Stat601(Section001): Homework #4

Due on Apr. 6, 2021 at 11:59pm

Instructor:Ziwei Zhu

Tiejin Chen **tiejin@umich.edu**

Problem 1

With the random seed 100, We can get the first componet for pca is $[0.01005952, 0.01005264, 0.99989887]$ and the leading factor analysis direction is $[1.00006882, 0.99993068, 0.09975296]$. And we can see that if we use first componet mulitply with the data, we will get $Z_i = 0.01005952Y_{i1} + 0.01005264Y_{i2} + 0.99989887Y_{i3} \approx Y_{i3}$. Thus we can say PC component aligns itself in the direction of Y_3. And for FA, we have $Z_i = 1.00006882Y_{i1} + 0.99993068Y_{i2} + 0.09975296Y_{i3} \approx Y_{i1} + Y_{i2}$. And we can say leading factor essentially picks up the correlated component $Y_1 + Y_2$.
Now let us see why. Consider the true cov matrix of Y is:

$$Cov(Y) = \Lambda\Lambda^T + \Phi = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1.000001 & 0 \\ 0 & 0 & 100 \end{bmatrix}$$

The generated Y should have very close covariance matrix with this. First, we consider PCA. Using $\Lambda^*_{PCA} = [\Lambda_{P1}, \Lambda_{P2}, \Lambda_{P3}]^T$ to present first component of PCA and Φ^*_{PCA} to present the estimated Φ in PCA. We know in PCA, we assume each X_i have same variance, thus $\Phi^*_{PCA} = diag(\Phi_P, \Phi_P, \Phi_P)$ where Φ_P is the estimated X_i' variance. Then we know:

$$\Lambda^*_{PCA}(\Lambda^*_{PCA})^T + \Phi^*_{PCA} = \begin{bmatrix} \Lambda^2_{P1} + \Phi_P & \Lambda_{P1}\Lambda_{P2} & \Lambda_{P1}\Lambda_{P3} \\ \Lambda_{P1}\Lambda_{P2} & \Lambda^2_{P2} + \Phi_P & \Lambda_{P2}\Lambda_{P3} \\ \Lambda_{P1}\Lambda_{P3} & \Lambda_{P2}\Lambda_{P3} & \Lambda^2_{P3} + \Phi_P \end{bmatrix} \approx \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1.000001 & 0 \\ 0 & 0 & 100 \end{bmatrix}$$

We know:

$$\Lambda^2_{P1} + \Phi_P \geq \Phi_P \rightarrow \Phi_P \leq 1, \Lambda^2_{P1}\ leq1$$

$$\Lambda^2_{P2} \leq 1$$

$$\Lambda^2_{P3} + \Phi_P \approx 100 \rightarrow \Lambda^2_{P3} \geq 99$$

This is a approximated result. However we can still see $\Lambda_{P3} >> \Lambda_{P1}, \Lambda_{P3} >> \Lambda_{P2}$. Hence after we centered the data, $\Lambda_{P3} \approx 1$.
In other words, PCA needs to extract most information from Y_3 to 'explain' the cov matrix as much as possible because 100 is the largest number and is much much bigger than other number in matrix and Φ_P is fixed and a small number for all three variables. That is the reason PCA aligns itself in the direction of $Y3$.

Now let us see factor analysis, the key difference here is FA assumes different variances for different X_i. We use $\Lambda^*_{FA} = [\Lambda_{F1}, \Lambda_{F2}, \Lambda_{F3}]^T$ to present leading factor and Φ^*_{FA} to present the estimated Φ in FA. And we know $\Phi^*_{FA} = diag(\Phi_{F1}, \Phi_{F2}, \Phi_{F3})$. Now we have:

$$\Lambda^*_{FA}(\Lambda^*_{FA})^T + \Phi^*_{FA} = \begin{bmatrix} \Lambda^2_{F1} + \Phi_{F1} & \Lambda_{F1}\Lambda_{F2} & \Lambda_{F1}\Lambda_{F3} \\ \Lambda_{F1}\Lambda_{F2} & \Lambda^2_{F2} + \Phi_{F2} & \Lambda_{F2}\Lambda_{F3} \\ \Lambda_{F1}\Lambda_{F3} & \Lambda_{F2}\Lambda_{F3} & \Lambda^2_{P3} + \Phi_{F3} \end{bmatrix} \approx \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1.000001 & 0 \\ 0 & 0 & 100 \end{bmatrix}$$

The key difference with above analysis is that we do not need to care about the diagonal line of the matrix because Phi_{Fi} can be different from each other and can be any number greater than 0. For example $\Lambda^2_{P3} = 99$ then Φ_{F3} could be 1 and $\Lambda^2_{P3} = 1$ then $\Phi_{F3} = 99$. And it will not affect Φ_{F1} and Φ_{F2} at all. Thus we only need to care about $\Lambda_{F1}\Lambda_{F2} \approx 1$ and $\Lambda_{F1}\Lambda_{F3} \approx 0, \Lambda_{F2}\Lambda_{F3} \approx 0$. And we also need to make $\Lambda^2_{F1} \leq 1, \Lambda^2_{F2} \leq 1$. Notice this does not need meet strictly. However that is our goal. Since $\Lambda^2_{F1} \leq 1$, to make $\Lambda_{F1}\Lambda_{F3} \approx 0$, $\Lambda_{F3} \approx 0$. And to make $\Lambda_{F1}\Lambda_{F2} \approx 1$, $\Lambda_{F1} \approx 1, \Lambda_{F2} \approx 1$. And that gives us the result that leading factor picks up the correlated component $Y_1 + Y_2$.
In other words, FA takes more care about the covariance bewteen each variables instead of variance as we do in PCA. Thus, FA will get the most information from correlated Y_1 and Y_2 and ignore Y_3.

Problem 2

(a)
we use $\lambda = 0.05$, And we get the result:

- standardized data. For test data, we will use mean and std from training set to standardize them. After 87 epoches, the algorithm converges, And the training mean error rate is 0.071731 while the test mean error rate is 0.073011.
- log transformed data. After 504 epoches, the algorithm converges, And the training mean error rate is 0.05771 while the test mean error rate is 0.05671.
- discretized data. After 16 epoches, the algorithm converges, And the training mean error rate is 0.057059 while the test mean error rate is 0.080834.

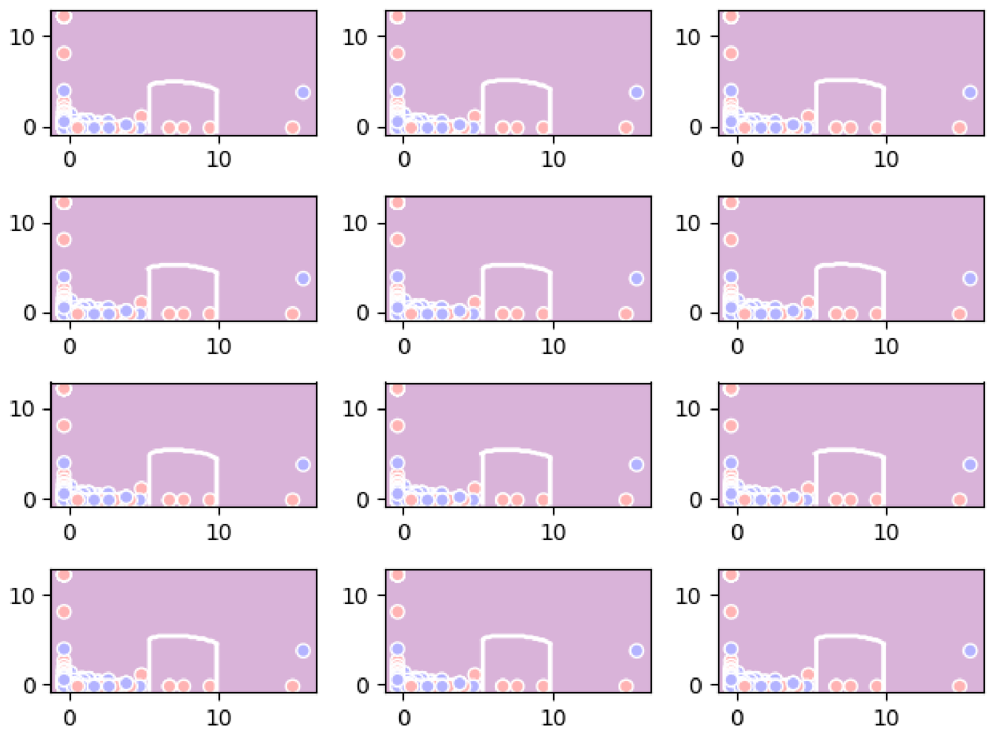
(b)

- standardized data. For test data, we will use mean and std from training set to standardize them. The test mean error rate for LDA is 0.09126.
- log transformed data. The test mean error rate for LDA with log transformed data is 0.064537.

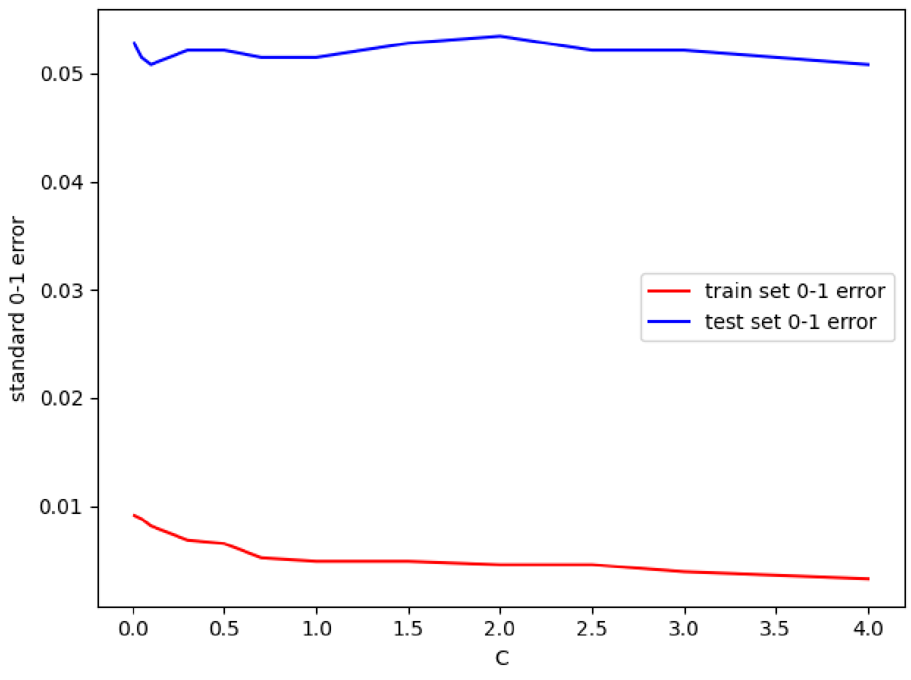
(c)
For Naive Bayes and discretized data, we get test mean error rate is 0.081486.
(d)
We will use codes from lab this part. And to make our codes as simple as possible, we delete some repeated job code and make many codes annotation.

1. standardized data
   In this part, we will discuss two model's performance on standardized data.
   - kernel rbf
     First we adjust C to see the its affect. We use 12 different $C = [0.01, 0.05, 0.1, 0.3, 0.5, 0.7, 1.0, 1.5, 2.0, 2.5, 3.0, 4.0]$. And we get the plots of boundry line:

Problem 2 continued on next page. . .



We can see change C does not affect the boundry shape. And we also draw the plot of training and test mean error rate:

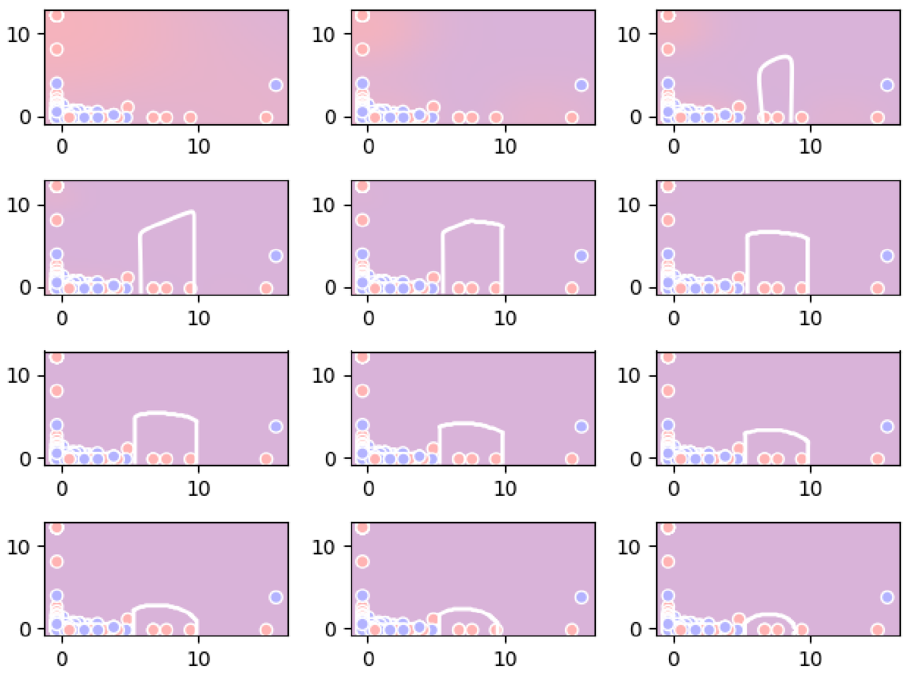


We can see it does not affect model too much.

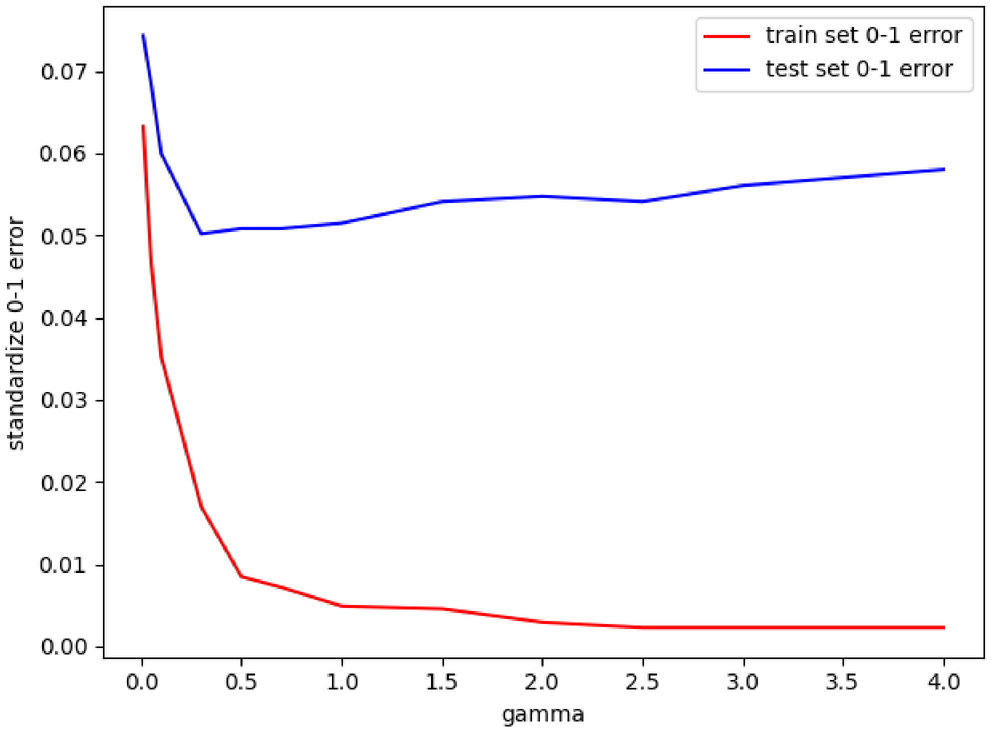
Now we consider γ. Similarly, we use 12 different $gamma = [0.01, 0.05, 0.1, 0.3, 0.5, 0.7, 1.0, 1.5, 2.0, 2.5, 3.0, 4.0]$.

Problem 2 continued on next page. . .

And we draw boundry:



We can see with the change of gamma, the boundry change quite a bit. And the plot of training and test mean error rate:

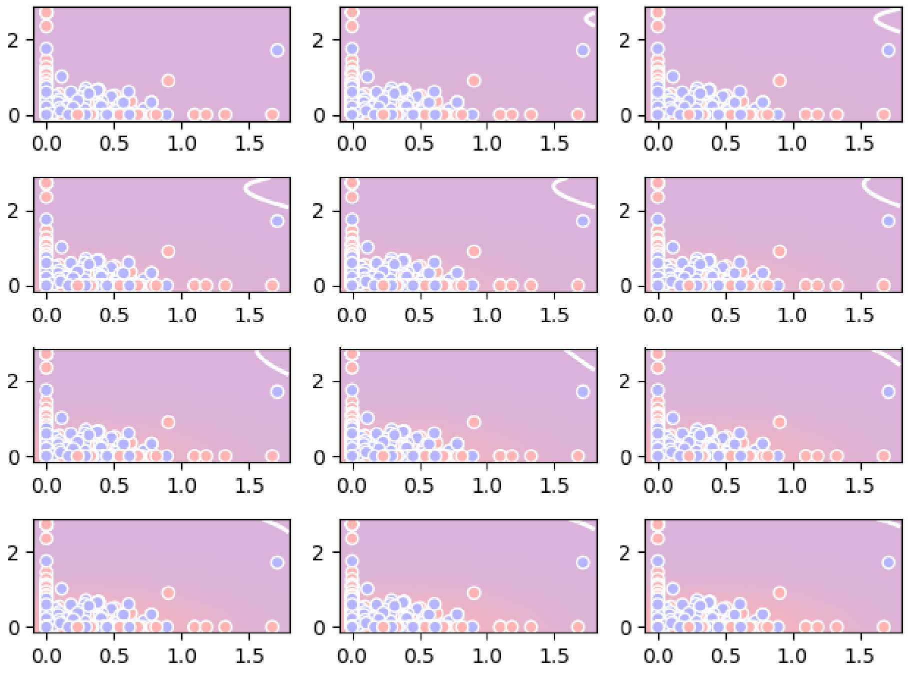


We can see both test and train error go down as gamma increse before around 0.5. And after

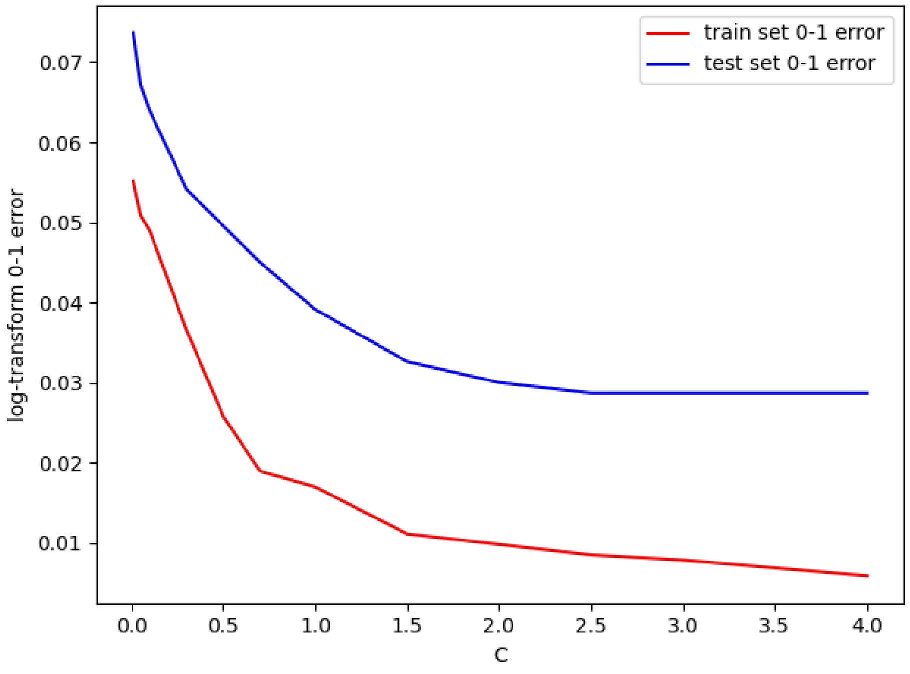
Problem 2 continued on next page. . .

that, as gamma increses, the train error decreases slowly while test set error also increases. It is a bit like overfitting.

- log-transformed data. Agian we first consider C.

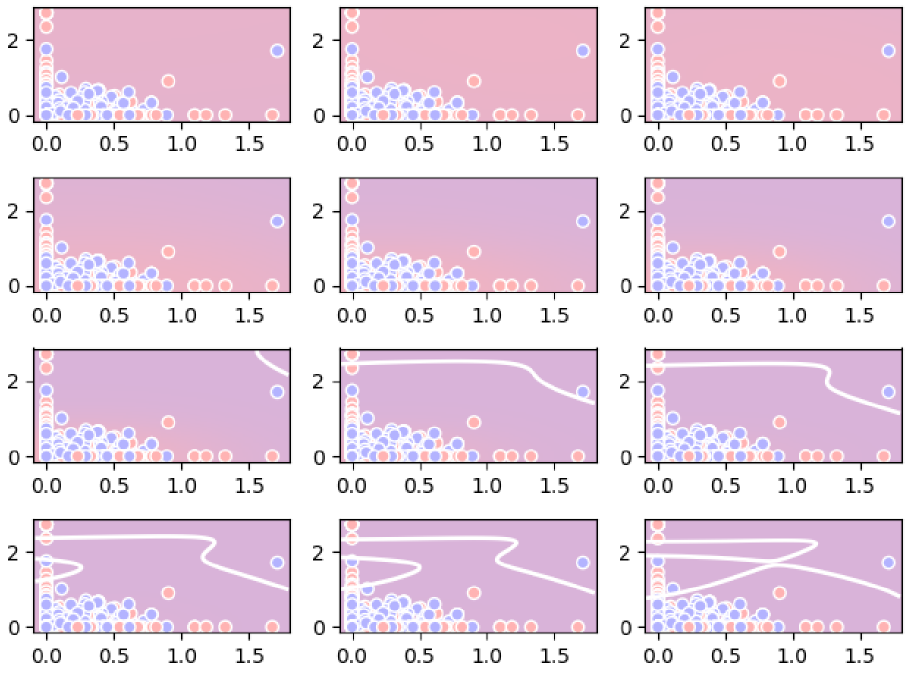


It seems bigger C make the boundry line more straight.

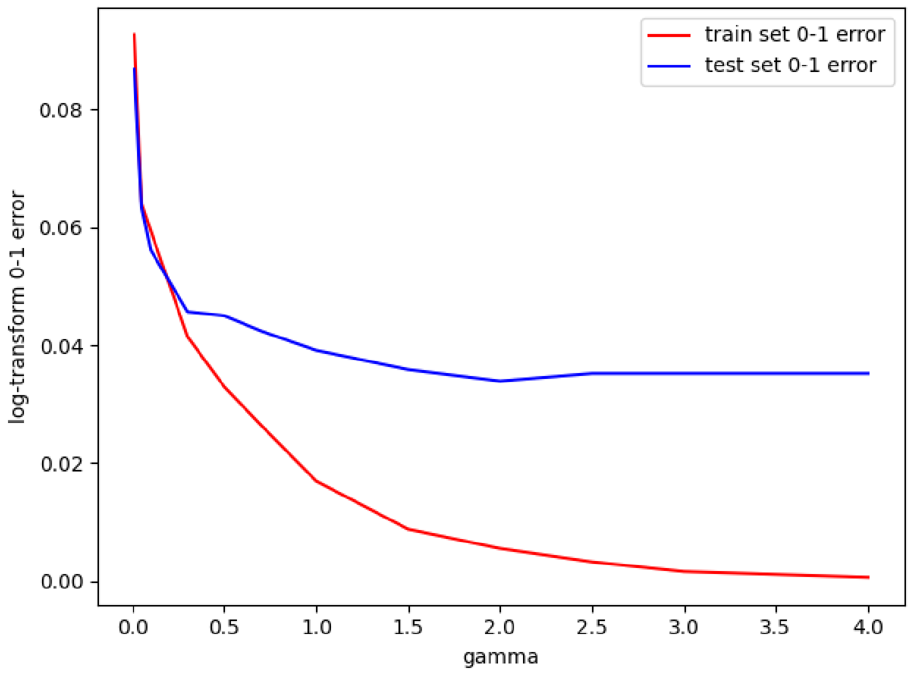


Unlike the previous situation, with log-transformed data, both test and train error decrease as C increases.

For gamma's boundry plot, we have



The boundry changes a lot with the change of gamma. We have

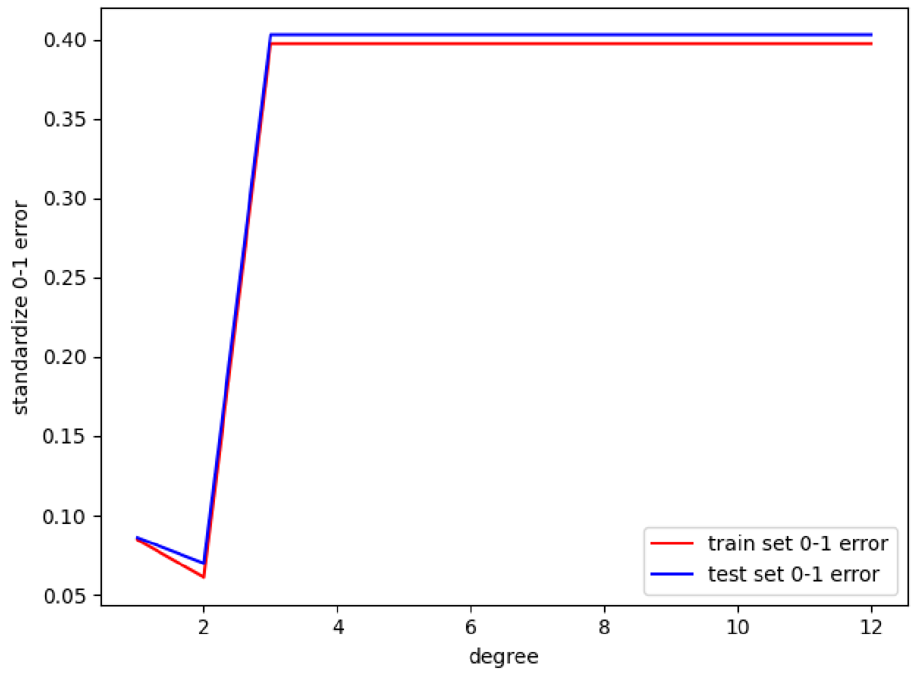


It shows the similar trend with standardized data. However turning point here is around 1.5 instead of 0.5.

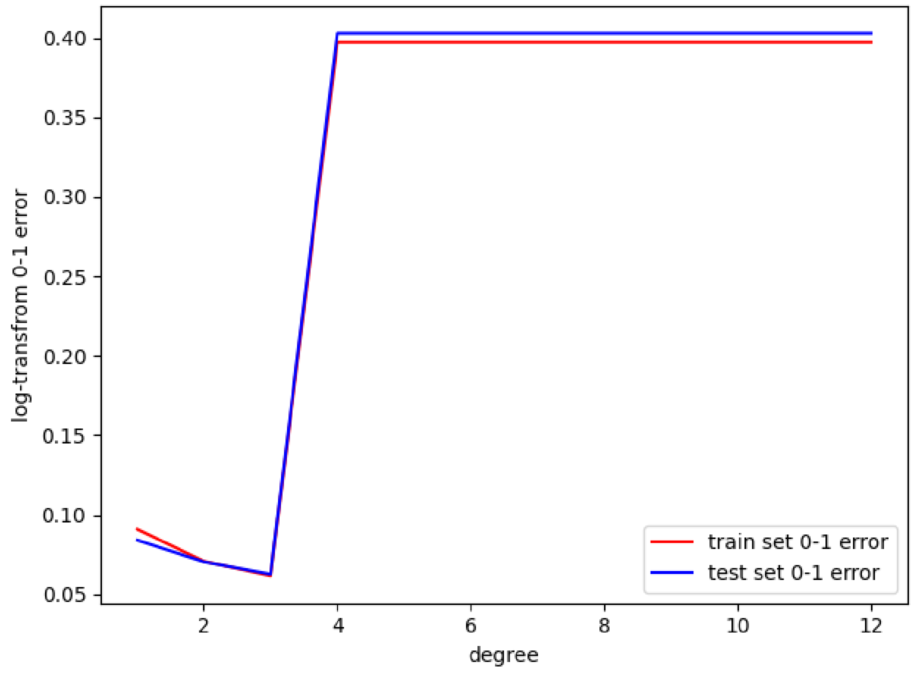
Problem 2 continued on next page. . .

2. polynomial kernel For polynomial kernel, we only consider degree. And we will use $\gamma = gamma = \frac{1}{57}$ which is the default setting from sklearn polynomial kernel here.

   - standardized data We can get the plot:



   - log-transformed data We can get the plot:



Both data shows similar trend, we should only consider degree less than 4.

Problem 2 continued on next page. . .

For the final test error in table, we will use GridSearchCV from sklearn to re-find a best hyperparameter. And for rbf kernel, we will search from $C = [0.01, 0.05, 0.1, 0.5, 1.0, 4.0, 10.0]$, $gamma = [0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 2.0]$ For polynomial kernel, we will search from $degree = [1, 2, 3, 4]$, $gamma = [0.001, 0.005, 0.01, 1/57, 0.05, 0.1, 0.5]$. And we can get the table for all test error.

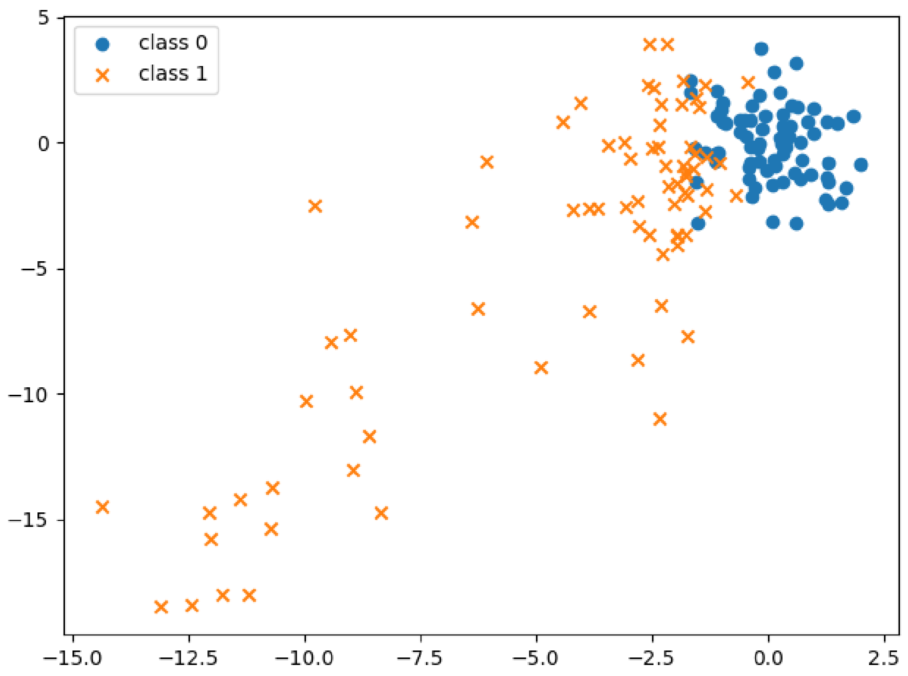
	standardize	log-transform	discretized
logit_regression	0.073011	0.056710	0.080834
LDA	0.091264	0.064537	NaN
Naive_Bayes	NaN	NaN	0.081486
rbf_logit	0.046280	0.026728	NaN
poly_logit	0.069752	0.058018	NaN

We can see the best method should be kernel logistic regression, And log-transform is the best preprocessing method to this data. We can also get LDA with standardized data is the worst method.

Problem 3

(a)

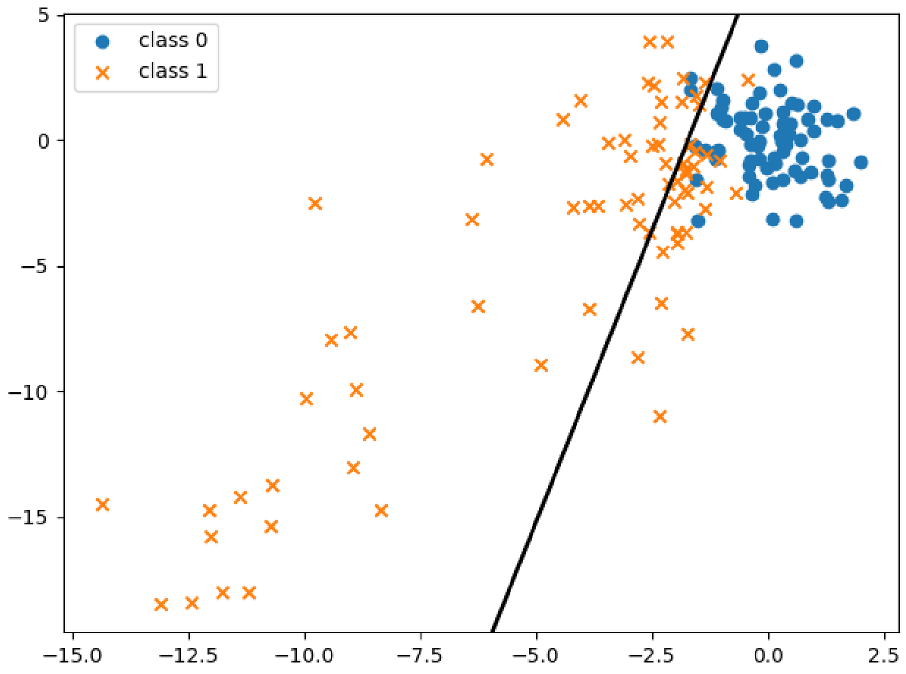
We get the plot:



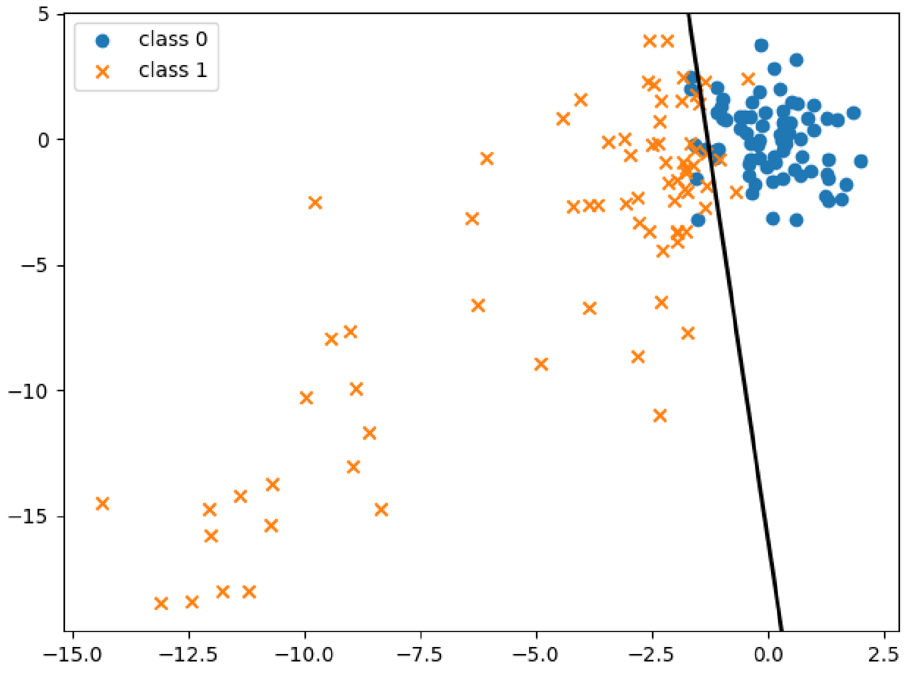
(b)

posterior probability of class 1 is 0.5. And we get the plot:

Problem 3 continued on next page. . .

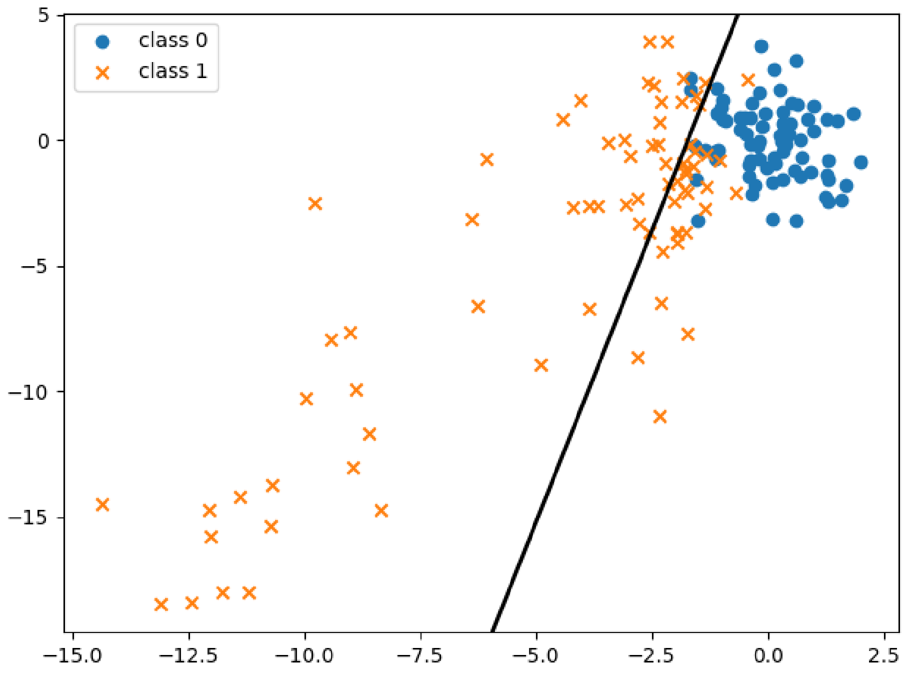


(c)
We get the plot:



(d(e))
We get the plot:

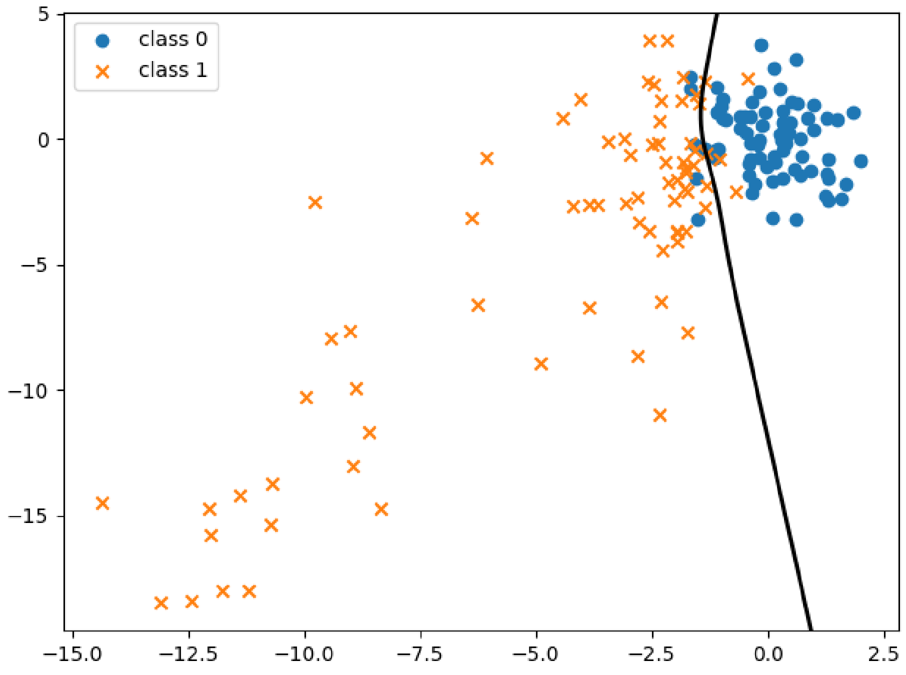
Problem 3 continued on next page. . .



It is the same as LDA.

(e(f))

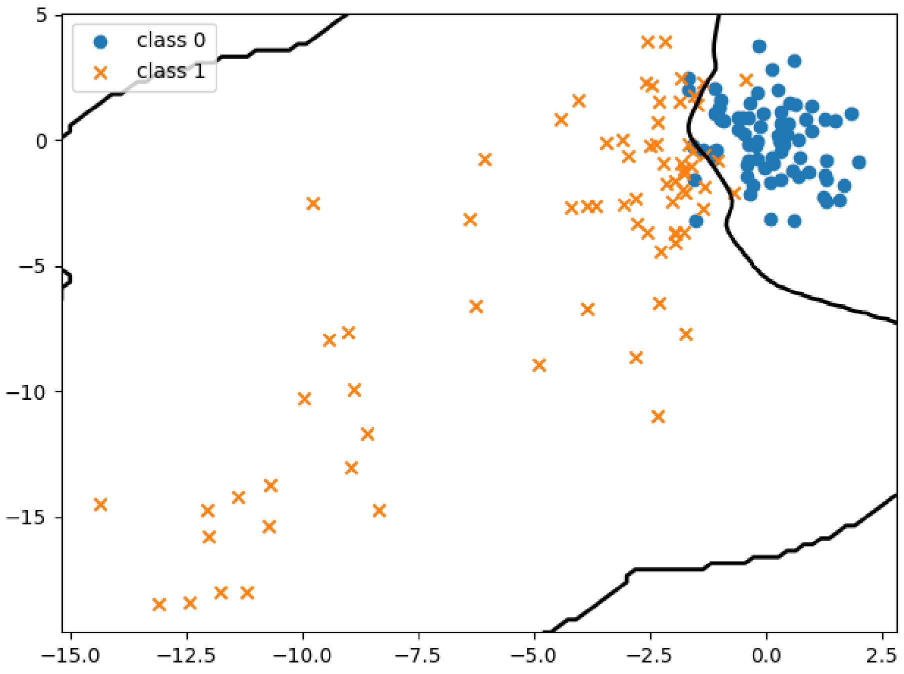
We get the plot;



(f(g))

We get the plot;

Problem 3 continued on next page. . .



(g(h))
We get the test error rate result for all the method:

LDA	logit	linear regression	addtive logit	kernel logit
0.1733	0.1	0.1733	0.1	0.0933

And we can see kernel logit with Gaussian kernel is the best method here. And additive logistic regression and normal logistic regression are the second best method. Linear regression and LDA get same result under this dataset. However, both additive logistic regression and kernel logistic regression are much complex than normal logistic regression while they only have same or a little bit higher performance on test set. Thus we should consider only use normal logistic regression.