ECS 171 Homework 3 Report

Yilun Yang
912425744

Problem1 contain: Problem1.m, lassofit.m
Problem2 contains: bootstrapping .m
Problem3 contains: Problem3.m
Problem4 contains: Problem4.m, SVM_binary.m, SVM_multiclass_pkg.m, ROCPRCcurve.m
Problem5 contains: Problem5.m, SVM_composite_pkg.m, ROCPRCcurve.m.
Problem6 contains: Problem6.m, ROCPRCcurve.m.

1. *Create a predictor of the bacterial growth attribute by using only the expression of the genes as attributes. Not all genes are informative for this task, so use a regularized regression technique (lasso, elastic net, ridge) and explain what it does. Which one is the optimal constrained parameter value (usually denoted by λ)? Report the number of features that have non-zero coefficients and the 10-fold cross-validation generalization error of the technique.*

   In the first problem, I choose lasso regression because it can select informative genes for us, that is, lasso may produce some zero estimated coefficients and allows variable deletion. Like ridge and many other regularization techniques, lasso is used for model selection, in particular to prevent overfitting by penalizing models with extreme parameter values.

   In the first problem, I used lasso function in matlab and I also implemented my own version of lasso with proximal gradient descent and cross-validation.

   (1) Lasso function:

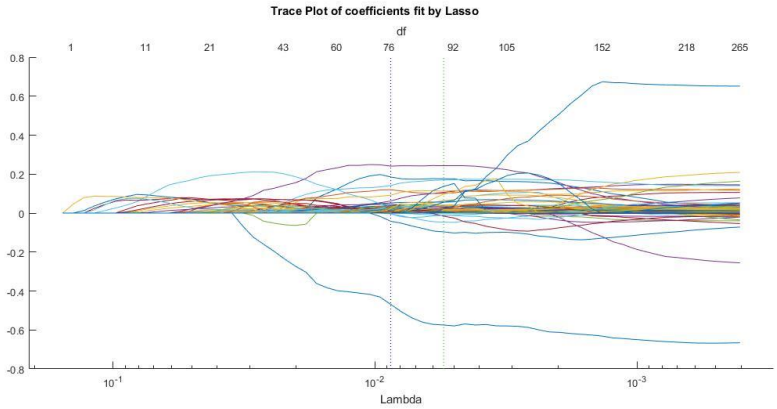
   $$(1/n)\sum(Y_i - \beta_1 X_{i1} - \cdots - \beta_p X_{ip})^2 + 2\lambda \sum |\beta_j|$$

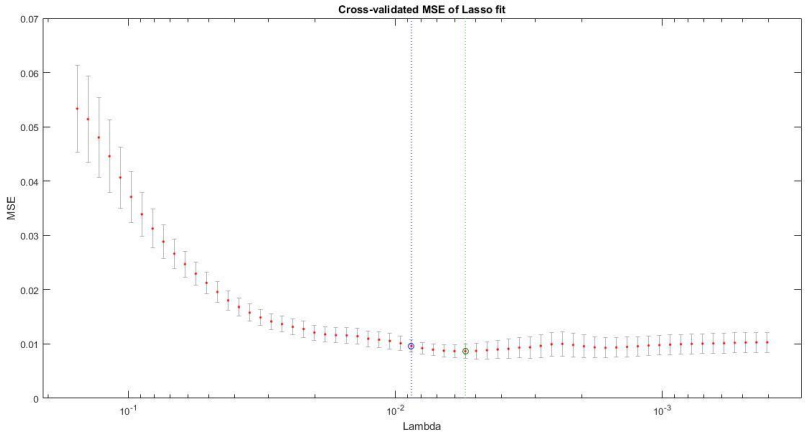
   Lasso function minimize the above function and we got the optimal lambda as 0.0087 and there are 76 non-zero coefficients, 10 fold CV mse is 0.0096. Coefficient trace plot and CV mse plot are also given below.

   (2) My own version lassofit:
   On the contrary to default lasso function, I minimized a different function:

   $$||Y - X\beta||^2 + k||\beta||_1 = \sum(Y_i - \beta_1 X_{i1} - \cdots \beta_p X_{ip})^2 + k\sum|\beta_j|,$$

   Thus my optimal lambda should be similar to 2n * previous lambda(n is number of sample), which is 3.4. I actually got optimal lambda as 5, 102 non-zero coefficients and 10 fold CV mse as 0.0321. This is pretty close to the toolbox function.





2. *Extend your predictor to report the confidence interval of the prediction by using the bootstrapping method. Clearly state the methodology and your assumptions.*

I resampled the dataset 100 times and built 100 models for those datasets. For each model I can get a prediction Y. With these 100 predicted Y, I can calculate mean and standard deviation of them and then get the confidence interval for Y. Specifically,

$$E(\hat{Y}) = \frac{\hat{Y}_1 + \ldots + \hat{Y}_{100}}{100} \quad \text{and} \quad sd(\hat{Y}) = \sqrt{\frac{\sum_{i=1}^{100} (\hat{Y}_i - \overline{\hat{Y}_i})^2}{100}}$$

Also, according to law of large number, we can approximately treat distribution of Y as normal distribution and thus a 95% CI of Y is:

$[\overline{\hat{Y}_i} - 1.96sd(\hat{Y}), \overline{\hat{Y}_i} + 1.96sd(\hat{Y})]$.

In my program, I put varargin as an argument, if testdata is provided, we predict the testdata's confidence interval, otherwise, we calculate training data's confidence interval. Note testdata can be a matrix, thus we can estimate many samples at one

time.

3. *What is the predicted growth for a bacterium whose genes are expressed exactly at the mean expression value? That is, for a particular gene, its expression is equal to the gene's mean expression across all the samples.*

   Because we firstly centered the response variable Y which is growth rate and features X to build lasso model (This is necessary for lasso model building), the predicted growth should be $Y = \bar{Y} + (\bar{X} - \bar{X})\beta = \bar{Y} = 0.3936$.

4. *Create four separate SVM classifiers to categorize the strain type, medium type, environmental and gene perturbation, given all the gene transcriptional profiles. For each classifier (4 total) report the number of features and the classification performance though 10-fold cross-validation by plotting the ROC and PR curves and reporting the AUC/AUPRC values.*

   Firstly, I implemented my own version of binary SVM with sequential minimal optimization. We provide box constraint, base class (+1 class), three different kernels (linear, 2nd order polynomial and RBF) and optional testdata. We can get coefficients and intercept for only linear kernel because it's hard to calculate and interpret the coefficients and intercept for other non-linear kernels. For all the kernels, we can get prediction class and score for testdata if provided, otherwise, prediction of training data will be given.

   Compared with fitcsvm function in Matlab, my own version can get the same result with same precision (if iteration is enough big, generally near 1000000). However, speed of my SVM is much slower. Thus, for the following problems, I will use fitcsvm and its complemented functions in the toolbox.

   I implemented four classifiers to categorizing strain, medium, stress and gene perturbation. Since each variable has so many classes (usually more than 10), I decided to use one versus all method to deal with the multi-class problem.

   In detail, we firstly build binary SVM models as many as the number of class in the response variable. In each binary model, we use 10 fold cross validation to get the prediction score for each sample. A positive score means the sample should be classified in this class, otherwise, shouldn't. A score far from 0 means we are more confident of the classification. So, after 10 times of binary classification, we may get a score matrix with each column as a sample and each row as the scores for a specific class. As we discussed, if the score is bigger, we are more confident of the classification. We should find which class has the biggest score for each sample and then classify the sample into that class. Comparing the result with true y class, we can find the classification error as well.

For this particular problem, if we want to draw ROC and PRC curves, we should have the score matrix we got previously, and the true class number (say, we first unique y in the program, if a sample's y is the 3rd class in unique(y), then it's true class number is 3) and also thresholds. We first have to standardize the score matrix due to the great dispersion of scores and then map scores into values with in [0, 1]. With different thresholds, we can have different predicted score matrix then we can calculate TP, TN, FP, FN, Recall, Precision as we need. Finally, we can draw ROC and PRC with the values we calculated. For AUC and AUPRC, we use small trapezoid areas to approximate the true AUC and AUPRC.

Note, the TP, TN, etc. we calculated are not totally the same with the traditional concepts which are only applicable to binary SVM. We here redefined them to make them useful in multi-class problems. TP here means correctly classified, TN mean correctly not classified, FP means incorrectly classified, FN means incorrectly not classified.

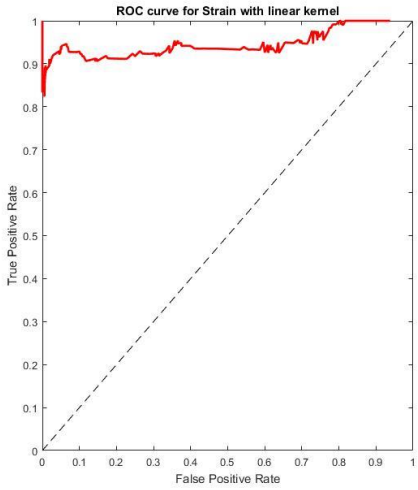
For Strain, I built three models with linear, polynomial and RBF kernel. The number of feature is 76 which is consistent with what we got from lasso in problem1. The 10 fold cross validation misclassification rate are: 0.1753, 0.1959, 0.2062. AUC values are 0.8850, 0.8495, 0.8508. AUPRC are 0.1538, 0.5249, 0.1394.

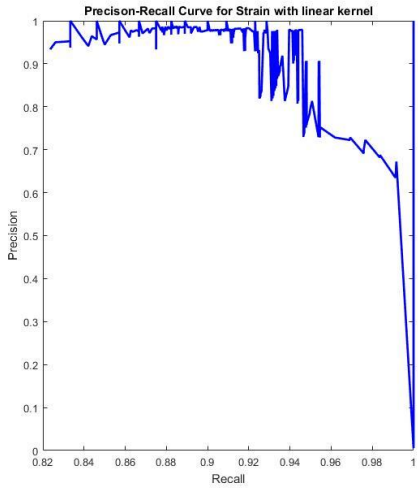
For Medium, I built three models with linear, polynomial and RBF kernel. The number of feature is 76 which is consistent with what we got from lasso in problem1. The 10 fold cross validation misclassification rate are: 0.2474, 0.3402, 0.4124. AUC values are 0.8669, 0.7381, 0.7744. AUPRC are 0.6801, 0.6947, 0.2201.

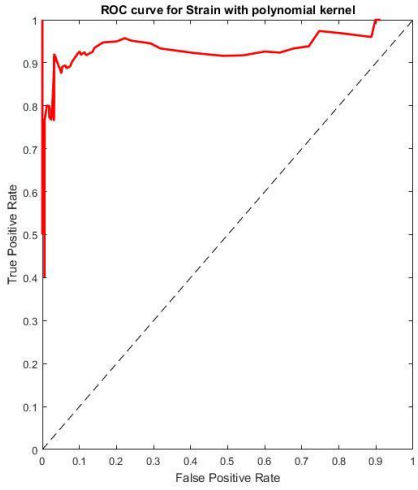
For Stress, I built three models with linear, polynomial and RBF kernel. The number of feature is 76 which is consistent with what we got from lasso in problem1. The 10 fold cross validation misclassification rate are: 0.2062, 0.2010, 0.2629. AUC values are 0.8441, 0.8386, 0.8321. AUPRC are 0.5072, 0.8249, 0.2212.

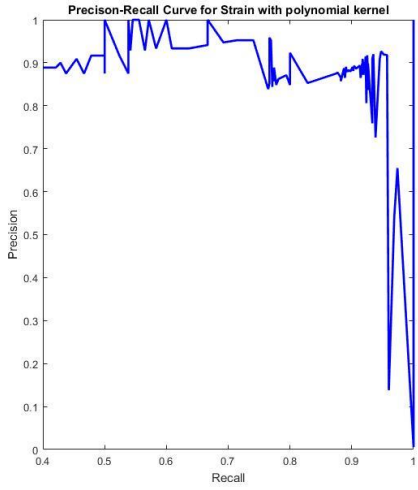
For Perturbation, I built three models with linear, polynomial and RBF kernel. The number of feature is 76 which is consistent with what we got from lasso in problem1. The 10 fold cross validation misclassification rate are: 0.2113, 0.2062, 0.1649. AUC values are 0.8370, 0.7834, 0.9040. AUPRC are 0.1152, 0.3286, 0.4742.

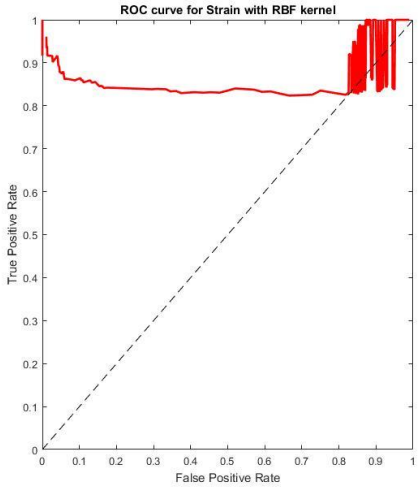
Some PRC curves' x axis don't begin from 0 thus the AUPRC is very small. That's really weird and I assume it is because there are many NaN points before the beginning point that can't be drew. Thus I primarily use AUC to evaluate the goodness of models. From the above analysis, we may conclude linear kernel is a good choice for all the cases. This make sense, since we only have 194 samples and there are 76 features, we would assume a very sparse space of our SVM model, thus a linear kernel is already enough for separation.

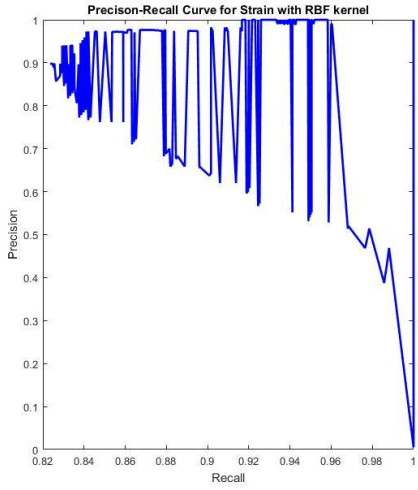

ROC curve for Strain with linear kernel
True Positive Rate
False Positive Rate

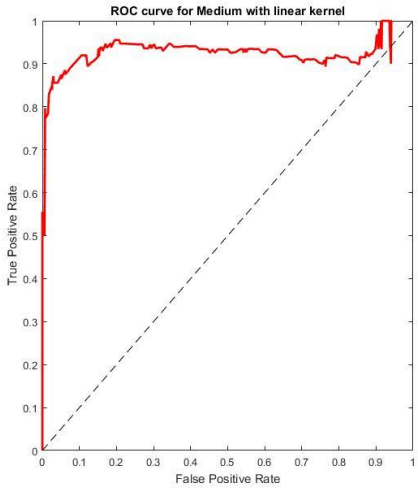

Precison-Recall Curve for Strain with linear kernel
Precision
Recall

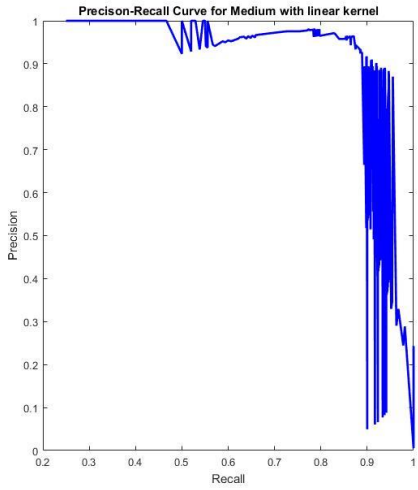

ROC curve for Strain with polynomial kernel
True Positive Rate
False Positive Rate

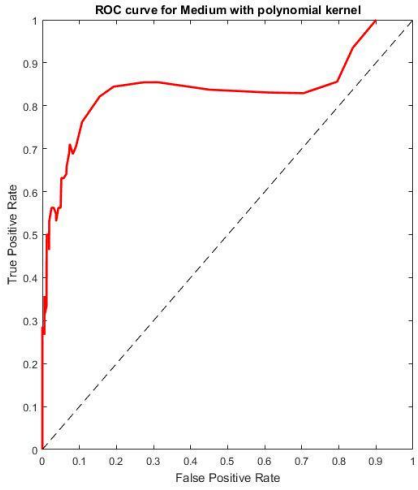

Precison-Recall Curve for Strain with polynomial kernel
Precision
Recall

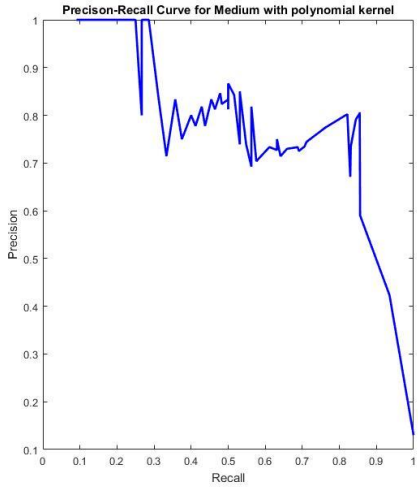

ROC curve for Strain with RBF kernel
True Positive Rate
False Positive Rate

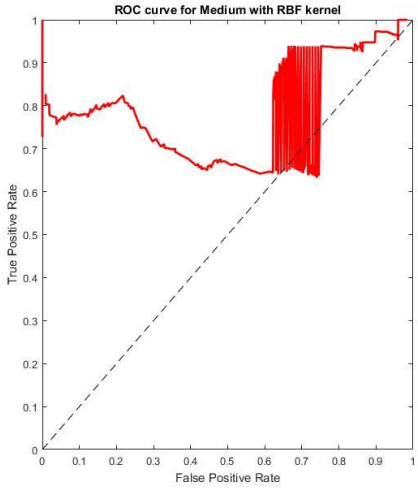

Precison-Recall Curve for Strain with RBF kernel
Precision
Recall

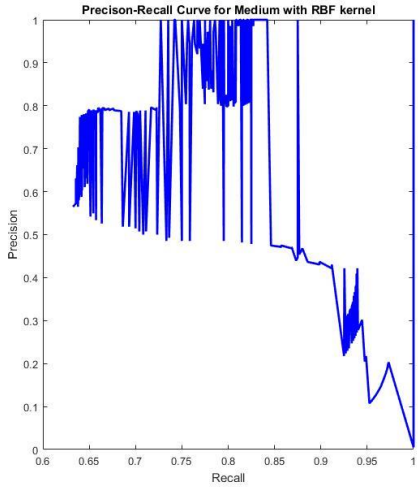

ROC curve for Medium with linear kernel
True Positive Rate
False Positive Rate

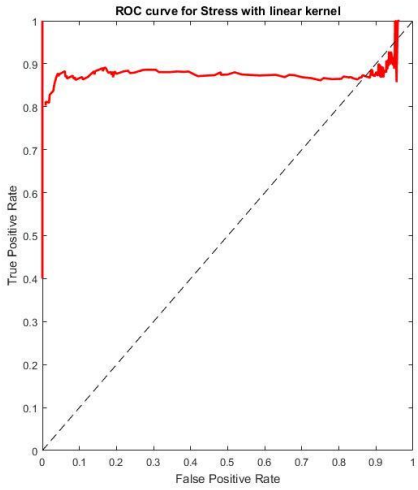

Precison-Recall Curve for Medium with linear kernel
Precision
Recall

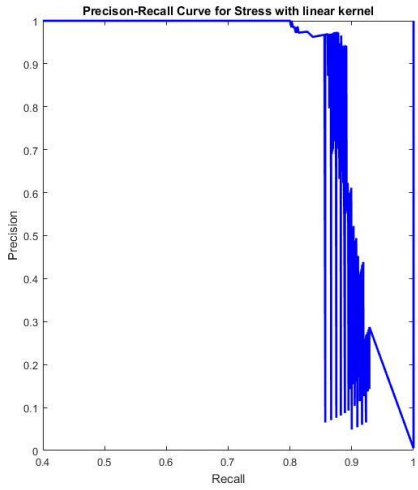

ROC curve for Medium with polynomial kernel
True Positive Rate
False Positive Rate

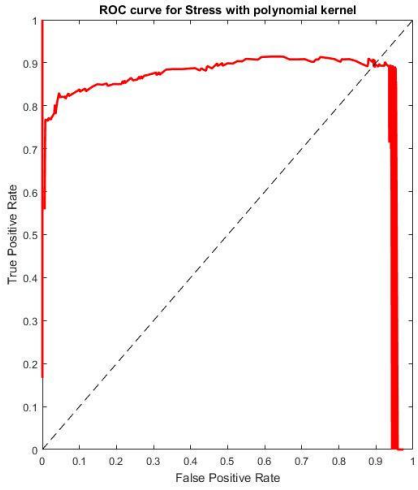

Precison-Recall Curve for Medium with polynomial kernel
Precision
Recall

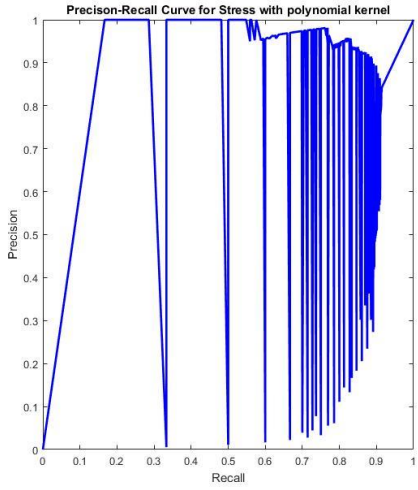

ROC curve for Medium with RBF kernel
True Positive Rate
False Positive Rate

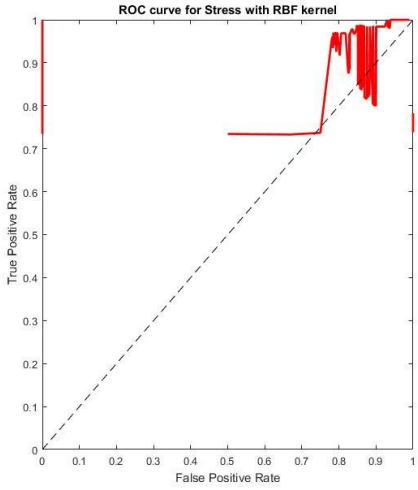

Precison-Recall Curve for Medium with RBF kernel
Precision
Recall

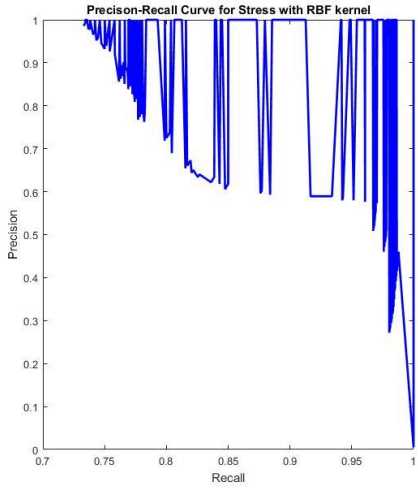

ROC curve for Stress with linear kernel
True Positive Rate
False Positive Rate

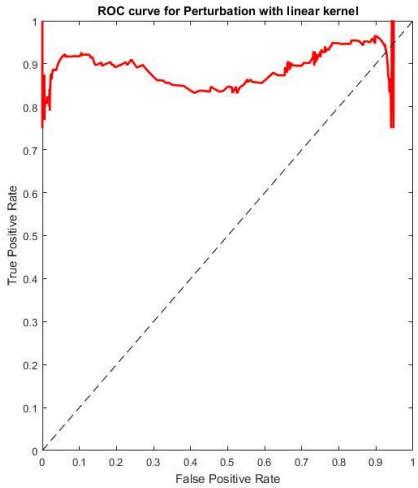

Precison-Recall Curve for Stress with linear kernel
Precision
Recall

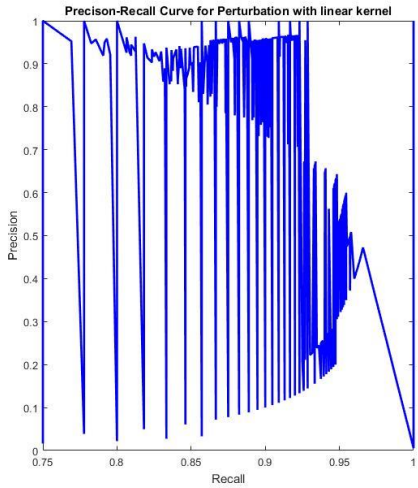

ROC curve for Stress with polynomial kernel
True Positive Rate
False Positive Rate

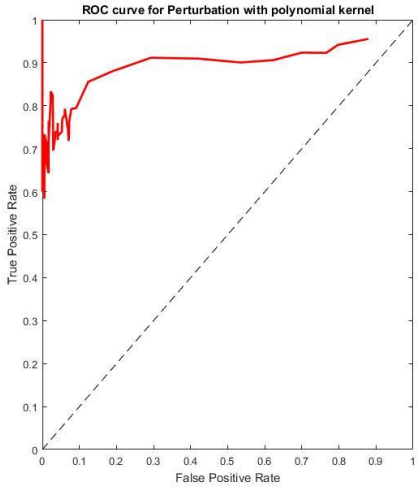

Precison-Recall Curve for Stress with polynomial kernel
Precision
Recall

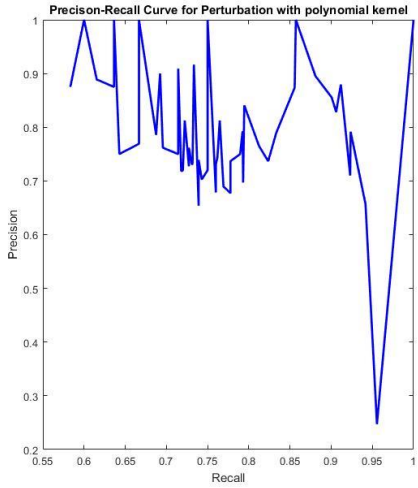

ROC curve for Stress with RBF kernel
True Positive Rate
False Positive Rate

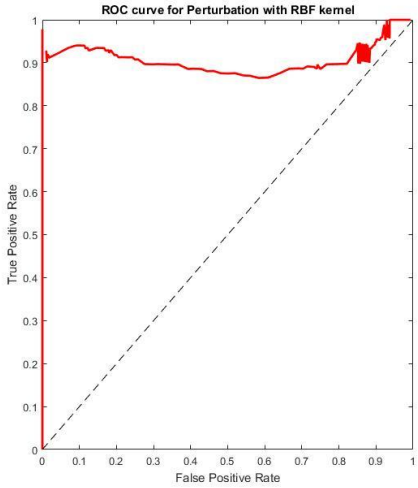

Precison-Recall Curve for Stress with RBF kernel
Precision
Recall

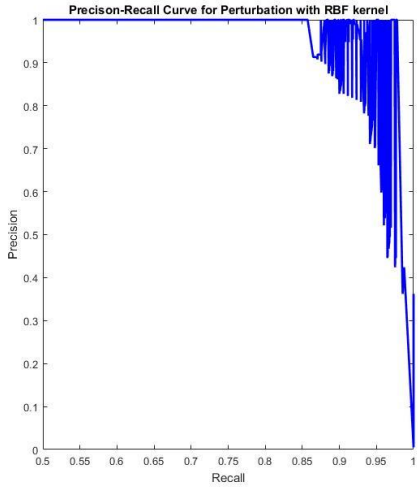












5. *Create one composite SVM classifier to simultaneously predict medium and environmental perturbations and report the 10-fold cross-validation AUC/AUPRC value. Does this classifier perform better or worse that the two individual classifiers together for these predictions? That is, are we better off building one composite or*

two separate classifiers to simultaneously predict these two features? What is the baseline prediction performance (null hypothesis)?

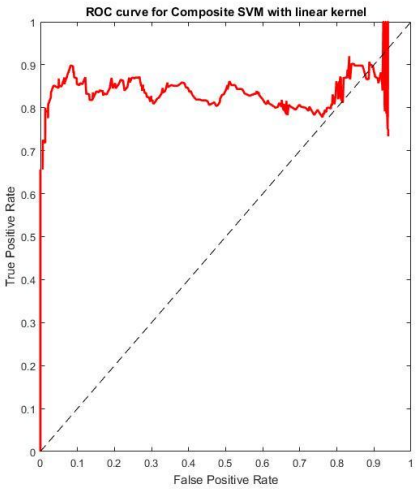
Since linear kernel is the best choice, we would only apply linear kernel in this problem for comparison. Composite model is very similar to the multi-class model except we have to combine variable classes together at first.

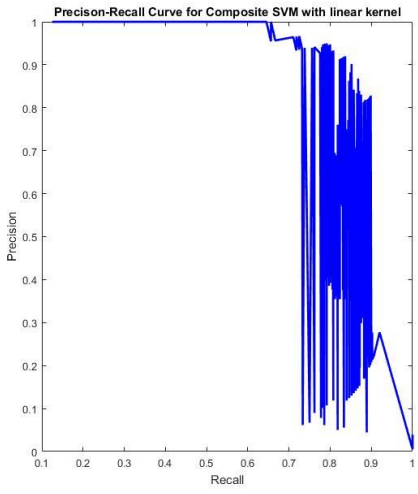
For separate model, the ROC and PRC plots are the same with the corresponding part in problem4 and so do AUC and AUPRC. AUC values are 0.8669, 0.8370 for medium and perturbation. AUPRC are 0.2201 and 0.4742. In the separate model, only the samples that be correctly classified by both Medium and Perturbation can be defined as correct classified. Thus the misclassification rate is 0.4588.

For composite model, AUC values is 0.8235, AUPRC is 0.7207. The misclassification rate is 0.3608. Although the AUC and AUPRC value are very close to separate model, the composite model has a much smaller misclassification rate. In this way, we may conclude the composite model is better.

However, we should also see not all class combinations are shown in the composite model. Totally there should be 180 classes but the composite model only have 26 of them. It means if a sample with class combination out of the 26 given classes, it will be certainly misclassified. In this way, composite model also has its negative effect side.

In the null hypothesis, we would classify randomly, thus for the separate model, the base line performance should be 0.0056 and for the composite model, the baseline performance should be 0.0385.





6. *Perform Principal Component Analysis, keeping only the 3 Principal Components (PCs) as features for the SVM classifier (no other features except of those three).*

Report the 10-fold cross-validation AUC/AUPRC value and plot the ROC/PR curves on the same plot as before. Do the PCs retain most of the classification performance while reducing the dimensionality?

In this problem, I tried to apply PCA to both original dataset which has more than 4000 features and the dataset after chosen by lasso. However, if we apply PCA on the original dataset and build SVM model, the misclassification will be more than 90% even if I standardize all the data. The Lasso chosen dataset is much better thus I will solve this problem based on the lasso dataset.

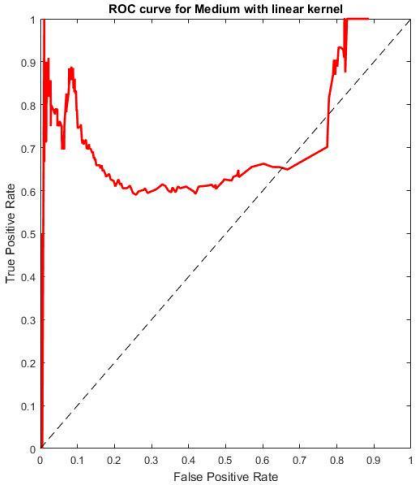
If we keep only the first three components, the variance ratio is only 0.4667 which means only 46.67% information is explained by the principle components and it seems not enough. We will compare the result with problem5 to see if it is better to only use three PCs.

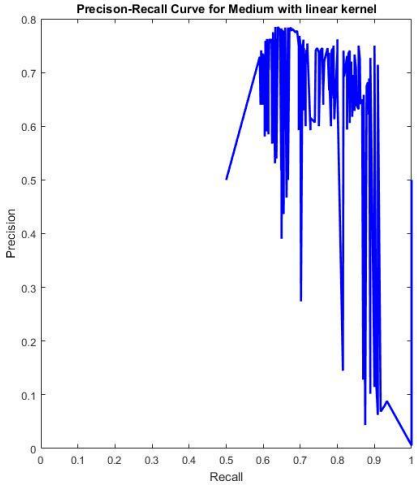
For separate model, AUC values are 0.6104, 0.8366 for medium and perturbation. AUPRC are 0.3919 and 0.1582. The misclassification rate is 0.6598.

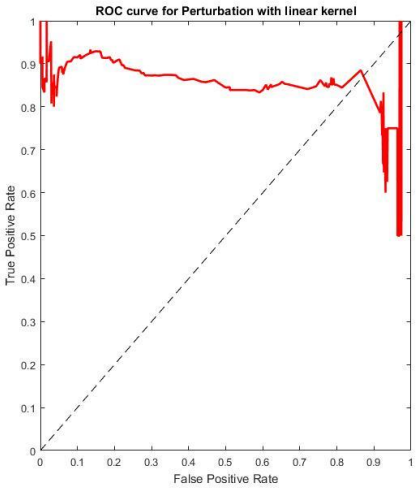
For composite model, AUC values is 0.3412, AUPRC is 0.3768. The misclassification rate is 0.6134.

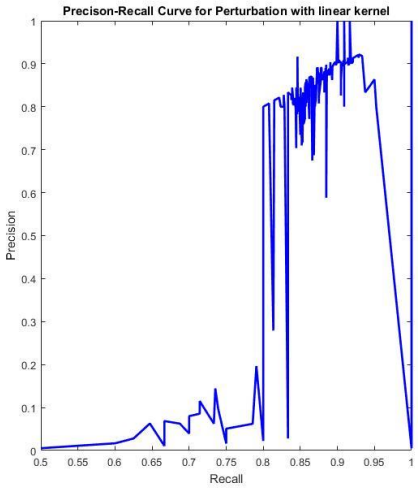
It's not hard to see AUC in PCs composite model and PCs medium model are much lower than the SVM models in problem5. On the contrary, PCs perturbation model is not influenced much by principle components. We may conclude PCs composite model performs badly primarily due to the bad performance of PCs medium model.

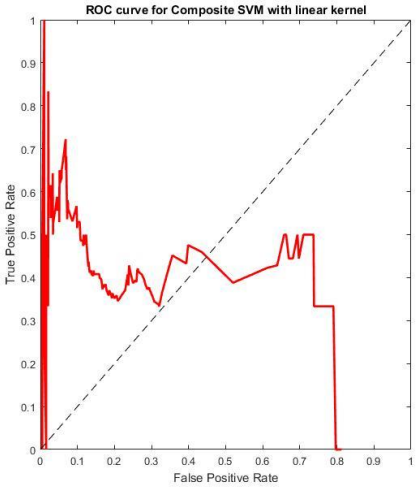
In conclusion, even if the PCs model retains some information and reduce the dimensionality, it's still not a good choice for us. However, we can promote the PCs model's performance by keeping more principle components.

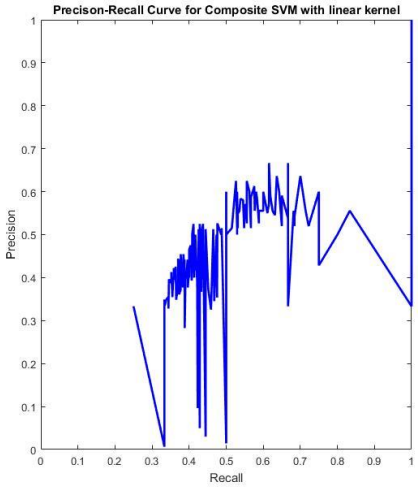





ROC curve for Perturbation with linear kernel
True Positive Rate
False Positive Rate


Precison-Recall Curve for Perturbation with linear kernel
Precision
Recall


ROC curve for Composite SVM with linear kernel
True Positive Rate
False Positive Rate


Precison-Recall Curve for Composite SVM with linear kernel
Precision
Recall