Name: Sairamvinay Vijayaraghavan
Student ID: 913603345

ECS 171 HW 3

1) I had used Lasso regression in order to perform regularized regression. The reason why I chose this was because it decreases the number of irrelevant features within the data and hence narrows down the number of features to regress against. Lasso regression performs L1 regularization which is used to minimize the number of features to regress against in the regular linear regression problem. The objective function to minimize for Lasso regression is as follows:

$$L = \sum( \hat{Y}i - Yi)2 + \lambda\sum |\beta|$$

Here, L is the loss function to minimize which is now equivalent to the sum of the RSS values for each sample and the beta **β** is the weight for a particular feature and lambda is the regularization constant which is used to control the features. The optimal weight **β** value is found as we minimize the loss function and hence if the weight for any feature ii is zero, then we don't need to use feature i since it is now reduced. This technique helps in dimensionality reduction and hence achieves our aim to cut down on the number of features to predict on. There is always a variance-bias tradeoff in picking the optimal regularization constant lambda. If the lambda value is high, then many features (coefficients or weights) will be set to zero and hence will be eliminated which would increase the bias to certain features. Meanwhile, small lambda values would preserve more features and hence it would lead to more variance. Hence an optimal value for the lambda has to be selected from a wide range of small and large lambdas.

For this problem, I had applied grid search on a range of lambda (named "alpha" inside sklearn package) values inside the lasso regression method. I chose lambda values as [1.e-06, 1.e-05, 1.e-04, 1.e-03, 1.e-02, 1.e-01, 1.e+00, 1.e+01,1.e+02] for finding the optimal lambda value. The optimal lambda value is the one which minimizes the mean square error obtained.

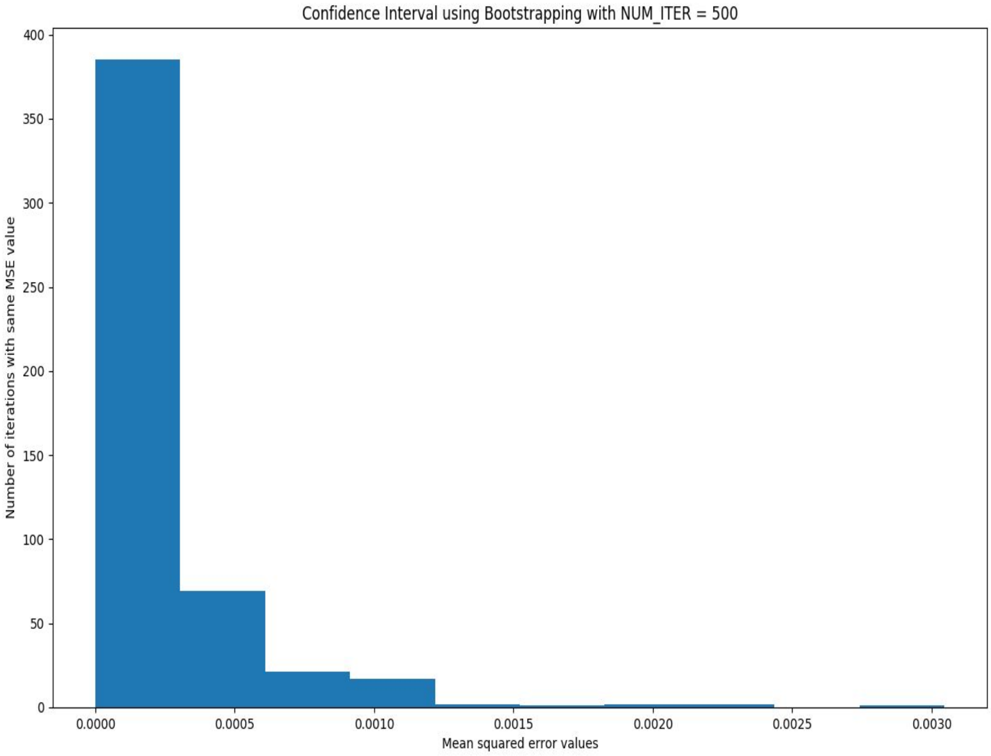
The optimal lambda constant was **0.0001 (1.e-04)** and the 5 fold cross-validation generalization error (the mean squared error) was **0.031665** and it preserved **252** features (or 0 non features) as the non-zero coefficients.

2) I used a simple bootstrapping method to find out the confidence interval for the predicted "GrowthRate" in the dataset. I had run a large number of iterations (500) of this method where in every iteration, I use a training set which is a random sample of 100 samples from the data set (which is ideally different in every iteration since I did not set the random state) picked from the data set and the Lasso model was trained

on it to predict the mean growth rate using the mean expression value. I had then collected all the mean squared error values between the true mean growth rate and the predicted mean growth rate in every iteration. The key assumption to this problem is that I assumed that the mean expression value can be used to obtain a prediction of the mean growth rate.

3) The confidence interval of **95**% was obtained for the predicted growth for a bacterium whose genes are expressed exactly at the mean expression value in between the error rates **0.00000 and 0.00111**. These were the **mean squared errors** obtained at every iteration.

Graph for the bootstrapping distribution is as follows:

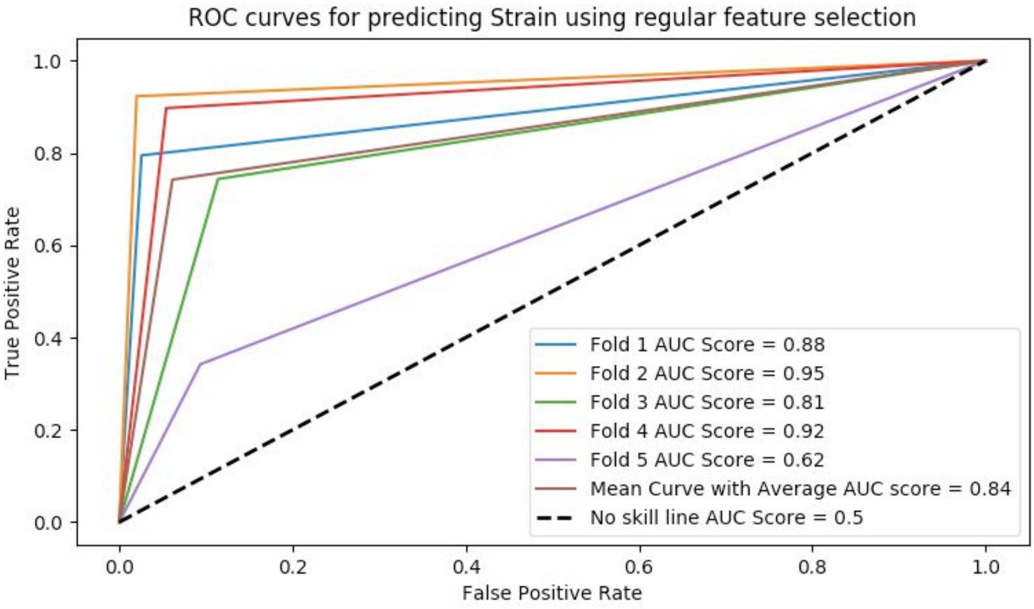


4) I used a wrapper method (SelectFromModel) to help me identify the relevant features to use while predicting.

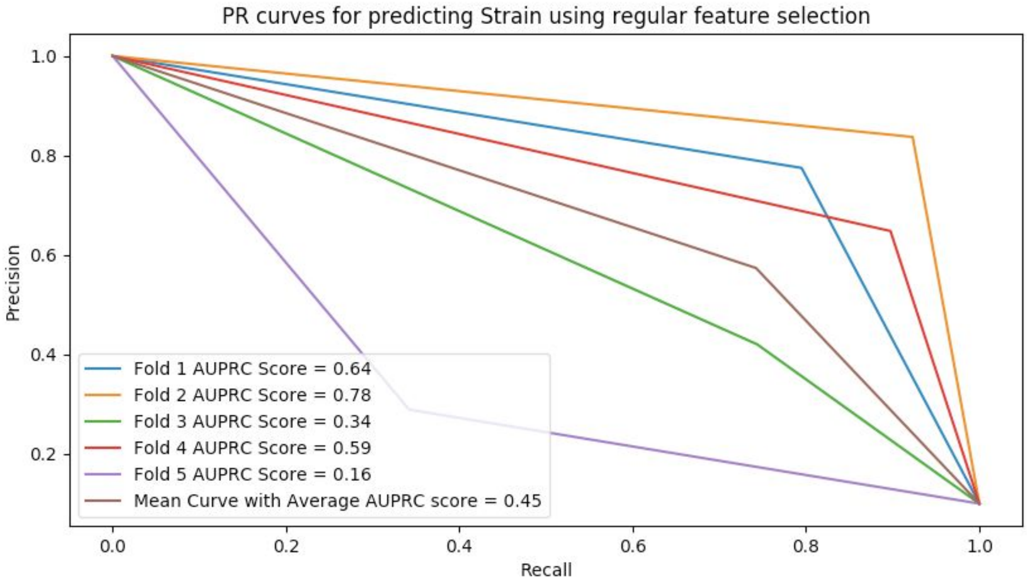
**NOTE**: NO_SKILL_LINE always has an area of 0.5 in ROC.

COLUMN TYPE PREDICTED: **STRAIN**
NUM_FEATURES USED: **226**

**ROC PLOT:**



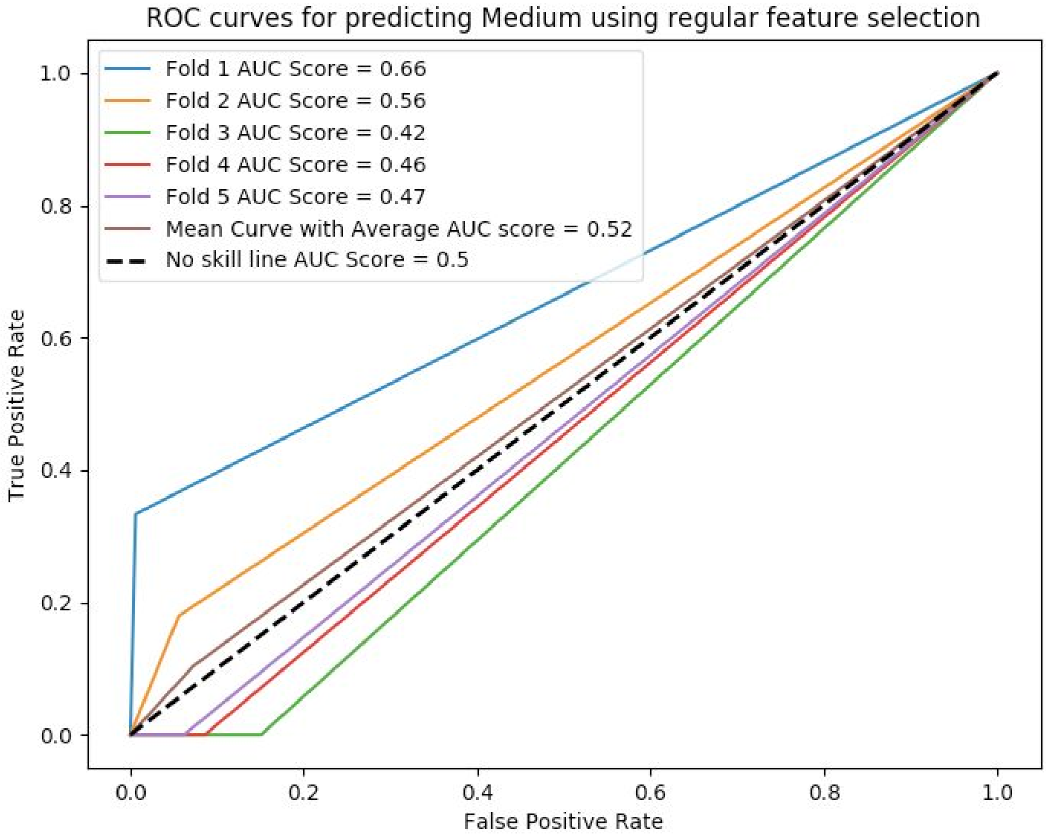
**PR PLOT:**



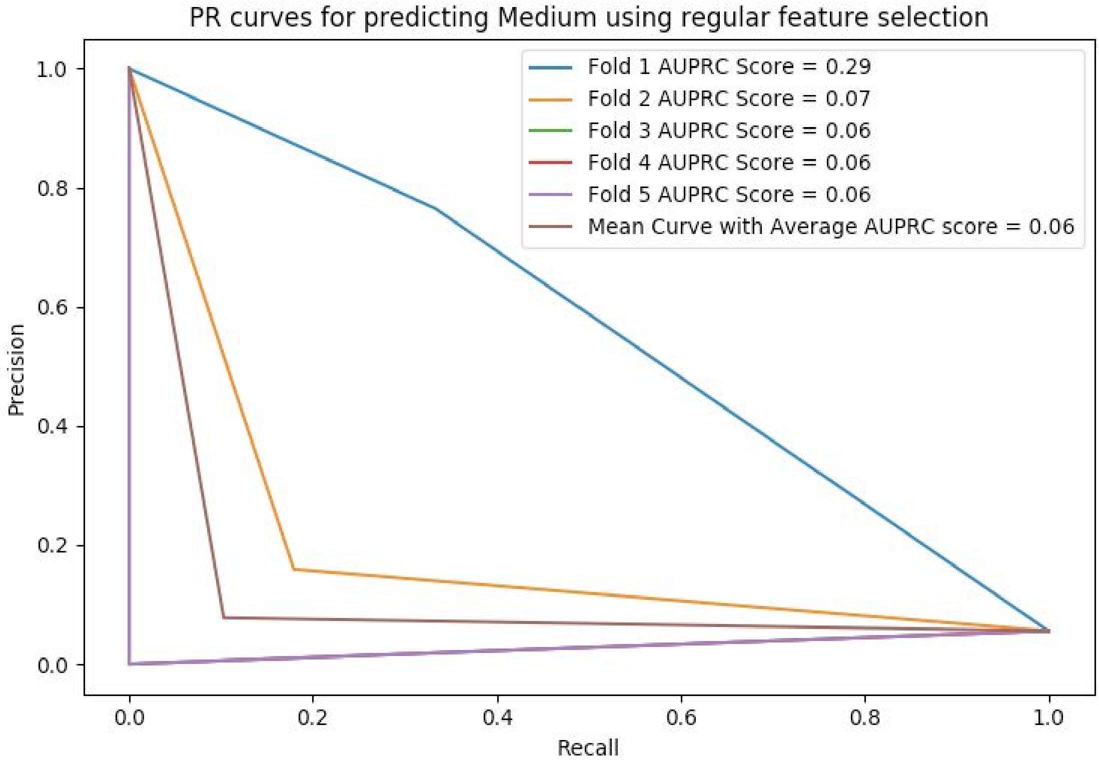
COLUMN TYPE PREDICTED: **MEDIUM**

NUM_FEATURES USED: **41**

**ROC PLOT:**

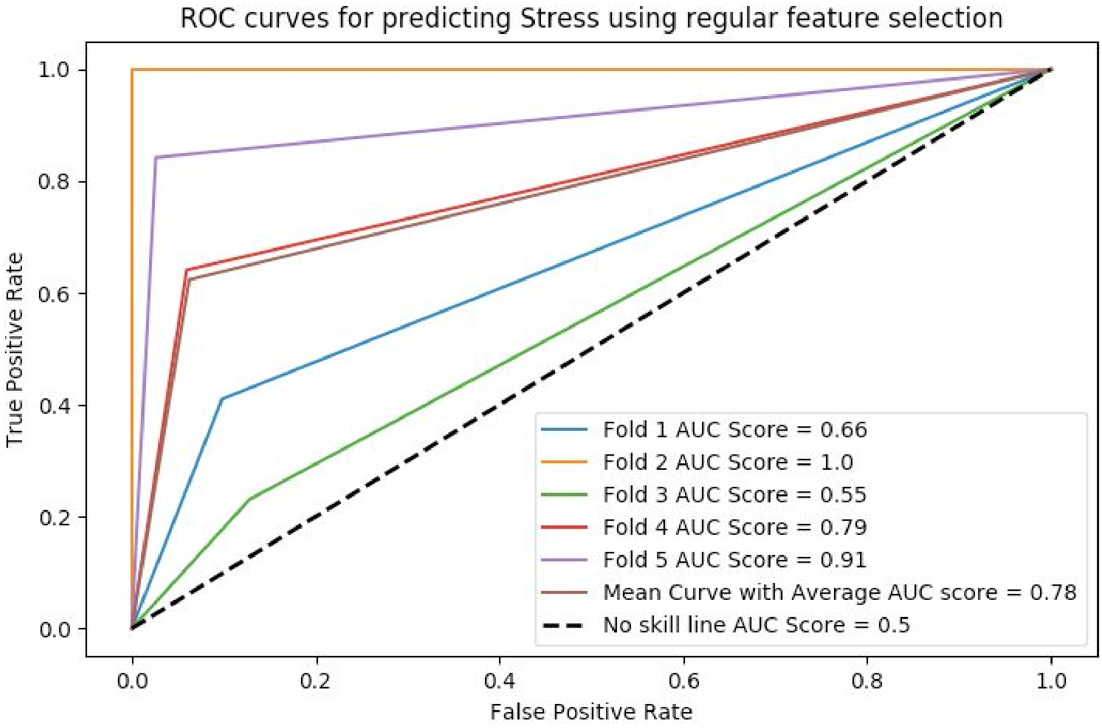


**PR PLOT:**

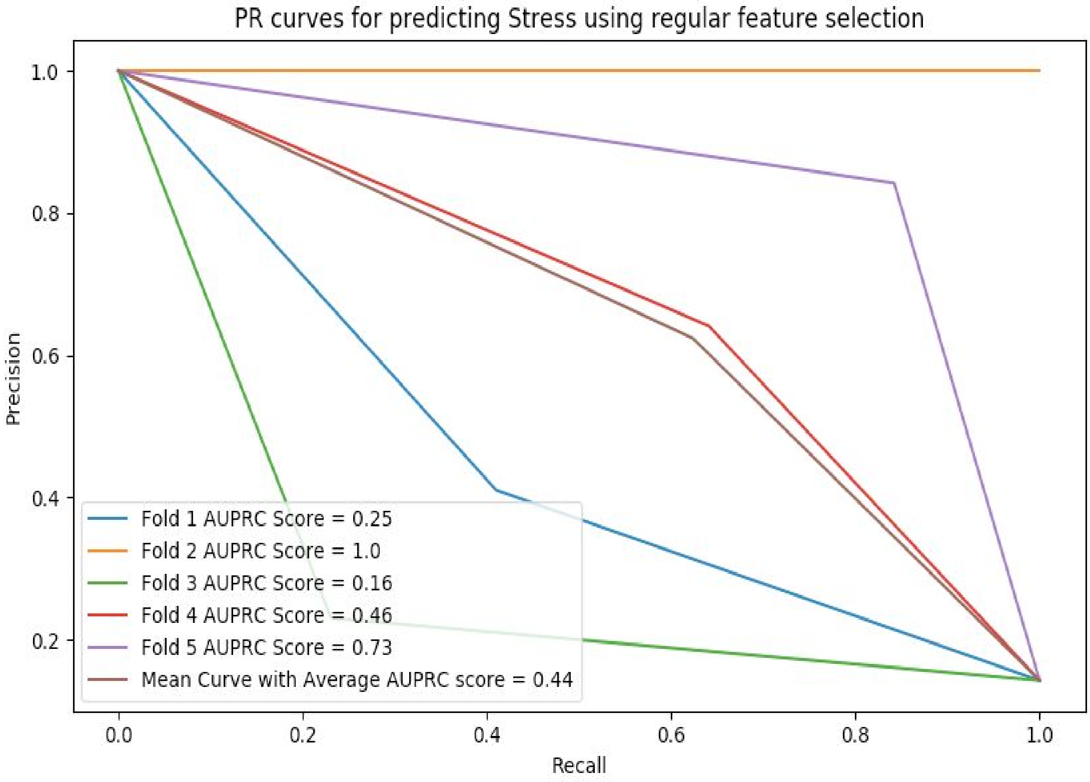


COLUMN TYPE PREDICTED: **STRESS**
NUM_FEATURES USED: **2**

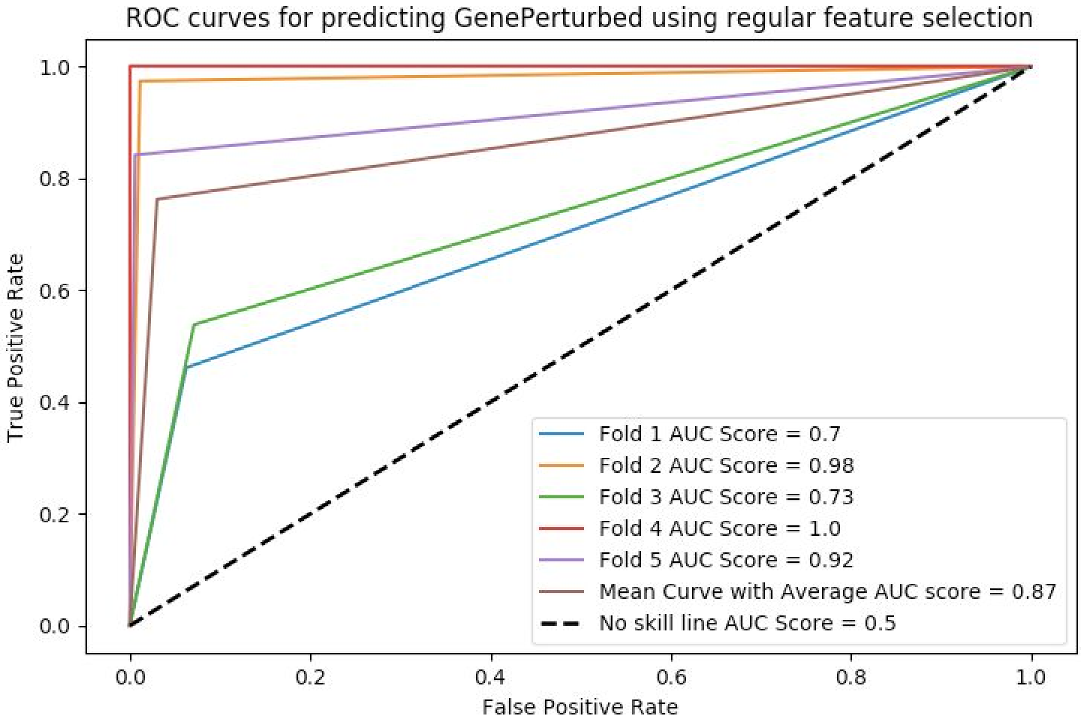
**ROC PLOT:**



**PR PLOT:**



COLUMN TYPE PREDICTED: **GENE PERTURBED**
NUM_FEATURES USED: **13**



**PR PLOT:**

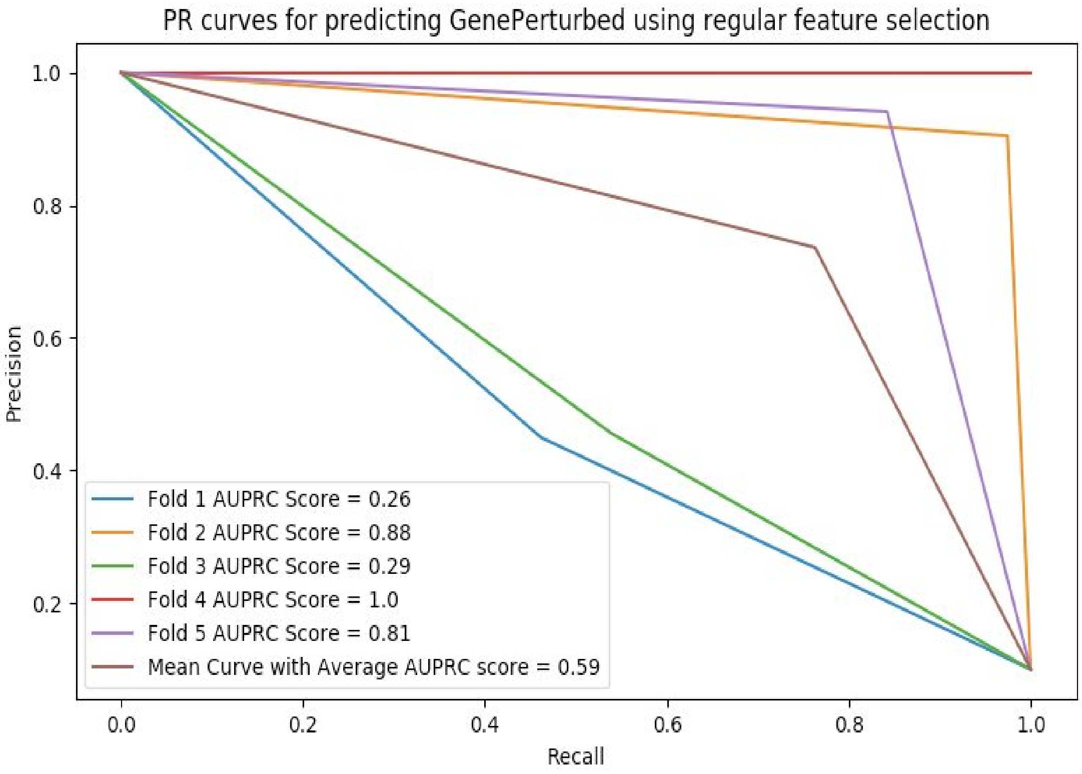
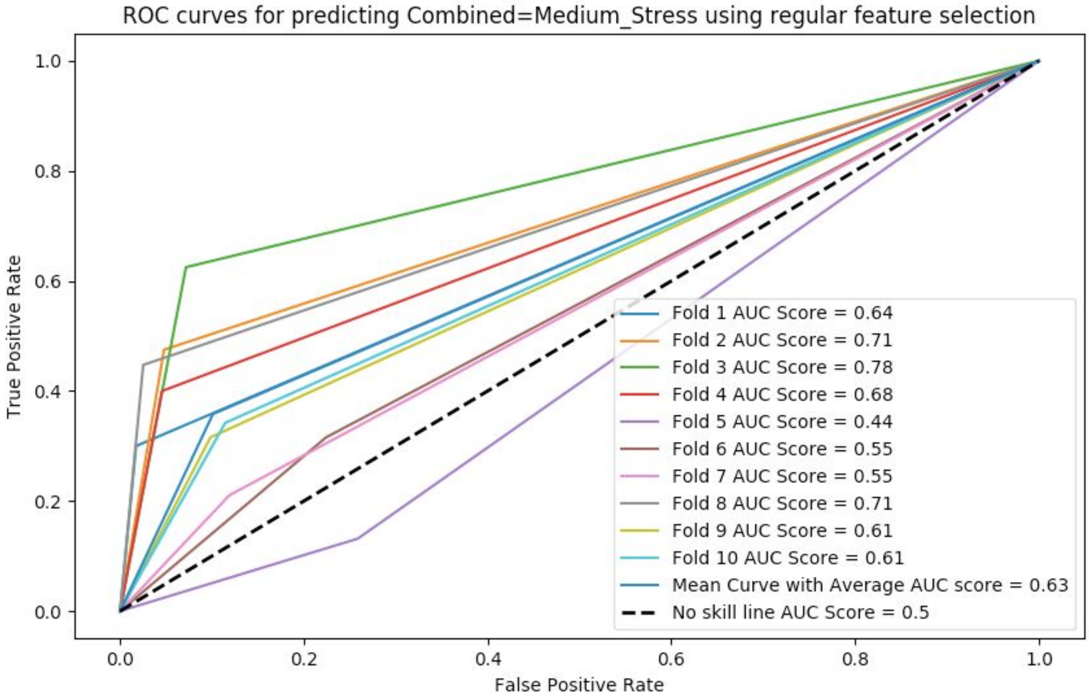


Table: This problem uses only regular feature selection (SelectFromModel wrapper) These scores are the average AUC for ROC (column 1) and PR (column 2) curves after 5 fold Cross-Validation

| OUTPUT / METRIC | MEAN AUC UNDER ROC | MEAN AUPRC UNDER PR | NUMBER OF FEATURES USED |
|---|---|---|---|
| STRAIN | 0.84 | 0.45 | 226 |
| MEDIUM | 0.52 | 0.06 | 41 |
| STRESS | 0.78 | 0.44 | 2 |
| GENE-PERTURBED | 0.87 | 0.59 | 13 |

5) The composite classifier yielded results as follows:

**ROC PLOT:**



**PR PLOT:**

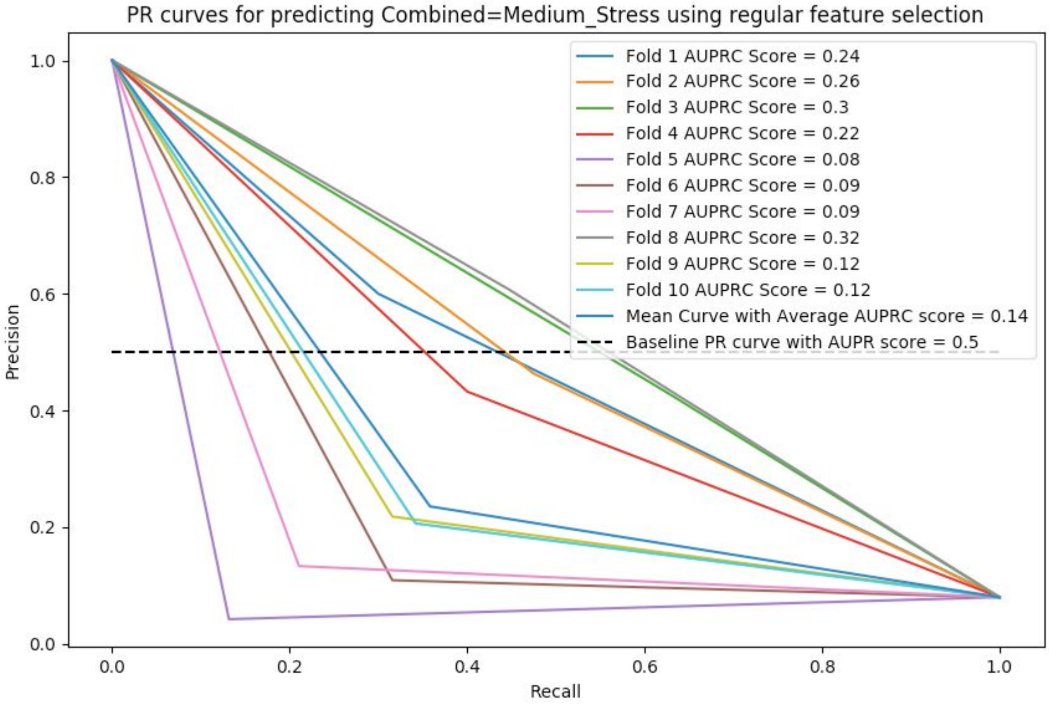


TABLE for the AUC scores for ROC and PR curves for combined Medium and Stress outputs using 10 fold Cross-Validation:

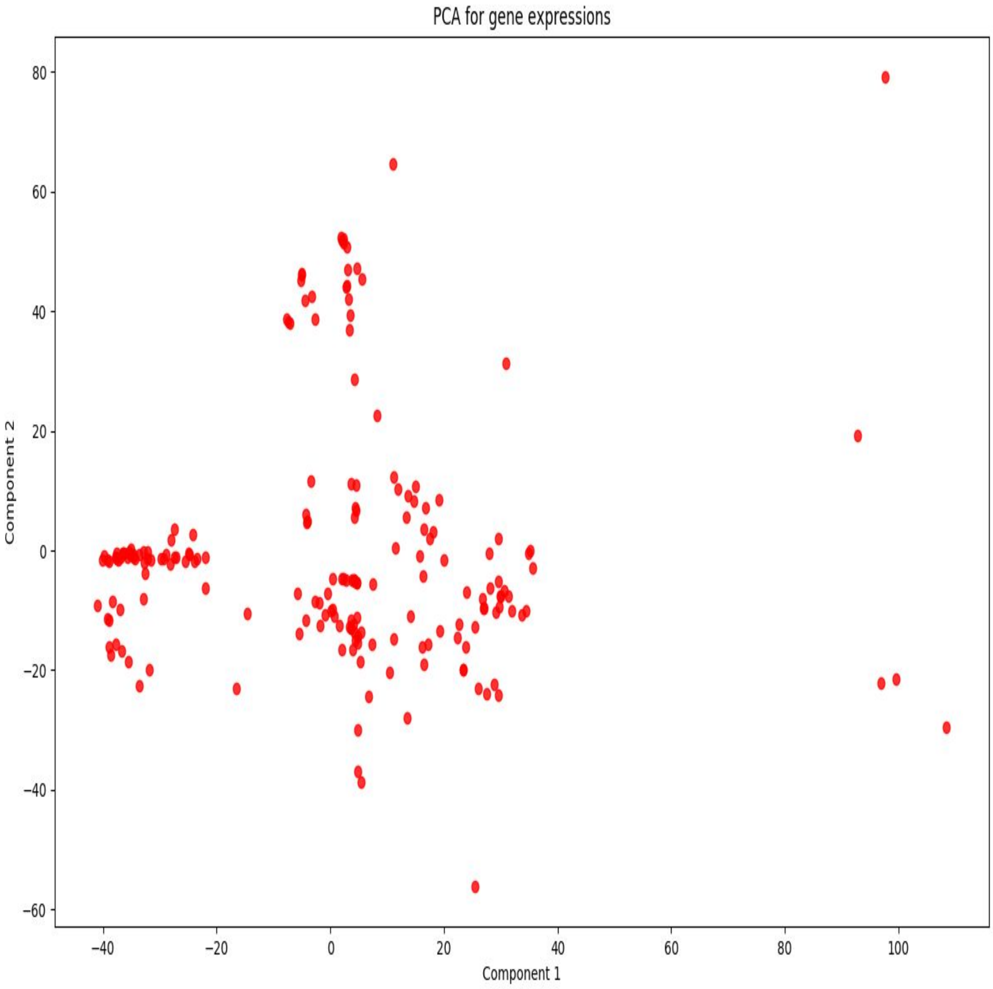
| OUTPUT | AUC score for ROC curve | AUPRC score for PR curve |
|---|---|---|
| Medium and Stress together | 0.63 | 0.14 |

The combined classifier does better than Medium classifier while it does poor compared to the Stress classifier. While looking at the PR curves, a similar trend is observed. The ROC curve does better than baseline for composite while it does worse than baseline for PR curve. The baseline model has AUC score = 0.5 which is same prediction for each sample.

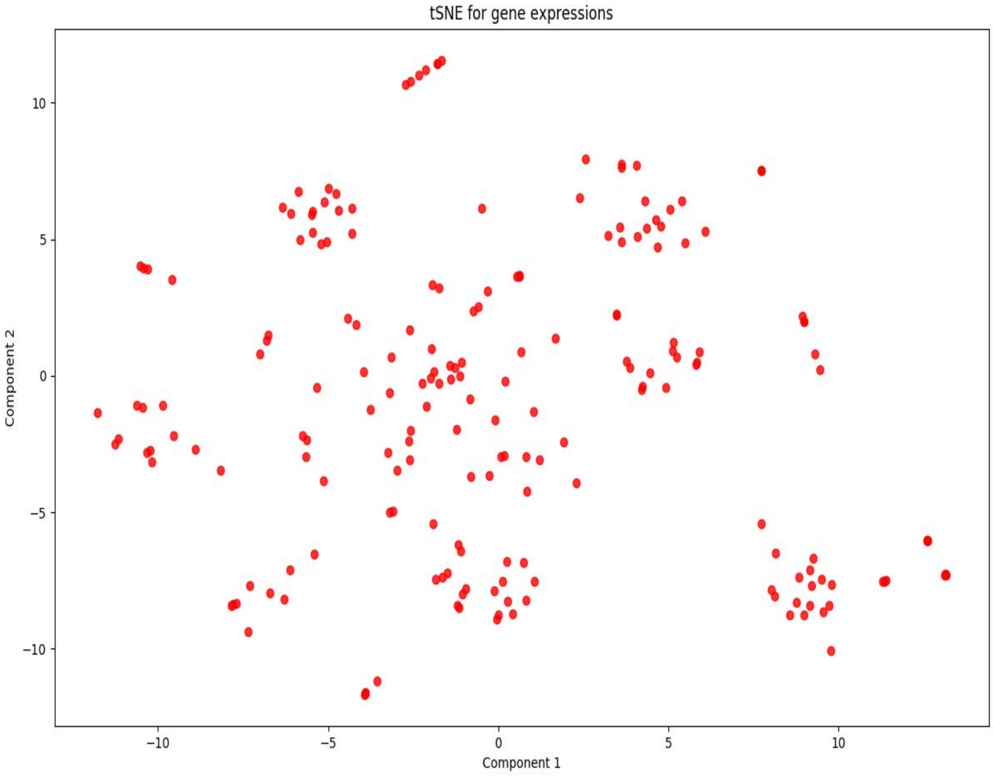
6) PCA and tSNE yielded the following plots:

NOTE: the input had been scaled in order to get a proper visualization of the distribution.

PCA PLOT:


PCA for gene expressions
Component 2
Component 1
80
60
40
20
0
−20
−40
−60
−40
−20
0
20
40
60
80
100

tSNE plot:



7)

Table shows the average AUC and AUPRC scores for each of the outputs (from question 4) for dimensionality reduction using PCA technique after 10 fold Cross-Validation .

PCA:

| OUTPUT / METRIC | MEAN AUC UNDER ROC | MEAN AUPRC UNDER PR | NUMBER OF FEATURES USED |
|---|---|---|---|
| STRAIN | 0.80 | 0.49 | 2 |
| MEDIUM | 0.50 | 0.06 | 2 |
| STRESS | 0.78 | 0.43 | 2 |
| GENE-PERTURBED | 0.89 | 0.67 | 2 |

Table shows the average AUC and AUPRC scores for each of the outputs (from question 4) for dimensionality reduction using tSNE technique after 10 fold Cross-Validation.

tSNE:

| OUTPUT / METRIC | MEAN AUC UNDER ROC | MEAN AUPRC UNDER PR | NUMBER OF FEATURES USED |
|---|---|---|---|
| STRAIN | 0.85 | 0.55 | 2 |
| MEDIUM | 0.50 | 0.06 | 2 |
| STRESS | 0.78 | 0.44 | 2 |
| GENE-PERTURBED | 0.89 | 0.66 | 2 |

The best approach in general would be the algorithm which provides the highest AUC scores in general. It can be noticed that the highest AUC score for each classifier as follows:

STRAIN: best is **TSNE**
MEDIUM: best is **regular feature selection (SelectFromModel)**
STRESS: **all are very similar to each other**
GENE-PERTURBED: **PCA and TSNE** do equally better than regular feature selection