Project 3: Unsupervised Learning and Dimensionality Reduction

CS-7641 Machine Learning
Spring 2015
Name: Fujia Wu
GT Account: fwu35

1. Implementation of Tools and Dataset

For this assignment, I used the scikit-learn tool for clustering, dimension reduction and feature selection. For neural network learning, I used the neurolab module. The following table summarizes the specific modules that I used for all requirements of this project.

KMeans	sklearn.cluster.KMeans
EM/GMM	sklearn.mixture.GMM
PCA	sklearn.decomposition.PCA
ICA	sklearn.decomposition.FastICA
RCA	sklearn.random_projection.GaussianRandomProjection
Univariate Feature Selection	sklearn.feature_selection.SelectKBest
Neural Network	neurolab.net.newff

The datasets I used is the Car and Mushroom datasets I used in Project 1. The Car dataset is a multi-class problem with 4 classes, 6 features and 1728 instances. The Mushroom dataset is a binary class problem with 22 features and 8124 instances.

2. Clustering

2.1 *KMeans*

The KMeans clustering is implemented in the two files: kmeans_car.py and means_mushroom.py. Before running the clustering, the data feature values are converted into numeric values and then standardized using the sklearn.preprocessing.scale module, so that the results are independent of the absolute values of the features.

To judge the performance of the clustering, I computed the following scores using the sklearn.metrics module: 1) Homogeneity Score; 2) Completeness Score; 3) V-measure Score; 4) Adjusted Random Score; 5) Adjusted Mutual Information Score; 6) Mean Silhouette Coefficient. These scores are different measure of the averaged "similarity" of all the data within each cluster. The specific definitions of these scores can be found in the scikit-learn

documentation (http://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics). They all vary from 0 to 1, or from -1 to 1, with the higher score indicating "better" clustering.

I ran the KMeans clustering with different given number of clusters (n_clusters), and then observed the running time and the quality of the clustering. For simplicity, only the Homogeneity Score and Mean Silhouette Coefficient are considered in the following. Figure 1 plots the results for KMeans on the Car dataset and Mushroom dataset. First, it is that there is some scattering in the curves. This is because I started all KMeans with random initial means. Second, it is seen that the Homogeneity Score increases with the number of clusters. For Car data, the Silhouette coefficient increases with number of clusters slowly; while for Mushroom data, the Silhouette coefficient first increases and then decreases with the number of clusters. For both datasets, the running time increases with the number of clusters. Therefore, the number of clusters should be decided based on the considerations on both the running time and performance measures.

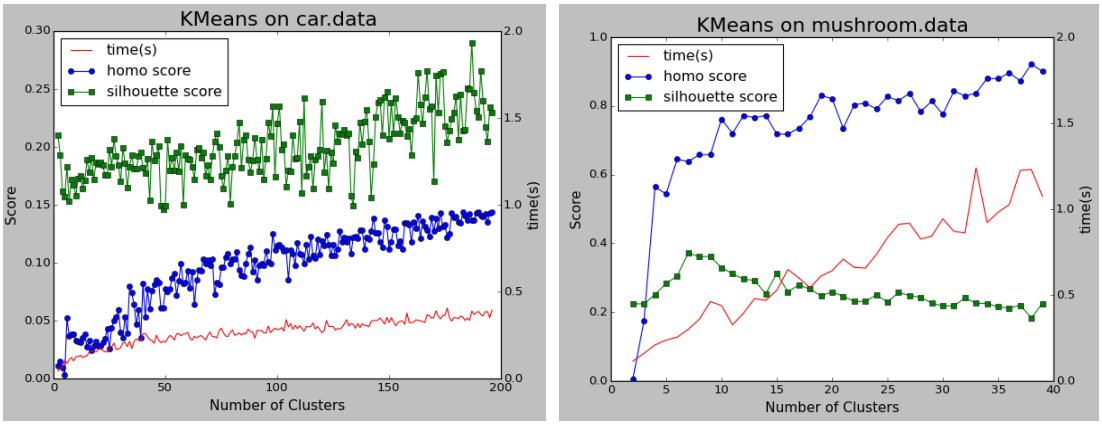

Figure 1. Running time, Homogeneity Score and Mean Silhouette Coefficient for KMeans on the Car and Mushroom datasets

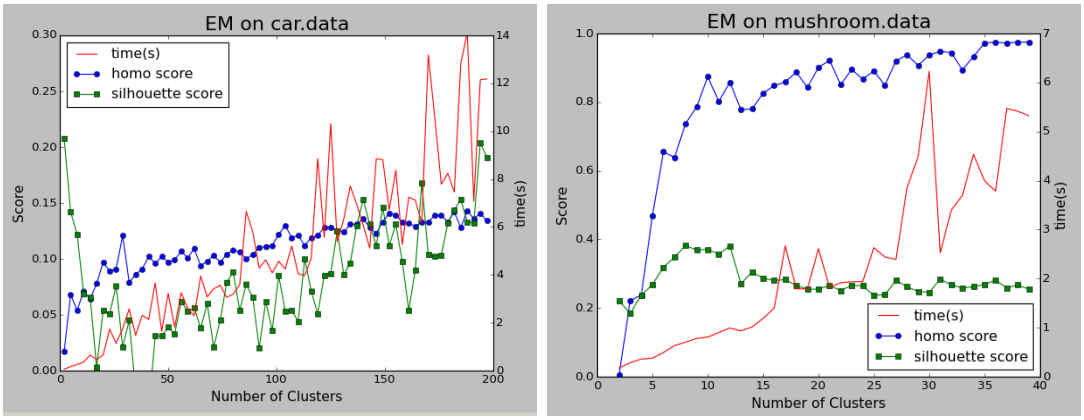

Figure 2. Running time, Homogeneity Score and Mean Silhouette Coefficient for EM on the Car and Mushroom datasets

2.2 *Expectation Maximization*

The Expectation Maximization clustering is implemented in the two files: em_car.py and em_mushroom.py. The same data preprocessing is done as in the case for KMeans. To judge the performance of the EM clustering, I used the same performance as in the case for KMeans. Figure 2 plots the running time, Homogeneity Score and Mean Silhouette Coefficient as a function of the number of clusters.

Similarly, it is seen that as the number of clusters increase, the performance score can be increases with the expense of increased running time. The difference is that the EM algorithm costs significantly more running time, compared to the KMeans algorithm. This is obviously reasonable as the EM is more complex. For each step in EM, the probability density function needs to be evaluated and optimized, while for KMeans only the mean and nearest center needs to be calculated. The performance scores for KMeans and EM are similar; expect that KMeans seems to have higher Silhouette coefficient than EM on Car dataset, while EM seems to have higher Homogeneity score than KMeans on Mushroom dataset. No general conclusion can be drawn at this point.

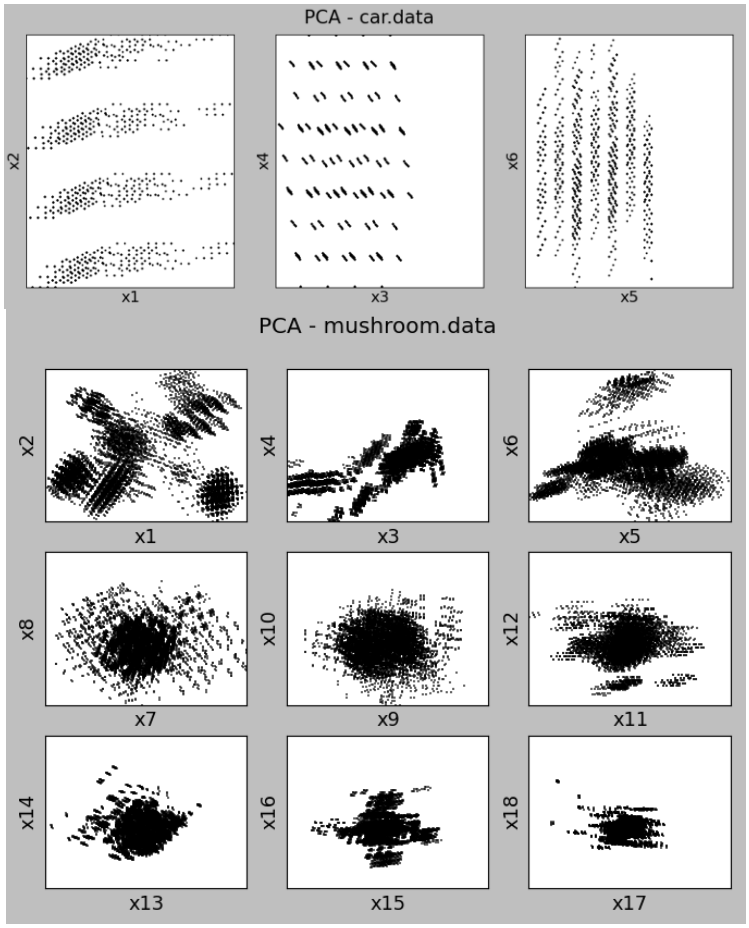

Figure 3. Data after PCA against the first 6 PCA components on the Car data and the first 18 PCA components Mushroom data

3. Dimension Reduction

3.1 PCA

The Principal Component Analysis is implemented in the two files: pca_car.py and pca_mushroom.py. Figure 3 plots the Car and Mushroom data set after PCA against first 6 PCA components on the Car dataset and the first 18 PCA components Mushroom dataset. The Mushroom dataset has 22 features in total. From the plot for the Mushroom data, it is seen clearly that the data scattering (variance) is the largest for 1st and 2nd PCA component and then the rest of the components. The variance of the data scattering clearly decreases as the PCA component number increases. This result is reasonable because the algorithm PCA is to identify the linear combination of features that maximize the variance. The Car dataset only has 6 features; therefore, the effect of PCA is not quite obvious but still it can be seen that the data scattering against the 1st and 2nd PCA components is larger than the rest 3rd-6th PCA components. The running time for PCA on these two datasets are pretty fast: 0.001s and 0.014s respectively for Car and Mushroom datasets.

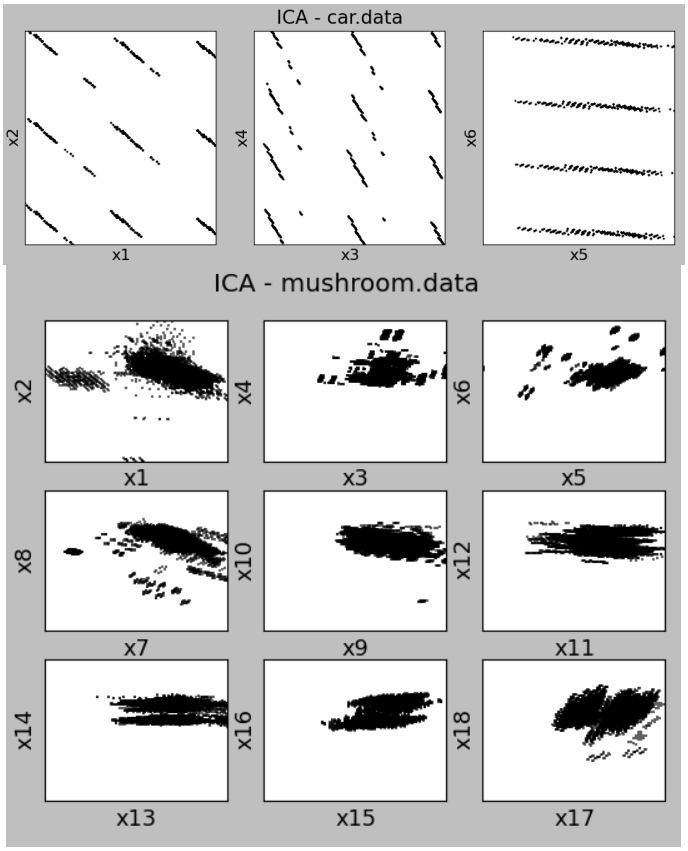

Figure 4. Data after ICA against the first 6 ICA components on the Car data and the first 18 ICA components Mushroom data

3.2 ICA

The Independent Component Analysis is implemented in the two files: ica_car.py and ica_mushroom.py. Figure 4 plots the Car and Mushroom data set after ICA against first

6 ICA components on the Car dataset and the first 18 ICA components Mushroom dataset. From Figure 4, it is seen that different from PCA there is no specific trend in the data scattering (variance) against the ICA component number. This makes sense because ICA does not maximize the variance but seek the mutually independent components. On the other hand, although ICA does not maximize the data variance, it is seen from Figure 4 that the ICA-transformed data form on parallel lines (instead of random scattering). This is because ICA is seeking linear combination of features that are mutually independent, and as a result the data variation in each plot in Figure 4 tends to form parallel lines because in each plot the x ICA component and y ICA component are very independent. The running time for ICA on these two datasets is much slower compared to PCA: 0.008s and 16.833s respectively for Car and Mushroom datasets. This is reasonable because ICA needs to calculate the mutual information.

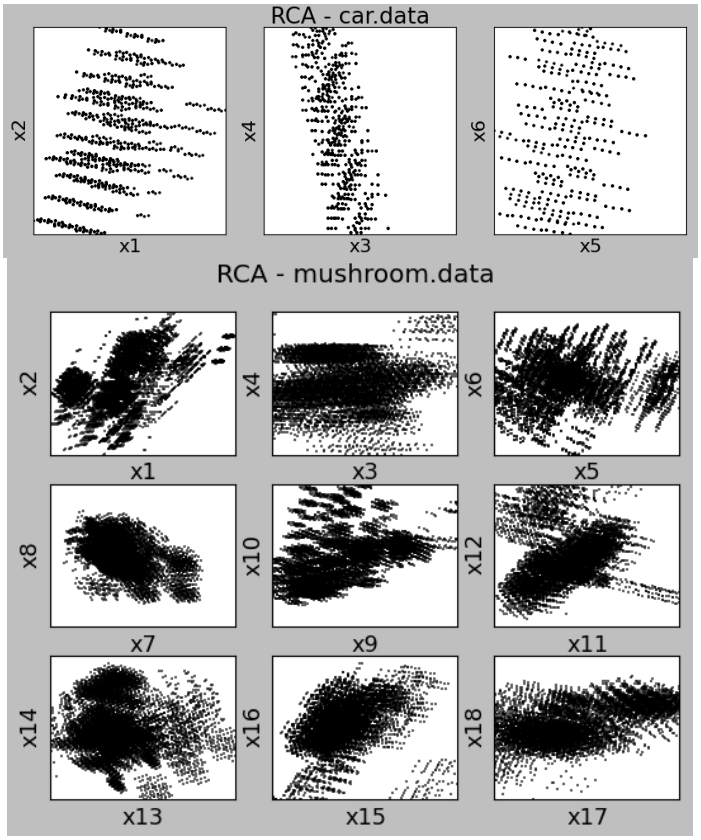

Figure 5. Data after RCA against the first 6 RCA components on the Car data and the first 18 RCA components Mushroom data

3.3 Random Projection (RCA)

The Random Projection or Random Component Analysis (RCA) is implemented in the two files: rp_car.py and rp_mushroom.py. Figure 5 plots the Car and Mushroom data set after RCA against first 6 RCA components on the Car dataset and the first 18 RCA components Mushroom dataset. Again, from Figure 5 it is seen that the data variance does not change obviously against the RCA component number. Also the scattering of data in each plot

seems quite random. The running time for RCA is even faster than PCA: 0.001s and 0.008s respectively for Car and Mushroom datasets. This is reasonable because RCA does not do any optimization for maximum variance or independence.

3.4 Univariate Feature Selection

The final feature select algorithm I used is the Univariate Feature Selection algorithm. This algorithm selects the features based on the calculated score based on each feature and the label value. The score I chose is the chi-square test measures. The chi-square test measures dependence between stochastic variables, so using this function "weeds out" the features that are the most likely to be independent of class and therefore irrelevant for classification. Figure 6 plots the results of Univariate Feature Selection on the Mushroom dataset. The number data points in Figure 6 are much smaller than the previous case because this is the selected features from the original features, not linear combination of the original features. There are only limited values for each feature and thus many data points overlap on each other. From Figure 6, the features from x1 to x18 are in best-worst ordering. But no specific pattern can be seen here because the selection rule depends on the labels.

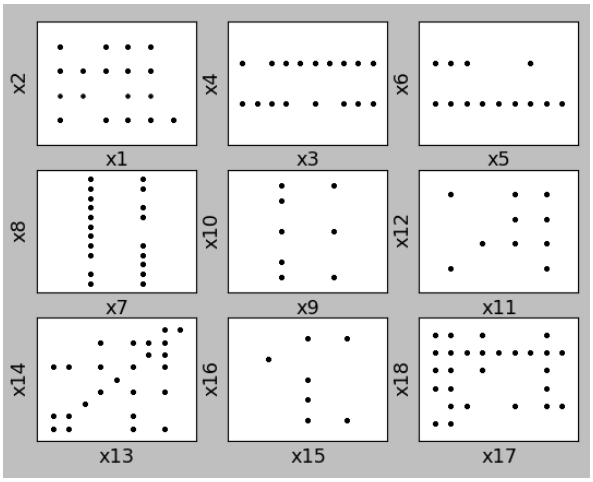

Figure 6. Mushroom dataset plotted against the first 18 features selected from the 22 features

4. Clustering after Dimension Reduction

I performed the KMeans and EM clustering algorithm again after the data is reduced by PCA. This also allowed the clustering results to be visualized. I first reduced the Car and Mushroom datasets using the PCA algorithm to 2 dimensions (*i.e.*, only keep 2 principal components). Then I performed the KMeans and EM clustering algorithms on the reduced datasets. Figure 7 and Figure 8 plot the clustering result on the Car and Mushroom datasets respectively.

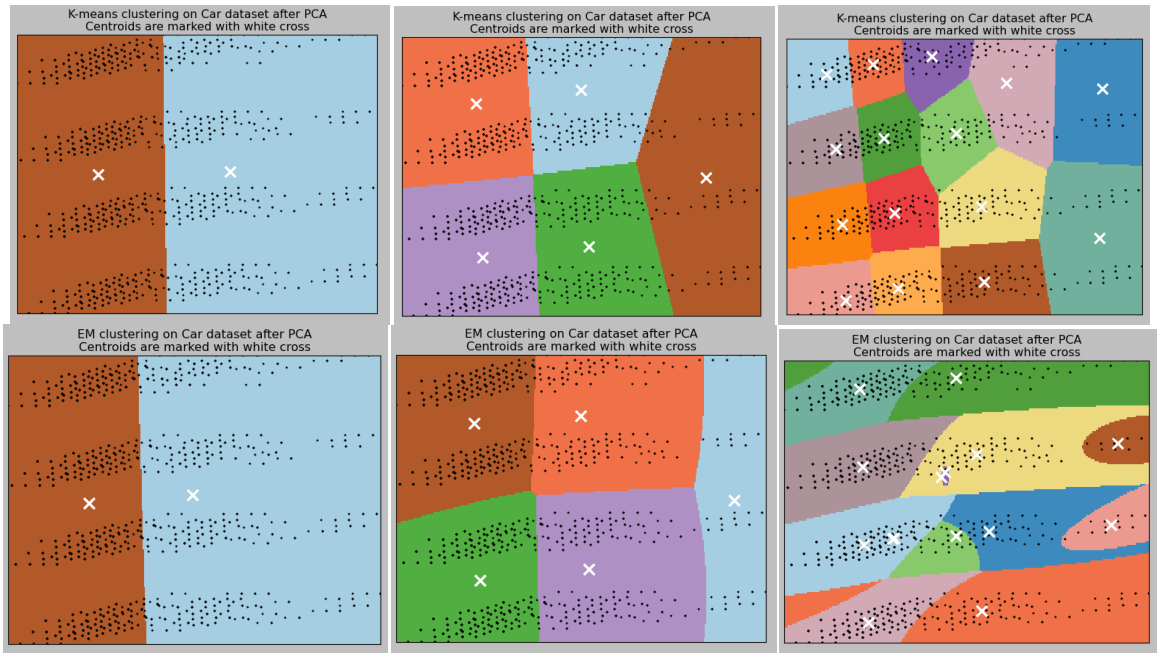

Figure 7 The clustering results of KMeans (top 3 subfigures) and EM (bottom 3 subfigures) on the Car dataset with number of clusters = 2 (left), 5 (middle) and 15 (right).

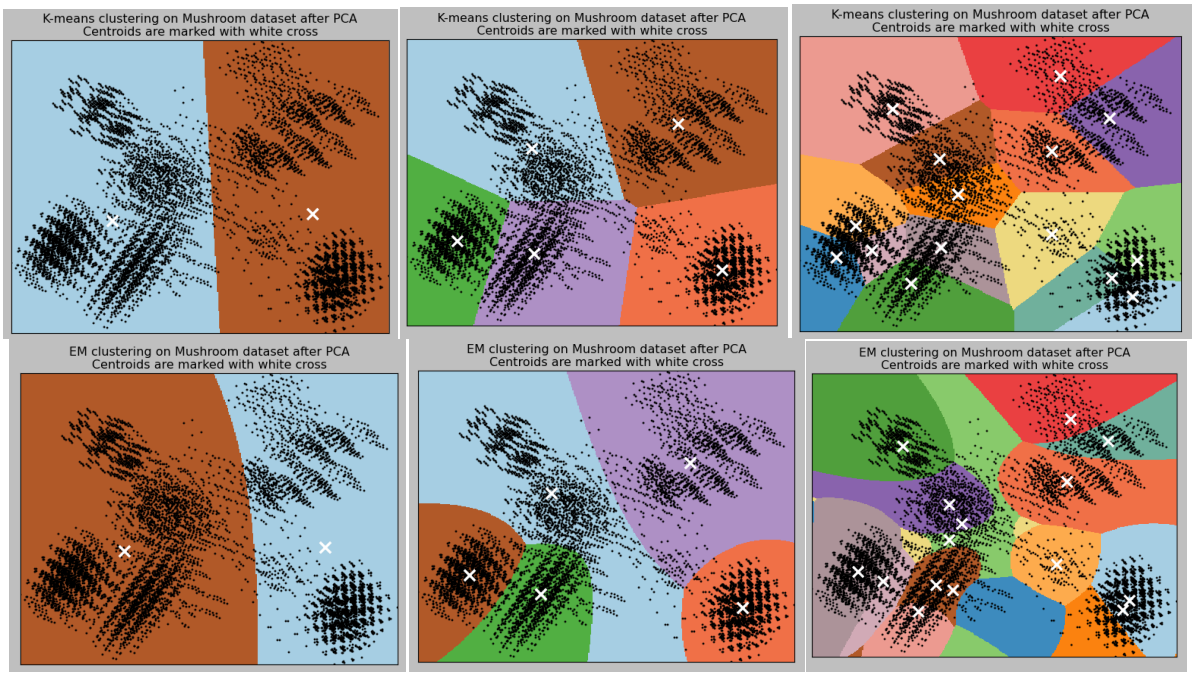

Figure 8. The clustering results of KMeans (top 3 subfigures) and EM (bottom 3 subfigures) on the Mushroom dataset with number of clusters = 2 (left), 5 (middle) and 15 (right).

From Figures 7 and 8, it is seen that KMeans and EM algorithms have different behaviors. KMeans is based on the distance purely, therefore, it literally cluster the data based on the number of data points and their distance. The boundary lines tend to straight lines. On the other hand, the EM algorithm here assumes the data are generated by Gaussian

probability density functions. Therefore, the boundary lines between different clusters tend to be circular arc lines. In addition, the EM assumes (n_cluster-1) Gaussian functions (where n_cluster is the number of clusters), and the data points not in these (n_cluster-1) clusters are assigned to the final cluster. This is why in the EM results (Mushroom data) there is always a cluster that has a irregular shape.

5. Run Neural Network Learner after Dimension Reduction

I have completed only half of this part of the assignment. The neural network is taking too long time to run in my case, and I was not able to find enough time to conduct the data plotting and analysis on this part. Another reason is that I did Project 1 in Matlab, and Project 3 using sklearn, I could not reuse the code in Project 1, which delayed my progress. Nevertheless, the following are the neural network learning code on the Mushroom dataset in the file pca_mushroom.py.

```
# Perform Neural Network Learning
skf = StratifiedKFold(labels, n_folds=4)
train_index, test_index = next(iter(skf))
x_train = reduced_data[train_index]
x_test = reduced_data[test_index]
y_train = labels[train_index]
y_test = labels[test_index]
size = len(x_train)
y_train = y_train.reshape(size,1)
for N in range(1, 4):
    t0 = time()
    print("running NN with N=%d " % (N))
    x_run = x_train[:,:N]
    input = np.array([[-7, 7] for x in range(N)])
    net = nl.net.newff(input,[10, 1])
    error = net.train(x_run, y_train, epochs=100, goal=0.01)
    t = time()-t0
    print("NN time spent: N=%d, t=%0.3f" % (N, t))
```

My plan was to run NN learner with only a few PCA or ICA components and monitor the training and testing error. At the same time, I would like to calculate the running time. With only a few components (features), the NN learner should be able to run with much less time but with comparable accuracy.

6. Summary

In this assignment, I used 2 clustering algorithms: KMeans and Expectation Maximization, as well as 4 dimension reduction (feature selection) algorithms: PCA, ICA, RCA and Univariate Feature Selection. I have applied them on the Car and Mushroom datasets that I used in Project 1. Results and features of each algorithm are presented and discussed.