Yu Wang

Handwritten digits Recognition by Machine Learning Techniques

Introduction

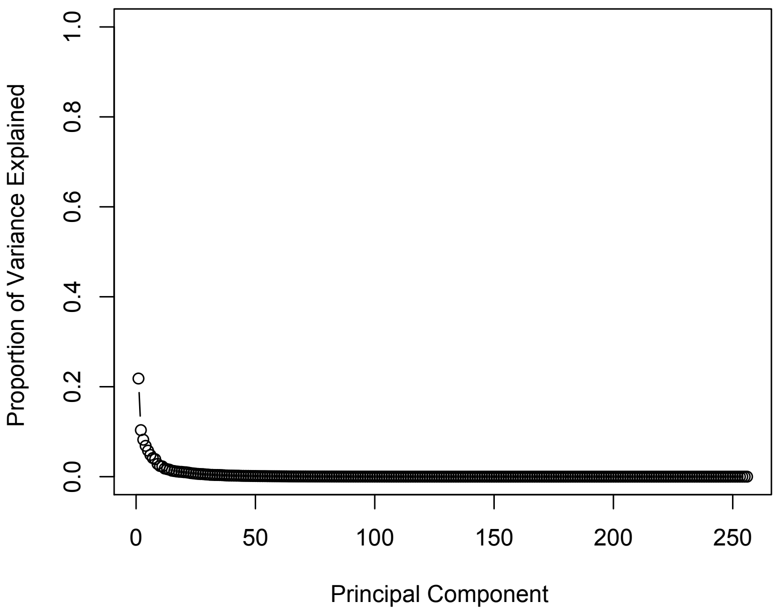
As complex data collected in many fields, it becomes emerging to obtain automatic tools for analyzing these data. Machine learning techniques are such tools, for grouping data into intrinsic clusters.
We use A database of scanning images of handwritten digits from US Postal Services envelopes. Each observation is a handwritten digit (from 1 to 10), given by a 16_16 grayscale images. There are 256 features and 9298 observations in total, each feature is a grayscale intensity. In our project, we aim at finding an automatic procedure for recognizing digits given this 16_16 grayscale images.
Source: “usps” from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Methodology and Results

1. PCA

We perform the PCA and draw screen plot representing the proportion of the variances explained by the PCs in figure 1.

Figure 1: Proportion of Variance Explained by PCs



There are 256 variables and 256 PCs in total. It is rational to keep less than 20 PCs from the screen plot. If we assume the cumulative PVE is no less than 80%, we can reduce the dimension to 16 according to table 1.

Table 1. Cumulative PVE of First 16Pcs

PC	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
PVE	0.218	0.103	0.082	0.068	0.057	0.048	0.041	0.039

Cumulative PVE	0.218	0.321	0.403	0.472	0.530	0.578	0.619	0.659
PC	**PC9**	**PC10**	**PC11**	**PC12**	**PC13**	**PC14**	**PC15**	**PC16**
PVE	0.028	0.023	0.022	0.018	0.016	0.0155	0.013	0.0124
Cumulative PVE	0.687	0.710	0.733	0.7518	0.768	0.7841	0.797	0.8099

Unfortunately, we are unable to visualize the 16 PCs in a plot, and the first two PCs only explained 32.1% of the y variation, thus, the plot for PC1 and PC2 is not representative. Methods below are applied to the full data and the first 16 PCs of the data. Then we can compare their results with the true label and predicted clustering, which represented by error rate.
For Validation sets, the full data is separated into two parts, 80% of the full data is the training set and the 20% left is the testing set, however, in KNN test, we applied K-fold to access the optimal K.
In the computation, we permute predicted cluster labels in the ground truth and use the minimal percentage as the error.

2. LDA

Linear Discriminant Analysis (LDA) method find 9 LDs to separate training data into 10 clusters, the proportion of trace of each LD is listed in table 2.1., which indicates the percentage of contribution performed by each LD during the discrimination. The correct prediction result applied in testing data showed in table 2.2. Compared to the previous method we performed, LDA generates a much better clustering result with correct prediction rate 91.39%.
We applied LDA method to 16 PCs we obtained from the PCA step, also include 9 LDs and separate the 80% training PC data to 10 clusters (table 3). The correct prediction rate is 88.33%, which is little lower than performed on full data, but still good.

Table 2.1. Proportion of Trace of Each LD

LDs	**LD1**	**LD2**	**LD**	**LD**	**LD5**	**LD6**	**LD7**	**LD8**	**LD9**
Proportion	0.3489	0.2007	0.1143	0.1068	0.0793	0.0648	0.0362	0.0296	0.0195

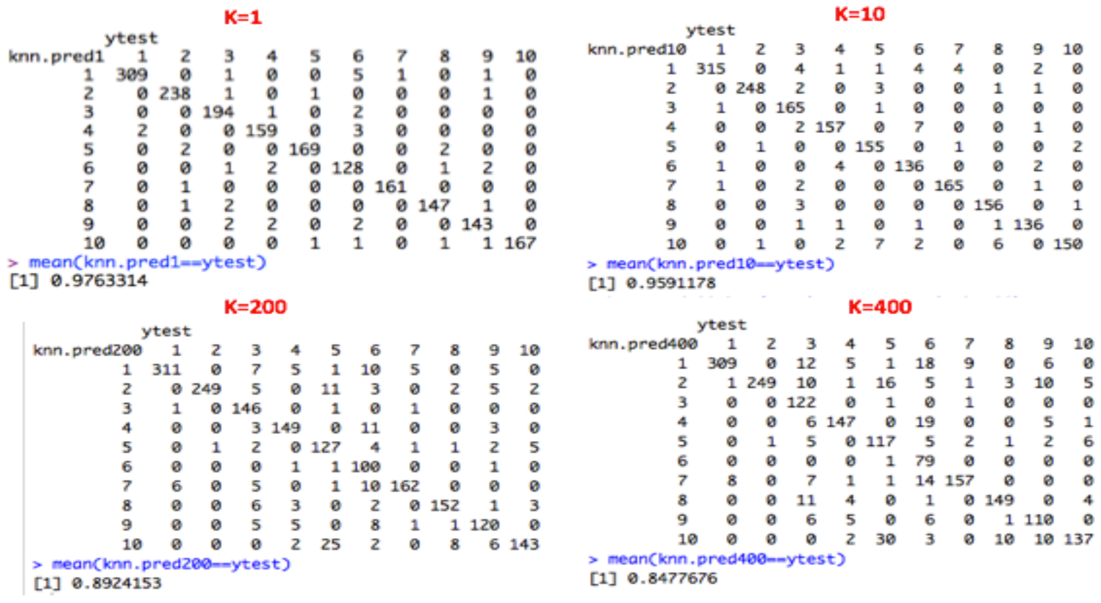
Table 2.2. Error Rates on Full Data and 16 PCs.

	Full data		**16PCs**	
	Test	**Train**	**Test**	**Train**
Correct Rate	91.39%	92.88%	88.6%	88.31%
Error Rate	8.61%=1-0.9139	7.12%=1-0.9288	11.4%=1-0.886	11.69%=1-0.8831

3. KNN

K-nearest neighbors gave the best results among all the methodologies we implemented. Figure 3.1. shows the typical prediction table with the largest numbers on the diagonal and all the other entries rather small. This method usually generates a percentage of correct prediction as high as 95% and test error rate 5% or even lower when k is below 10.

Figure 3.1. *prediction table of k=1,10,200, and 400 by setting seed (565)*



K=1

```
          ytest
knn.pred1   1   2   3   4   5   6   7   8   9  10
       1  309   0   1   0   0   5   1   0   1   0
       2    0 238   1   0   1   0   0   0   1   0
       3    0   0 194   1   0   2   0   0   0   0
       4    2   0   0 159   0   3   0   0   0   0
       5    0   2   0   0 169   0   0   2   0   0
       6    0   0   1   2   0 128   0   1   2   0
       7    0   1   0   0   0   0 161   0   0   0
       8    0   1   2   0   0   0   0 147   1   0
       9    0   0   2   2   0   2   0   0 143   0
       10   0   0   0   0   1   1   0   1   1 167
> mean(knn.pred1==ytest)
[1] 0.9763314
```

K=10

```
          ytest
knn.pred10   1   2   3   4   5   6   7   8   9  10
        1  315   0   4   1   1   4   4   0   2   0
        2    0 248   2   0   3   0   0   1   1   0
        3    1   0 165   0   1   0   0   0   0   0
        4    0   0   2 157   0   7   0   0   1   0
        5    0   1   0   0 155   0   1   0   0   2
        6    1   0   0   4   0 136   0   0   2   0
        7    1   0   2   0   0   0 165   0   1   0
        8    0   0   3   0   0   0   0 156   0   1
        9    0   0   1   1   0   1   0   1 136   0
        10   0   1   0   2   7   2   0   6   0 150
> mean(knn.pred10==ytest)
[1] 0.9591178
```

K=200

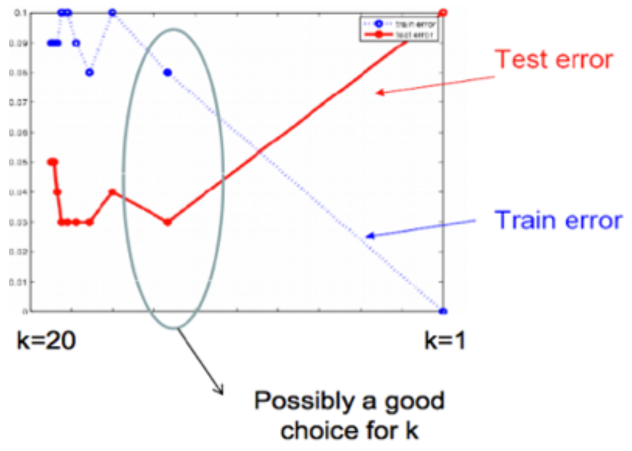
```
           ytest
knn.pred200   1   2   3   4   5   6   7   8   9  10
          1 311   0   7   5   1  10   5   0   5   0
          2   0 249   5   0  11   3   0   2   5   2
          3   1   0 146   0   1   0   1   0   0   0
          4   0   0   3 149   0  11   0   0   3   0
          5   0   1   2   0 127   4   1   1   2   5
          6   0   0   0   1   1 100   0   0   1   0
          7   6   0   5   0   1  10 162   0   0   0
          8   0   0   6   3   0   2   0 152   1   3
          9   0   0   5   5   0   8   1   1 120   0
          10  0   0   0   2  25   2   0   8   6 143
> mean(knn.pred200==ytest)
[1] 0.8924153
```

K=400

```
           ytest
knn.pred400   1   2   3   4   5   6   7   8   9  10
          1 309   0  12   5   1  18   9   0   6   0
          2   1 249  10   1  16   5   1   3  10   5
          3   0   0 122   0   1   0   1   0   0   0
          4   0   0   6 147   0  19   0   0   5   1
          5   0   1   5   0 117   5   2   1   2   6
          6   0   0   0   0   1  79   0   0   0   0
          7   8   0   7   1   1  14 157   0   0   0
          8   0   0  11   4   0   1   0 149   0   4
          9   0   0   6   5   0   6   0   1 110   0
          10  0   0   0   2  30   3   0  10  10 137
> mean(knn.pred400==ytest)
[1] 0.8477676
```

Figure 3.2. explains the logic of finding the optimum k: In the beginning when k values are large, both training errors and testing errors are not quite stable, going through some ups and downs. After a while both errors begin to decrease steadily due to the increasing flexibility of model. Then at a certain point, the test error reaches its last local minimum and starts to increase monotony ever since then—This is the moment after which the model would become too flexible that begins over-fitting the test data.

Figure 3.2. Procedure of Finding K.



Therefore, we aim to find this last local minimum by observing the test error rate for different k values. Ten-fold cross validation is implemented to find the percentage of correct classification and test error rate. Table 3.1. shows the test error rates of k values from 1 to 400 when setting seeds (565) and (500), as well as three random seeds generating by R studio. In order to shorten running time, figures in the table are computed by using the first 16 principle components rather than full data.

Table 3.1. Test Error Rates of K Value From 1 to 400 (seed = 565)

	set.seed(565)		set.seed(500)		random seed #1		random seed #2		random seed #3	
k	**%(correct classification)**	**test error**	**%(correct classification)**	**test error**	**%(correct classification)**	**test error**	**%(correct classification)**	**test error**	**%(correct classification)**	**test error**
1	0.9398	0.0602	0.9398	0.0602	0.9398	0.0602	0.9398	0.0602	0.9398	0.0602
2	0.9333	0.0667	0.9355	0.0645	0.9387	0.0613	0.9398	0.0602	0.9323	0.0677
3	0.9462	0.0538	0.9484	0.0516	0.9473	0.0527	0.9462	0.0538	0.9516	0.0484
4	0.9473	0.0527	0.9505	0.0495	0.9484	0.0516	0.9473	0.0527	0.9441	0.0559
5	0.9452	0.0548	0.9495	0.0505	0.9473	0.0527	0.9484	0.0516	0.9452	0.0548
6	0.9495	0.0505	0.9505	0.0495	0.9495	0.0505	0.9516	0.0484	0.9462	0.0538
7	0.9538	**0.0462**	0.9516	0.0484	0.9527	0.0473	0.9527	**0.0473**	0.9516	**0.0484**
8	0.9484	0.0516	0.9538	**0.0462**	0.9548	**0.0452**	0.9505	0.0495	0.9495	0.0505
9	0.9473	0.0527	0.9505	0.0495	0.9495	0.0505	0.9484	0.0516	0.9452	0.0548
10	0.9398	0.0602	0.9409	0.0591	0.9419	0.0581	0.9419	0.0581	0.9430	0.0570
11	0.9387	0.0613	0.9398	0.0602	0.9409	0.0591	0.9409	0.0591	0.9398	0.0602
...	...	...	...	...	...	...	...	...	...	...
100	0.8989	0.1011	0.8989	0.1011	0.8978	0.1022	0.8989	0.1011	0.9011	0.0989
...	...	...	...	...	...	...	...	...	...	...
200	0.8742	0.1258	0.8742	0.1258	0.8742	0.1258	0.8742	0.1258	0.8742	0.1258
...	...	...	...	...	...	...	...	...	...	...
300	0.8591	0.1409	0.8591	0.1409	0.8591	0.1409	0.8591	0.1409	0.8591	0.1409
...	...	...	...	...	...	...	...	...	...	...
400	0.8473	0.1527	0.8462	0.1538	0.8473	0.1527	0.8473	0.1527	0.8473	0.1527
...	...	...	...	...	...	...	...	...	...	...

The local minimum test error rate varies slightly between k=7 and k=8 under 10-fold cross validation, with the range of approximately 0.045 to 0.050, which is pretty low. The training error rate results are indicated in table 3.3.
Moreover, the whole process above could have been done by using full data, if the long waiting time is not an issue. Table 3.2. presents the comparison of the results using 16 PCs and full data, under k=1,10,200, and 400. It could be noticed that for each of the four k values, KNN-full data delivers a higher correct prediction rate and lower test error.

Table 3.2. Test Error Rates on Full Data and 16 PCs.

	k=1		k=10		k=200		k=400	
	%(correct prediction)	Test error	%(correct prediction)	Test error	%(correct prediction)	Test error	%(correct prediction)	Test error
KNN-16pc	0.9478	0.0522	0.9527	0.0473	0.8768	0.1232	0.8451	0.1549
KNN-full data	0.9763	0.0237	0.9591	0.0409	0.8924	0.1076	0.8478	0.1522

Table 3.3. Training Error Rate for K=7 & 8

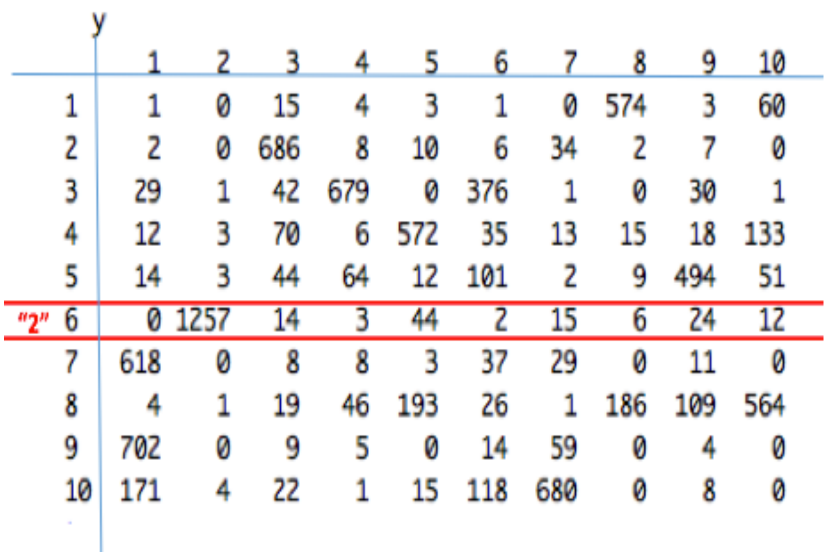
	Full Data		16 PCs	
K	7	8	7	8
Training Error Rate	2.37%	5.7%	3.64%	3.85%

Therefore, the optimum k for conducting KNN method would be either k equals to 7 or 8. The optimum test error rate would be roughly 0.045-0.05 by principle component analysis. The result would be improved slightly by using full data without PCA, though it would take much longer for computation.

4. K-Means

Similar with KNN, a classification table could be generated as shown in Figure 4.1.. However, instead of having largest numbers sitting on the diagonal, this table hardly have a clear pattern because the cluster labels from [1] to [10] are randomly assigned to each of cluster. For instance, for cluster labeled [6], though as many as 1257 data points are assigned to true value [2], the true label correspond to this biggest cluster should be labeled itself as number [6]. Similarly, the the largest number 686 for cluster [2] should be correspond to true label [3]. Therefore, all the ten clusters should be re-labeled in this way so that it would represent the “correct classification”.

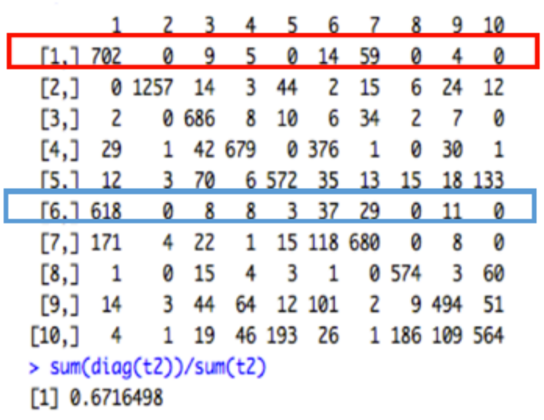
Figure 4.1. Classification Table for KNN



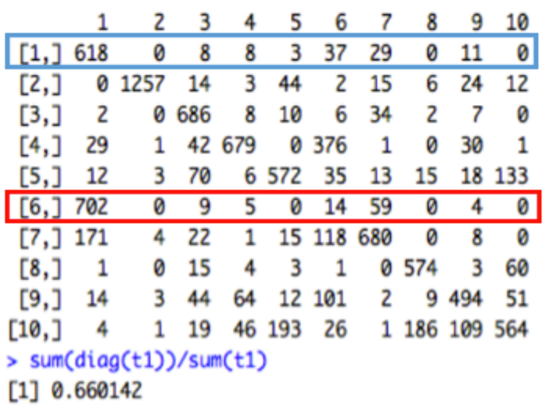
	y 1	2	3	4	5	6	7	8	9	10
1	1	0	15	4	3	1	0	574	3	60
2	2	0	686	8	10	6	34	2	7	0
3	29	1	42	679	0	376	1	0	30	1
4	12	3	70	6	572	35	13	15	18	133
5	14	3	44	64	12	101	2	9	494	51
“2” 6	0	1257	14	3	44	2	15	6	24	12
7	618	0	8	8	3	37	29	0	11	0
8	4	1	19	46	193	26	1	186	109	564
9	702	0	9	5	0	14	59	0	4	0
10	171	4	22	1	15	118	680	0	8	0

To make this happen, we could have programmed to maximize the ratio of “Between sum of square” over “Total sum of square” so that it would generate the result of such correct classification. However, the 10! permutation took too long to output the result. Therefore, we did the “re-ordering” by hand. We reordered rows based on observation such that the new label assigned to each row be the true label of the largest group within each row. In this way, the column sum stays the same so that the observations belong to each true label will not change with clustering labels. During this “re-labeling” process, a problem appears that both row #7 and row #9 are supposed to be relabeled as “1”while none of the ten rows are relabeled as“6”. Taking two situations into consideration, Figure 4.2. shows the two possible reordering of these two cases.

Figure 4.2. Two Possible Re-ordering Classification Tables.



```
        1    2   3   4   5   6   7   8   9  10
 [1,] 702    0   9   5   0  14  59   0   4   0
 [2,]   0 1257  14   3  44   2  15   6  24  12
 [3,]   2    0 686   8  10   6  34   2   7   0
 [4,]  29    1  42 679   0 376   1   0  30   1
 [5,]  12    3  70   6 572  35  13  15  18 133
 [6,] 618    0   8   8   3  37  29   0  11   0
 [7,] 171    4  22   1  15 118 680   0   8   0
 [8,]   1    0  15   4   3   1   0 574   3  60
 [9,]  14    3  44  64  12 101   2   9 494  51
[10,]   4    1  19  46 193  26   1 186 109 564
> sum(diag(t2))/sum(t2)
[1] 0.6716498
```



```
        1    2   3   4   5   6   7   8   9  10
 [1,] 618    0   8   8   3  37  29   0  11   0
 [2,]   0 1257  14   3  44   2  15   6  24  12
 [3,]   2    0 686   8  10   6  34   2   7   0
 [4,]  29    1  42 679   0 376   1   0  30   1
 [5,]  12    3  70   6 572  35  13  15  18 133
 [6,] 702    0   9   5   0  14  59   0   4   0
 [7,] 171    4  22   1  15 118 680   0   8   0
 [8,]   1    0  15   4   3   1   0 574   3  60
 [9,]  14    3  44  64  12 101   2   9 494  51
[10,]   4    1  19  46 193  26   1 186 109 564
> sum(diag(t1))/sum(t1)
[1] 0.660142
```

As shown in the graph, the percentage of correct classification are roughly 67.16% and 66.01%. Notice that figures are achieved by using all data. We also follow the same procedures by using the first 16 PCs, where we also got two possible rankings for cluster #1 and #6, with the corresponding correct classification rate to be 57.30% and 57.96% respectively.

Therefore, we claim the best for K means is to use all data without conducting principle component analysis, and the highest correct classification rate is 67.16%, suggesting the lowest test error rate 32.84%. This is a much higher test error compared with KNN, probably due to the fact that our data is more applicable to classification (supervised learning) approaches than clustering (unsupervised learning) methods since the true labels are available to all data.

Table 4.1. Training Error Rates on Full Data and 16 PCs.

	Full Data	**16 PCs**
Correct Rate	67.16%	57.96%
Error Rate	32.94%	42.04%

5. Hierarchical & EM

Since EM and K-means are similar, we did EM clustering as well. Hierarchical clustering is another major clustering method we applied and the results are shown in table 5.1. Table below gives the training error rate.

Table 5.1. Training Error Rate on EM and Hierarchical.

	Full Data	**16 Pcs**
Hierarchical Error Rate	60.10% = 1 - 0.3990	66.26% = 1 – 0.3372
EM Error Rate	NA (no converge due to large database)	33.08% = 1 - 0.6692

Conclusion

We summarize the clustering results in the following table.

Table 6. Summarize of Classification and Clustering.

	Full Data	**16 PCs**
Classification Testing Error Rate		
LDA	8.61%=1-0.9139	11.4%=1-0.886
KNN	2.37%	3.64%
Clustering Training Error Rate		
K-Means	32.84% = 1 - 0.6716	42.04% = 1 – 0.5796
EM	NA (no converge due to large database)	33.08% = 1 - 0.6692
Hierarchical	58.86% = 1 - 0.4114	66.26% = 1 – 0.3372

Among Classification methods, KNN perform the best in both full data and the first 16 PCs, none of the algorithm benefit from the dimension reduction procedure, that is reducing the dimension jeopardize the accuracy of the clustering. The second best method is LDA, which is also a classification method. Clustering methods K-means and EM clustering have similar performance in 16 PCs. Hierarchical perform the worst overall. It's probability because of the large database and the high dimension of the data. The clusters cannot be separated, this problem does not favor the unsupervised clustering methods, classification methods outperformed clustering methods because classification provided more information than clustering.

Even though we manually ranked the row orders to obtain the error rates, the training error rates of clustering are still far larger than classifications'. Moreover, due to the high dimension, it's tedious to permute the row order of prediction table to achieve an optimum error rate.

We believe that SVM and Spectral Clustering are accessible, but the database is so large that R studio is incapable of obtaining the result.

Since the dependent variables y are categories instead of numbers, PLS, PCR, Lasso and Ridge Regression are not feasible here, however, they may be good approaches if we can combine them with LDA or other classification models.