Unsupervised Learning and Dimensionality Reduction

k-means clustering | Expectation Maximization | PCA | ICA | Randomized Projections | Select k-best

1. Introduction:

1.1 Algorithm explanation

This is an analysis about unsupervised learning. The type of algorithms that I used is Clustering algorithm, specifically, k-means clustering and expectation maximization. Then, I will apply four dimensionality reduction algorithms to improve the performance of clustering algorithm.

1.2 Dataset

Wine Quality Dataset: A dataset with 4898 samples about the quality of wines. This dataset is associated with classification and regression. There are eleven features in this dataset. This means that there will be a high dimension situation occurring during the learning process. However, since the dimension reduction algorithm will be applied, it's no need to cut features based on their correlations. For PCA and ICA, the wine quality dataset will be a good example to reduce the dimension. Unlike the supervised learning which we did the binary classification, the unsupervised learning focuses on feature classification and feature learning so that not preprocessing needed for the quality feature. Url: https://archive.ics.uci.edu/ml/datasets/wine+quality

NBA Players Dataset: (I will say "players dataset" in the following)A dataset contains aggregate individual statistics from 67 NBA seasons. The csv file used in unsupervised learning is called Seasons_Satas.csv with 18147 instances. Originally, there are 53 features in this dataset so that the running time will be extremely high if not applying any reduction. In this way, the preprocessing will remove high correlated features, null features. In the raw dataset, the Pos features are labeled by letters which means this feature to be impossible to be applied. The preprocessing is scaled it into the range of 5. Url: https://www.kaggle.com/drgilermo/nba-players-stats#Seasons_Stats.csv.

2. Implementation:

2.1 k-means clustering algorithm

2.1.1 Brief explanation: For this algorithm, we need randomly pick some centers, calculating the score, than relocate the centers and recalculating until we cannot better off the situation. The parameter in code: 'n_clusters' is just how many centers we want to pick. Basically, if there is a set of objects X and distance D(X,y) = D(y, x), x and y belongs to X.

2.1.2 sklearn.metrics.normalized_mutual_info_score(): This is a method in Sci-kit learn library. The mutual information shows the reduction in the entropy of class labels that we get if we know the cluster labels. Think about the information gain in decision tree. Now, we use NMI as a measurement of the quality of clustering. Basically, if the NMI of a clustering is high, then it just means this clustering has a well performance.

2.1.3 sklearn.metrics.silhouette_score(): Silhouette analysis is used on the study of separation distance between the clusters. More specifically, the silhouette plot shows a measure of how close each point in one cluster is to points in the neighboring cluster. The silhouette coefficients near +1 indicate that the sample is far away from the neighboring clusters. The value 0 means the sample is very close to the decision boundary.

2.1.4 analysis:

n_clusters	**NMI**	**Silhouette_score**
2	0.381	0.294
4	0.288	0.214
6	0.324	0.166

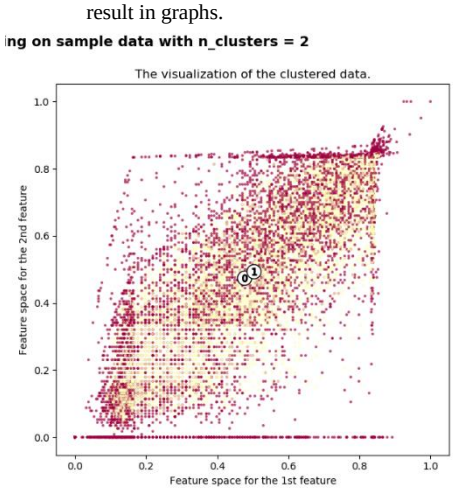
Table1 for Players dataset

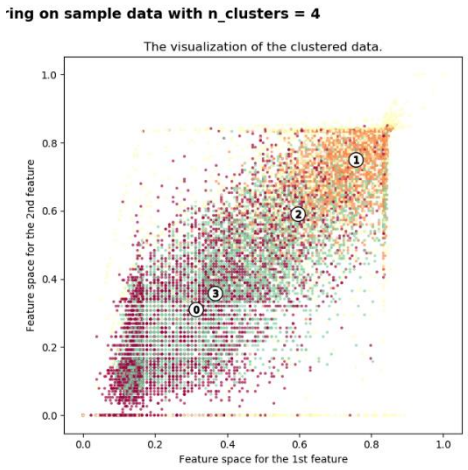
n_clusters	**NMI**	**Silhouette_score**

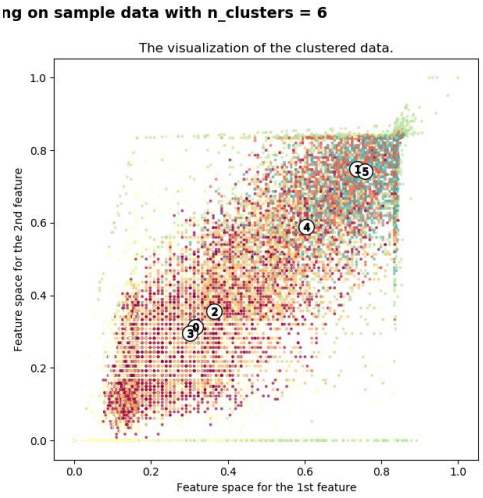
2	0.105	0.247
4	0.085	0.173
6	0.082	0.146
8	0.090	0.143

Table2 for Wine Quality dataset

Before applying any further graphic analysis, we could start think of those numbers. For the players dataset, the clustering has the best performance when giving n_clusters = 2 because the NMI is highest and Silhouette_score is highest. In this way, the clustering makes the best features selections. However, given the size and complexity of the original dataset, I doubt if the clustering did make a perfect job on the dataset. So I just run different values of n_clusters. Let's see the result in graphs.

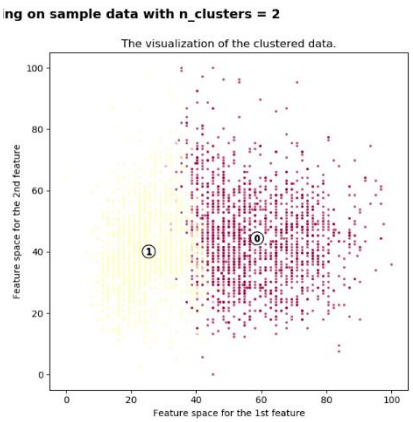


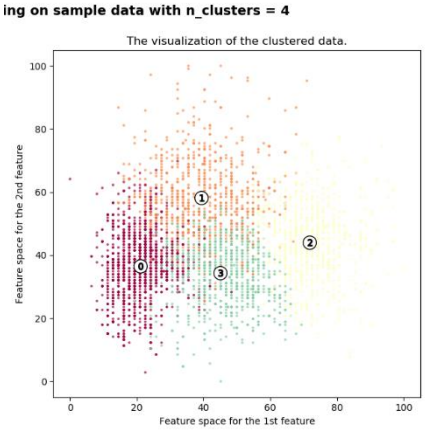


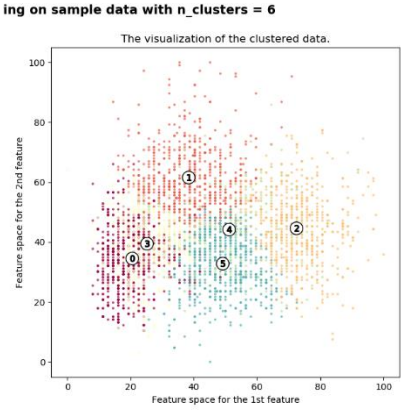


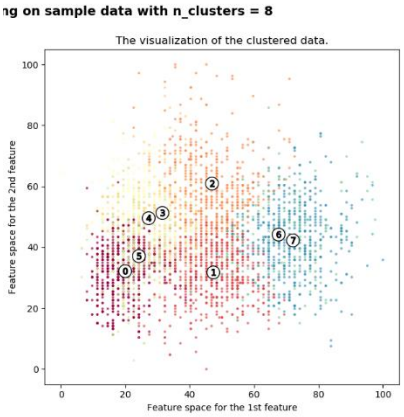
Just take a look of the three graphs. We used the different colors representing the different clusters. Ideally, what I want the clustering do is make some separation so that the same colored points mostly stay in one area with as less intersection as possible. In this way, the clustering algorithm bascially failed in n_clusters = 2 and n_clusters = 6 because ironically, the different colors are evenly distributed in the whole graph. In n_clusters = 4 graph, the green and orange points somehow is showing the separation. So, how does this happens? As we learned, the curse of dimensionality. I try to explain it in a simple way. Think of a sphere in 3-dimension, if our data distributed like the sphere, we only are able to understand them if we try to analyze them in 3-dimension. However, if we only work in 2-dimension, no matter how we approach them, we will always get the cycle.

Now, considering the low values of NMI and silhouette_score of players dataset, I am going to skip the pre-explanation of wine quality because it might just give us the same disappointed result. I will ust run the code and directly analyze the graphs.









Well, surpriseingly, the clustering did pretty good job in the wine dataset. Hence, there is one thing I need to declare first. The NMI value is not necessary to be very high in order to decide if the clustering has a good performance. So, given different values of n_clusters, the clustering always could make some feature selections so that different colors mostly stay in different area. However, if there are too much 'centers', the different colors will not stay in different area any more. Now, what makes the wine quality dataset different from the players dataset so that the clustering could have a good preformance here even without any dimension reductin?

One appearent reason is the size of features for the two datasets. Since there are only eleven features in the wine quality dataset and some of them are correlated. Based on my analysis of assignment1, there are actually only five features are related to the quality. Now, if you take a look of the graphs of n_clusters = 4 and n_clusters = 6, you will find that when the n_clusters < 5, the feature selections or feature classifications are very good. And the colors overlaping appears only we are trying to make centers mor than 5. Alse, it's necessary to mention that the correlations of those five features are pretty low which means they are quite mutually independet. That's why the clustering has such a good performance even without any dimension reduction. Furthermore, I think the dimension reduction algorithm will increase the preformance of clustering on players dataset larger then that on the wine quality dataset. And of course, the reason why the poor performance on the players dataset will just because too much features in the dataset, and low mutual independent.

2.2 Expectation maximization algorithm

2.2.1 Brief explanation: For this algorithm, generally it's very close to k-means clustering algorithm. For EM algorithm, it's just a way to find the max likelihood estimtes for model parameters. EM will iteratively approach to the optima with the maximum likelihood function. The core is Gaussian mixture model

2.2.2 GaussianMixture(): This is a useful method in Sci-kit learn which provide the Gaussian Misture model. So, a Gaussian Mixture Model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. We can think of this model as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent gaussians.

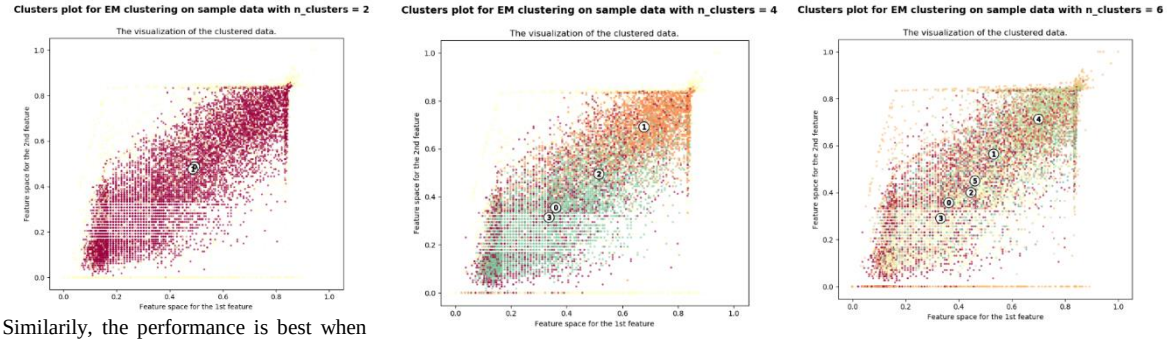
2.2.3 Pre-analysis: Before running any code to generate graphs or numbers, I want to make some pre-analysis. Though I might or highly possible be wrong, it's meaningful to make some assumption because it improves my ability of intuition to the performance of each algorithm. In my opinion, the EM algorithm will has a better performance if the dimensionality doesn't hinder the algorithm. Think about the wine quanilty dataset, if k-means clustering has a fine performance on that dataset, we should expect the EM algorithm has a better performance on the same dataset.

2.2.4 analysis: Before generating any graphs, let's calculate the NMI first.

n_clusters	NMI		n_clusters	NMI
2	0.29		2	0.049
4	0.22		4	0.056
6	0.22		6	0.048
			8	0.078

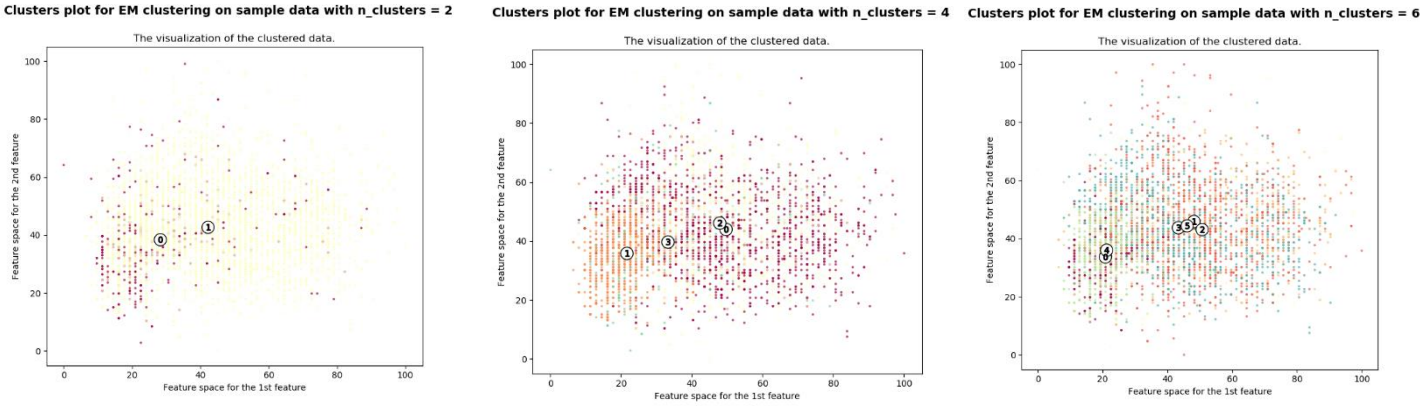
Players dataset wine quanlity dataset

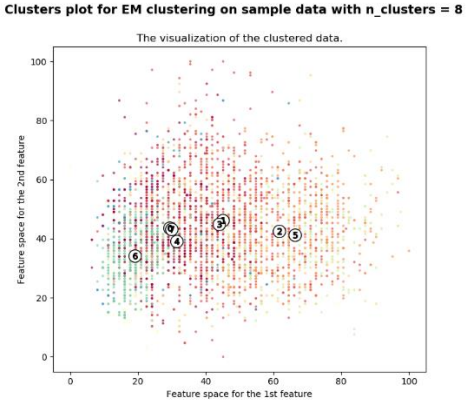
Well, I might wrong about the performance in 2.2.3. Since the average NMI of players dataset is much higher than the average NMI of wine quality dataset. But I need to get graphs before I start to look down my prediction abilities…



First I think we need to compare these three graphs with the k-means clustering algorithm's generating graphs. Similarily, the performance is best when n_clustering = 4. However, notice that there are less colors are separated in the EM generating graphs which means EM has the worse performance regarding same dataset than k-means clustering. Since EM tries to find the maximum likelihood, without any dimensionality reduction, there is one thing might happen. Say we have two mountains whose heights are close, if we observe them in front of one of mountains, then we will not be able to observe the second mountain. That's what happens now. Also, with max n_clusters' value, the result is quite ugly, actually seem as k-means clustering and the graph didn't shows any learnig effects.

Now, let's see wine quality dataset:





Comparing with k-means clustering in this dataset, the EM still has a worse performance. Knowing EM algorithm works on the Gausian Distribution, since there are less noisy features in wine quality dataset, generally the performance of this dataset is still better than the performance of players dataset.

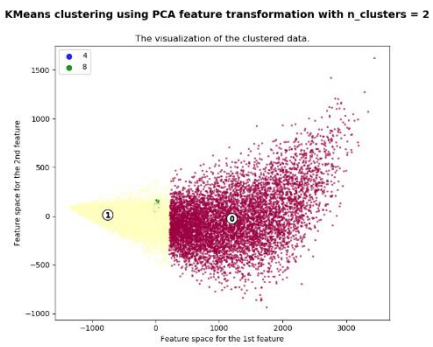
Now, I want to focus on why the overall performance of EM is worse then k-means clustering. Here we can think of the Gaussian Distribtion as the normal distribution and it always looks or the "middle high" area. However, what happens on our raw dataset is the ditribtion is quite even. More specificly, I really means the dataset "looks like" even since the dimensionality. Please recall the "mountain example" in several previous paragraphs. In this way, everything sounds make more sense and now we have a better idea of why such horrible things happen here.

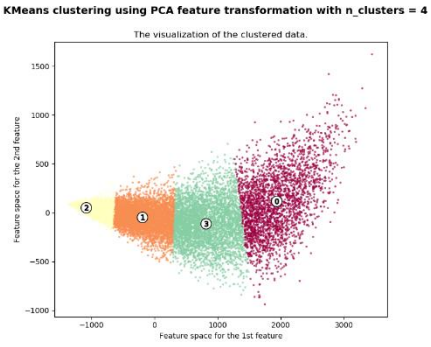
2.3 PCA

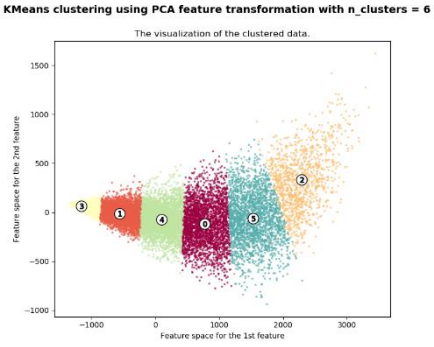
2.3.1 Brief explanation: PCA stands for Principle Component Analysis, a dimensionality reduction algorithm which applys on unsupervised learning algorithm. This algorithm is quite useful while applying to visualize high-dimensional data. In general, I could say that the main activity of PCA is reducing features. More specific, instead of making reductions directly, PCA would create some new features to stands for more old features. Then PCA would adjust the coordination system to finalize the better system.

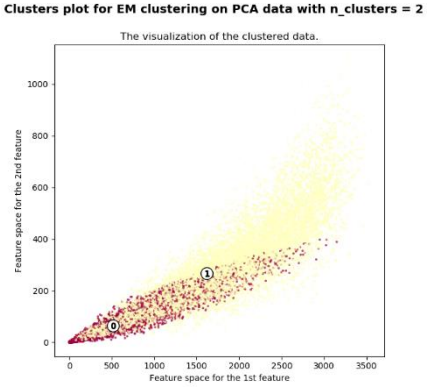
2.3.2 Pre-analysis: Since PCA would generate "new features" and adjust the dimensionality, the learning effect of the clustering algorithm should be much better than the algorithm without any dimensionality reduction. At least for the players dataset, I bet the learning effect will be increased a lot.

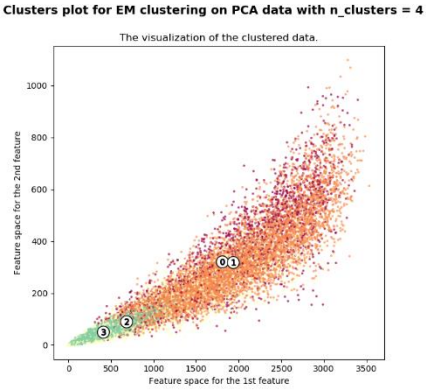
2.3.3 Analysis:

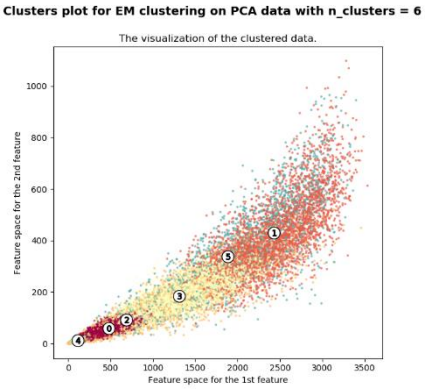






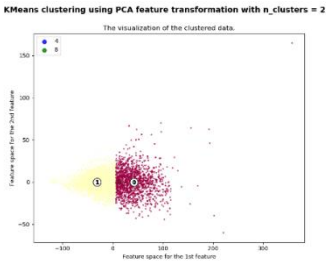


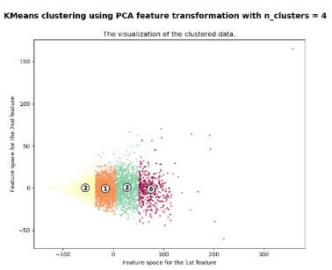


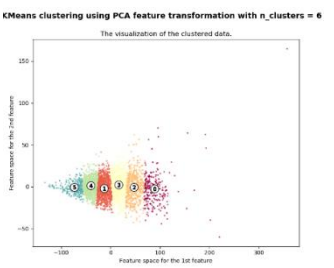


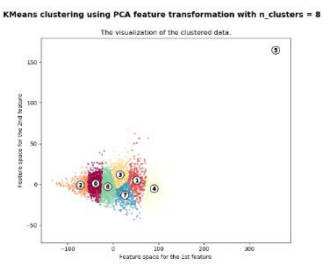
Now, let's observe the different learning result graphs. In general, PCA did exactly what we expected it do. I would use a new way to describe it: PCA essentially relocated plots by adjust te coordinations and make new features. Before running PCA, the feature separation is basically failed because all different kinds of features are wrapped together. Consider our dataset: Players dataset. This is a dataset contains 38 features, which at the same time means there are many of the features overlapping and be noisy and hindering the clustering algorithm to make the right separation. However, even though k-means clustering has a good performance, the EM algorithm doesn't perform as good as k-means. Now, I would like to analyze why k-means clustering can have a better performance while combining with PCA. Knowing PCA essentially will reconstruct the whole dataset and reduce the dimensonality as much as possible. In this case, regarding to the gaussian distribution, while the PCA working, it always find the kinda "gaussian area" and make some dimensionality reduction there. In this way, this kind of behavior will weaken the learning effect of EM algorithm.

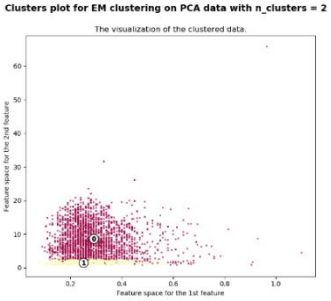
Now let's put the wine quality graphs on here: (first row is for the k-means, second row is for EM). In general, k-means clustering still has a better performance than EM algorithm, the reason we have already discussed in the previous paragraph. In this way, it's clearly more meaningful the make a comparison between two different datasets. By observation only, regarding k-means clustering algorithm, PCA almost improves k-means clustering on both datasets into the same performance level even if k-means clustering on wine quanlity dataset has an original better performance than players dataset. In this way, I think we can come up with a conclusion that PCA can always better off the learning effect regardless the original dimensional complexity.

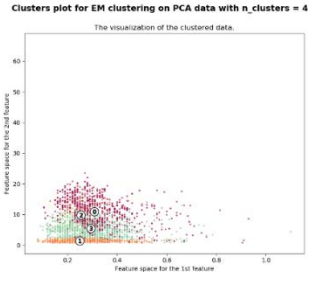


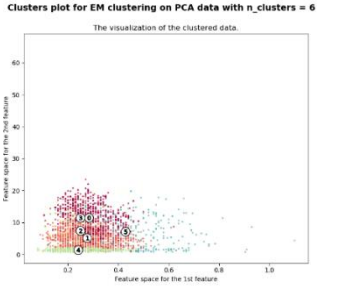


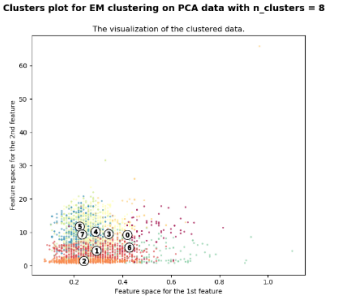












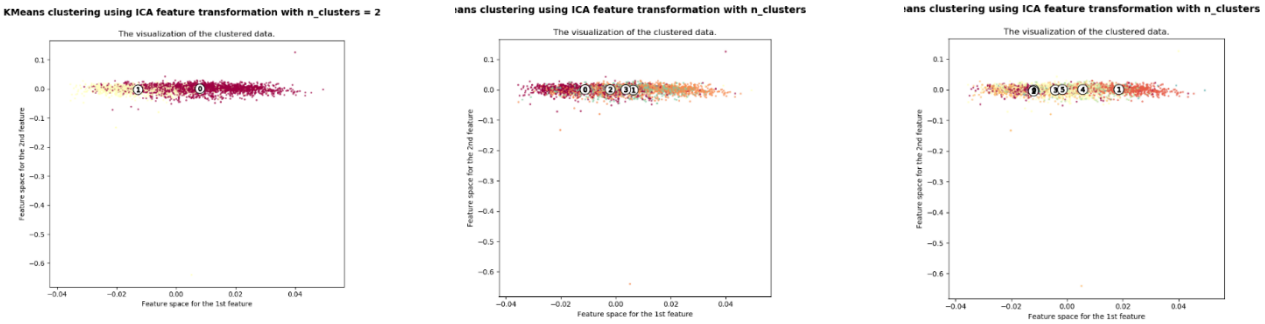
2.4 ICA

2.4.1 Brief explanation: ICA stands for Independent Component Analysis which is quite similar with PCA while ICA is about finding the independent features instead of PCA is finding the uncorrelated features.

2.4.2 Pre-analysis: Since I really don't have a picture about whether ICA could de better than PCA under what kinds of circumstances, I would skip this part here and go ahead to run the actually algorithm and see what's going on.

2.4.3 Analysis:

Let's start with wine quality. The first row, three pictures are about k-means clustering with n_clusters = 2, 4, 6.

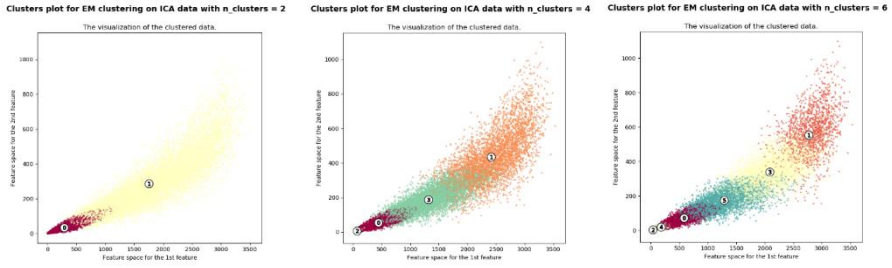


Knowing PCA doesn't better off the EM algorithm from the previous chapter, for saving time, I just try ICA on k-means clustering. What I will do is that if ICA also can better off the k-means clustering, then I could say ICA works in the same condition as the condition of PCA. If it's not, then we could just discuss the difference between ICA and PCA only use k-means clustering algorithm. Now if we go over the pictures, we definitely can tell that ICA doesn't better off the clustering and it even make the learning effect worse. So ICA would find the independent features and provide them to the algorithm.

Before we dig more, we might need to understand "independent features". Since I am using words instead of Latex so that there won't be mathematical explanation. Just think of what's gonna happen if two features are highly correlated, the plots consists by these two features will show some changing tendency instead of spread out in the plane. However, for independent, the plots consists lines which are perpendicular to each other.

Now, in wine quality dataset, if we are trying to find the independent features, we will eventually end up with getting some features that might still have some kinds of correlated relations and also might be many features while they only perpendicular to one feature. If these situations occur, the learning effect will definitely bad. By all means, the independent features might be helpful but at least up to now, we have to say the independent features might shouldn't be applied in such less featured dataset.

Now, since we know ICA cannot work well with k-means clustering algorithm, we might give it a try with EM algorithm. And this time we apply players dataset becase I've already said that Now, let's compare this with the previous EM algorithm with PCA. Clearly, when there are too many features in a dataset, ICA would do better than PCA. But how does it happen? Say if we run PCA and find those uncorrelated features, one issue that may happen is that the complexity will still be huge. However, the independent features are relatively rare which means we kinda make a double dimensionality reduction. And the dataset will look more organizaed since we shrink the complexity a lot.

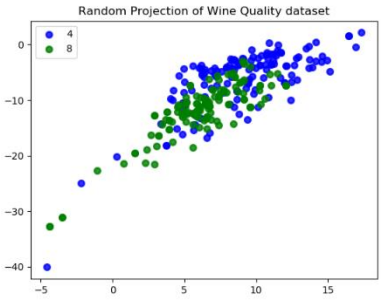


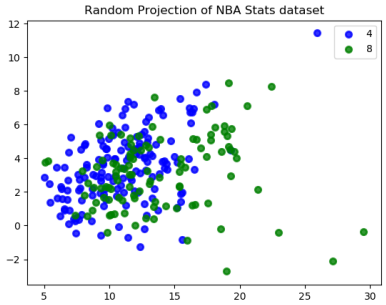
2.5 Randomized Projections

2.5.1 Brief explanation: This is a simple and computationally efficient way to reduct te dimensionality of the data. This algorithm will have the fast processing time and smaller model size but its accuracy would be influenced negatively. By Wikipedia, Random projection reduce the dimensionality of a set of points which lie in Euclidean space.

2.5.2 Analysis: Think of our dataset is a Matrix X with d columns and e rows. In this way, the dimension we have is d * e. When we apply this dimensionality reduction algorithm, the columns will shrink to a and the rows will shrink to b so that our new dimension will be a * b which is way more smaller than d * e. This algoritm generally performs better on the complex dataset whose features are huge. The reason is simple. Since this algoritm will reduce many features, this random reduction will make the learning accuracy of small dataset decrement a lot. However, regard to the large dataset, the situation will be different.

Also, I might need to mention RandomState, a parameter in sci-kit learn. Random_state will be prvoided to the random generator and will make sure the state is reproducible. From these two graphs, we can see the variance of them are quite small. If have to make a comparison, wine quanlity dataset provided a better performance. There is one reason across my mind. Since there are many features in players dataset and many of them are correlated, while doing randomized projectin, it's possible to make some wrong selection and leave those correlated features only. In this way, the learning effect will be bad and that just leads to the high variance.





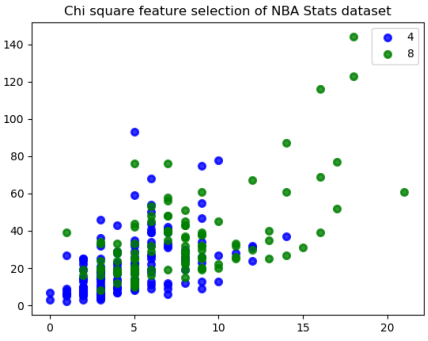
2.6 Select K-best algorithm

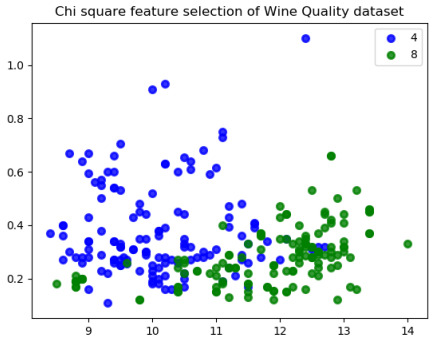
2.6.1 Brief explanation: This is a dimensionality reduction algorithm. It will select the top k features thathave maximum relevance with the target variable.

2.6.2 chi-squared distribution: A distribution of a sum of the squares of k independent standard normal random variables. We use it to test how likely it is that an observed distribution is due to chance. Also we can think of it as "goodness of fit" since it measures how well the observed distribution of data fits with the distribution.

2.6.4 Analysis:

From these two pictures, we find that wie quality has a better peformance since its variance is smaller. Think of k-best, it always picks the best features. Now, the first thing I want to discuss is overfitting. When I see the word best, we need to realize that the best features might make the training model to be too good. The other thing is about why the Players dataset has such a bad performance. Think of what will happen if most of features are almost equally good? I mean if there are not best features, then this algorithm might be very limited helpful since it will grab almost all of the features as "K best"



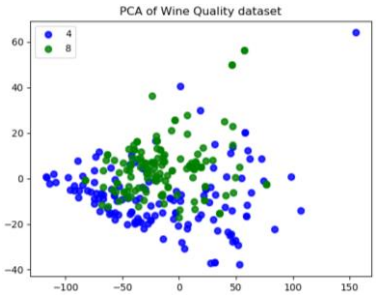


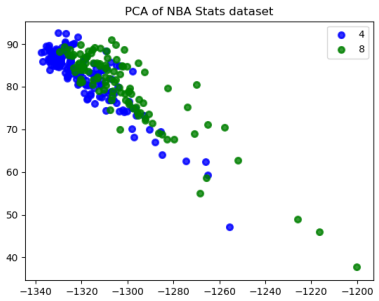
3. Analysis the data processing of PCA, ICA, and randomized projection:

In the past, we discuss the performance of unsupervised learning algorithm wihout dimensionality reduction, and the learning algorithm with dimensionality reduction. We analyzed the differences among dimensionality reduction algorithms and what kinds o situation they should be applied on. So in part three, we will focus on PCA and ICA, analyze their data processing abilities on different datasets. I are using the variance of features to support our analysis. The reason I didn't plot the data directly here is because the dataset volumn is still huge and it's hard to plot them directly. I am providing some graphs which you should be familiar with.

3.1 PCA:

Here is PCA graphs on the wine quality dataset and players dataset. The distribution of eigenvalues is more clear on Players dataset than on the wine quanlity dataset. So the eigenvalues show the change of features. While the eigenvalues are high, just like players dataset(NBA), it tells us that PCA does more work and feature transformation on this dataset. The reason that this kind situation happened on players dataset is because the players dataset is a hugs dataset, PCA will make many dimensionaliy reduction. Just think of what will happen if there are five features lie on the line x = y and the Euclidean length of the feature distributing on the x = y is very long. In this situation, the PCA will generate new features and also have a very large eigenvalue. Now back to the wine quality dataset. We've already know there are eleven features and five of them are uncorrelated. It's not hard to understand why the variance of eigenvalues are low and the distribtion of eignevalues are mostly concentrating around the original point(0.0)

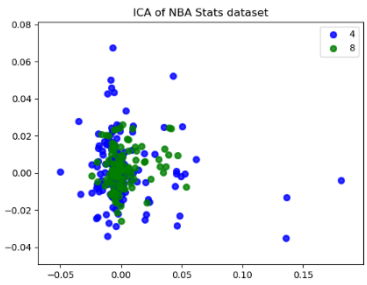


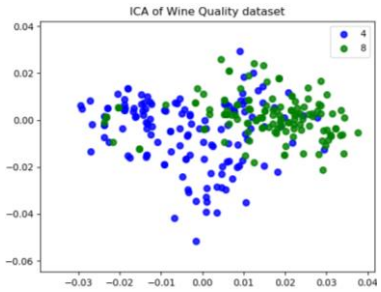


In general, the variance of PCA is not high by my observation during several times re-running. This situation might also related to the eigenvalue. If we see the data distribution of the data space as vector set, there will be several eigenspace. PCA actually do the feature transformation based on the eigenspace. Understanding that, we will know the features are quite close that each time PCA generates.

3.2 ICA:

Here are ICA graphs on both dataset. ICA has a better kurtotic distribution while ICA being applied on the players dataset. Since ICA is looking for the independent features, the better the kurtotic distributin, the features found by ICA will be more independent. The introduction page on canvas brought an interesting question: Do the projection axes for ICA seem to capture anyting "meaningful"? First we need to figure out what is "meaningful"? I think it means the features that can represent the dataset will be meaningful. So in this way, the features are independent doesn't make them meaningful because I could just add two irrelevant features into the dataset and make sure they are independent. Then if I run ICA and find them, does that help clustering? No, not at all. This is my opinion regarding ICA. Though the variance during several re-running is comparatively high. Based on the different processing orders, the features the ICA found would be different.





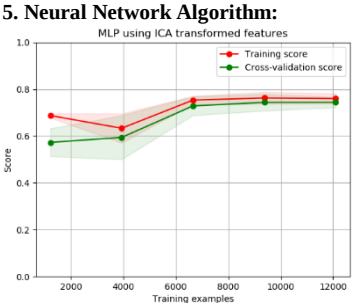
3.3 Randomized projection:

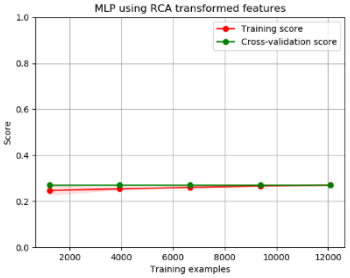
This dimensionality reduction's graphs have already been shown in the previous part. While re-running RP several times, the variance is not high unsurprisingly. Recall we set the random_state when applying RP in sci-kit learn. This is just make sure we the generator will return the same result given the specific input. Since the introduction page on Canvas didn't mention how to tune random_state, I also try different values of this hyperparameter. Under this situation, the variance is quite high.

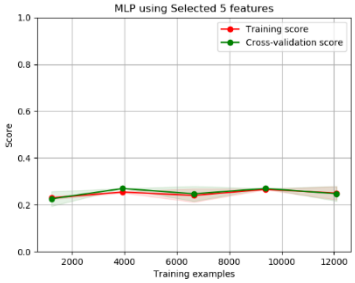
4. Reproduce clustering algorithms:

Have already done that in part two. I basically do the analysis, and dimensionlity reduction at the same time on part two.

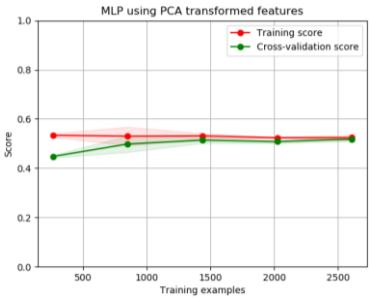
5. Neural Network Algorithm:

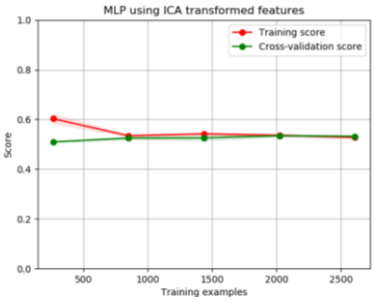


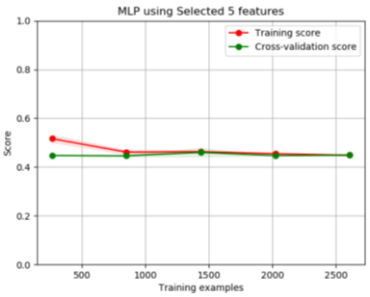




Players dataset

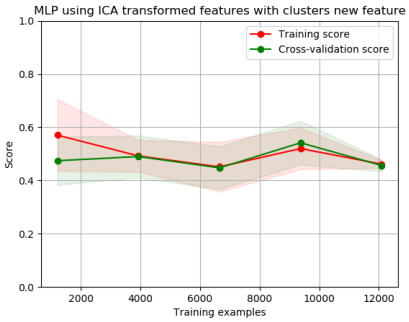


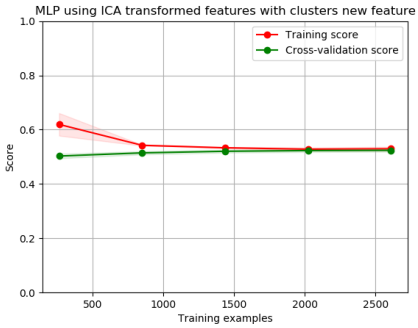




Wine Quality dataset

Before running Neural Network learner with clustering algorithm, I run the neural network algorithm with the dimensionality reduced dataset. Compare with the learning curve from assignment1 and assignment2, the score is very similar. This means that the neural network algorithm provides a same performance with or without the dimensionality reduction. Then, let's see what will happen if we process the data using clustering first: (left is for players dataset, right is for wine quality dataset)





Bascially, the performance are same. The clustering and dimensionality redution didn't improve the learning curve score.