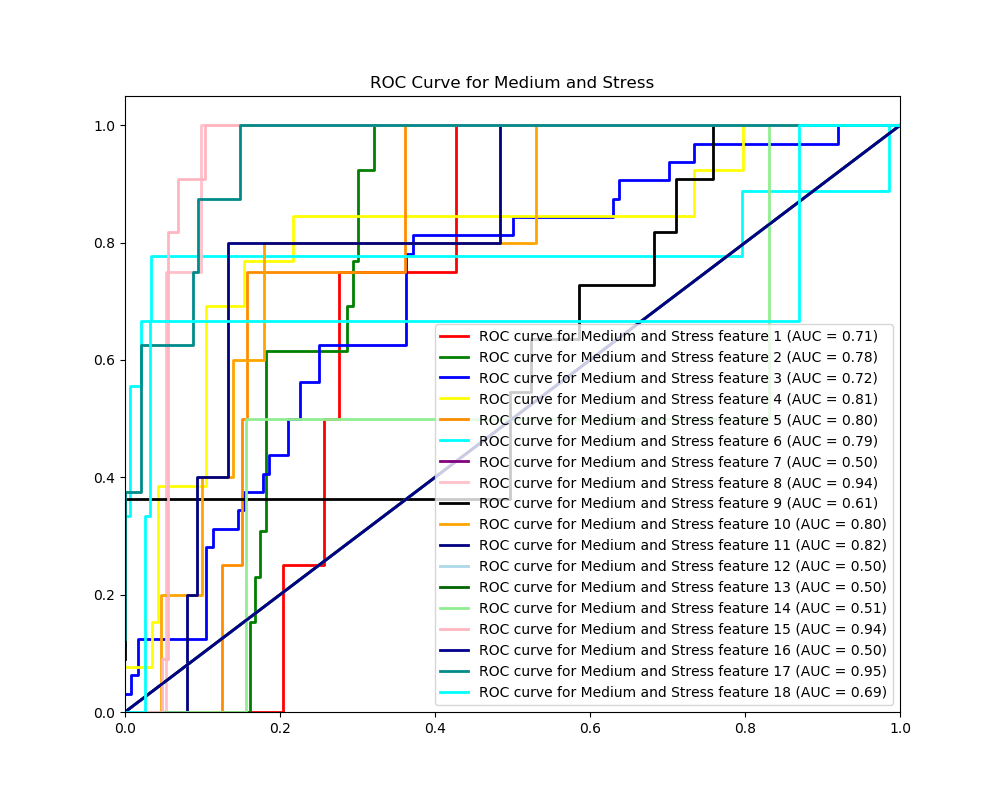


Problem 5

Before process the data, I merge those two classes together and obtain the data. Based on the results, it is more efficient and useful to use single classifier in my case since the combined one fail to recognizes some classes as their features are ignored in the process of feature selection.

Best Combined Classifier AUC: 0.67 AUPRC for best AUC: 0.0





Problem 6

This problem can be done in a similar fashion as problem 4. The only difference is that we are using PCA with value of three instead of the LinearSVC method to do the dimensionality reduction. PCs retain most of the classification performance as before.

Best AUC value: 1 AUPRC for best AUC value: 0.9951

