Assignment 3

Datasets –

Classifying the Quality of Wines based on its Physiochemical properties

This dataset from UCI ML Repository has 4500+ data points on the Portuguese Vinho Verde wine. Each data point has 11 physiochemical features (Citric acid, Chlorides, pH, SO2, etc.) which will be used to determine if the Wine is of high quality.

I found this fascinating because *nowhere in this dataset does it mention Grape type, Wine brand or Wine Selling Price*. These 3 features would probably be highly ranked by a normal Wine appreciator. To build a model without these features that might be able to predict the quality was very compelling to me.

The data was slightly skewed with 66.5% of instances being "Good wine" and 33.5% being "Bad".

Classifying the Gender of Abalones based on Physical properties

This dataset is also from UCI ML Repository and has over 2,800 data points. Each data point has 8 physical features (length, diameter, height, weight, etc.) which will be used to determine if the Abalone is Male or Female.

Predicting gender based on just a few physical characteristics is hard and that fascinates me.

The data is almost evenly split with 54% Male points and 46% Female points.

K-Means –

This is one of the best known and yet simplest Unsupervised algorithms. It can be used for a variety of applications including topic modeling, classification and anomality detection. It is generally fast and efficient for small values of K (as in our dataset). Downsides though are that the clusters produced are spherical. This affects scenarios where input data may have different cluster shape.

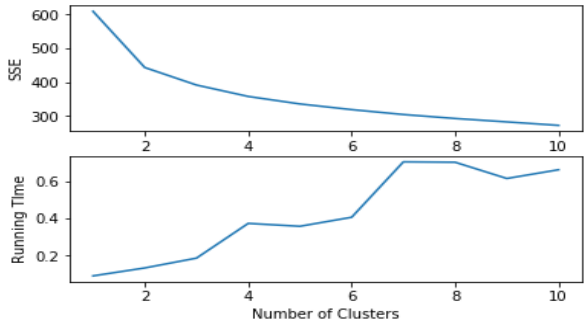
We use 2-methods to identify optimal cluster # -

1. **Elbow method** – this method plots the number of clusters against a metric (usually SSE or AIC or BIC). This way we can easily identify diminishing marginal returns. After some points (say 3 clusters) the metric won't go down all that much for subsequent clusters
2. **Silhouette Coefficient** – this displays how close each data points in a cluster is to the data points in the most neighboring cluster. Meaning we should select a Cluster where the silhouette coefficient is high

Wine Data –

After parameter tuning the k-value, we see that the elbow is around 4 clusters. While our actual dataset only has 2 classes (good or bad), this makes sense because the actual rating for the wines were numerical (between 1 and 10). We converted it into categorical by saying if the rating is over 5 then the wine is good, else bad. So, this is a regression problem that I converted into a classification one because of my interest in the dataset. From the bottom visual, we see as we noted earlier that higher number of clusters correlate with higher running time. It is interesting though that the silhouette coefficient is the highest around 2 and fall steadily after that.

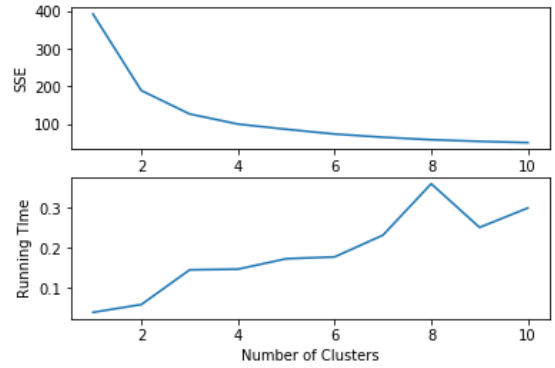
```
For n_clusters=2, The Silhouette Coefficient is 0.24619145662552291
For n_clusters=3, The Silhouette Coefficient is 0.1920098321984321
For n_clusters=4, The Silhouette Coefficient is 0.17089588913911696
For n_clusters=5, The Silhouette Coefficient is 0.16857005209355336
For n_clusters=6, The Silhouette Coefficient is 0.14416732636858587
For n_clusters=7, The Silhouette Coefficient is 0.14336640180799867
For n_clusters=8, The Silhouette Coefficient is 0.13931575000297736
For n_clusters=9, The Silhouette Coefficient is 0.13251723081081596
For n_clusters=10, The Silhouette Coefficient is 0.13752204223412343
```



Abalone Data –

After parameter tuning the k-value, we see that the elbow is around 2 clusters (though even 4 could be considered correct). While our actual dataset only has 2 classes (Male or Female). This could because of the data too. The actual dataset uses physical properties including sex to predict the number of rings on an abalone. We reversed that, where we are using all numerical physical properties including rings to predict gender. Like the above scenario, the silhouette coefficient is highest for k=2 (it picks back up around 8, but not as high).

```
For n_clusters=2, The Silhouette Coefficient is 0.4265455294723743
For n_clusters=3, The Silhouette Coefficient is 0.37226918888087324
For n_clusters=4, The Silhouette Coefficient is 0.3291521934350927
For n_clusters=5, The Silhouette Coefficient is 0.29751689597213826
For n_clusters=6, The Silhouette Coefficient is 0.3174548458548584
For n_clusters=7, The Silhouette Coefficient is 0.30420458708044623
For n_clusters=8, The Silhouette Coefficient is 0.31172791116891546
For n_clusters=9, The Silhouette Coefficient is 0.27977855001807256
For n_clusters=10, The Silhouette Coefficient is 0.2725295168439207
```

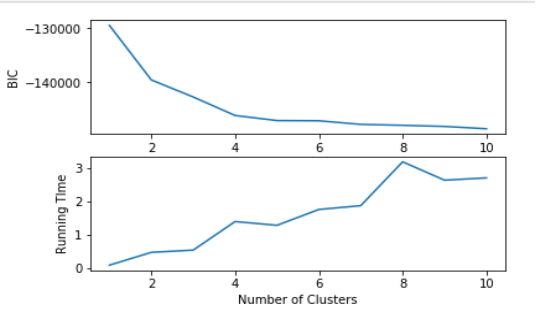


Expectation Maximization (GMM) –

Unlike K-means which hard assigns a data points to a cluster, EM assigns the points to a cluster softly. Meaning it gives a probability of a point associated with any centroid. We will be using Gaussian Mixture Modeling. The EM consists of 2 steps, the expectation step where each object is assigned to the most likely centroid and the maximization step where the centroids are recomputed based on the points associated to it. So, in GMM, in addition to centroids (which is the mean), we will use variance and covariance to assign points to clusters. Also unlike K-means GMM is not restricted to spherical shape.

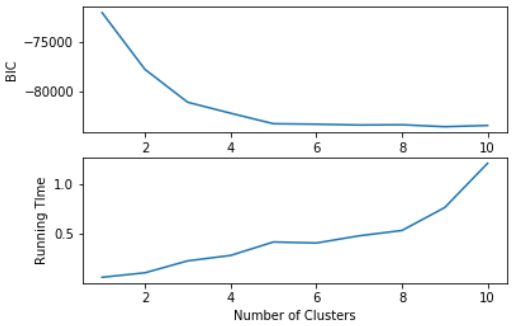
Wine Data –

Like K-means we see that optimal cluster is around 4. What stands out to me though is the running time. While even at 10 clusters K-means took around 0.3 seconds, GMM took 10 times longer. Unlike K-means where we used Sum of Squared errors, we will be using BIC (Bayesian Information Criteria).



Abalone Data –

Unlike K-Means where the number was closer to 4 clusters, here GMM predicts around 5. GMM takes longer to run for this data set too, compared to K-Means run time. Transforming the data might help get closer to the actual classifications.

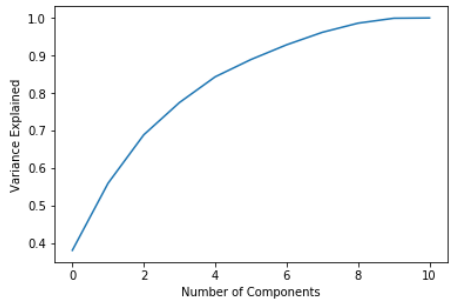


Principal Component Analysis –

PCA is a dimensionality reduction algorithm that linearly combines observed predictors into principal components. PCA is a unsupervised methods where the knowledge of the target variable has no effect on the component formation. PCA might improve linear regression problems. **But if we are using PCA to feed a neural network, we might be doing a redundant job as a NN probably might calculate the PCA itself.** Also, if the data is not linearly dependent, this might not work.

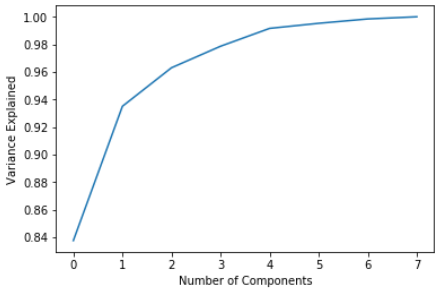
Wine Data –

Since PCA identifies components based on the maximum variance of the data points, we can check if a reduced number of components (reducing dimensionality) can reconstruct or explain as much variance as the actual unaltered points. In our case the first 6 principal components explain close to 90% of the variance.



Abalone Data –

The PCA effect is more shocking here as the first 2 principal components itself explain close to 96% if the variance in the data. Just the 1st principal component explains close to 94% variance.



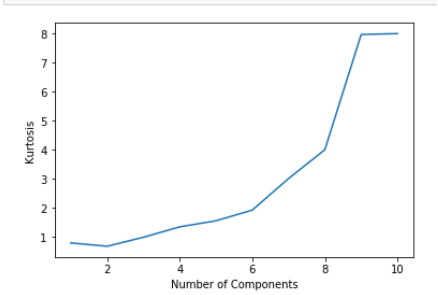
Independent Component Analysis –

This is generally used for blind source separation problems. The model sees the variables as a linear or non-linear mixture of some unknown variables. The unknown variables are assumed to be non-gaussian and mutually independent of each other. The difference between PCA and ICA is, PCA is used when we want to reduce the rank of the data represented, while ICA is used to find independent sub-elements.

Here we will be using Kurtosis to find the optimal number of components. If the kurtosis value is close to 0, then our components are gaussian. We need to find components which are non-gaussian (meaning kurtosis value > 0).

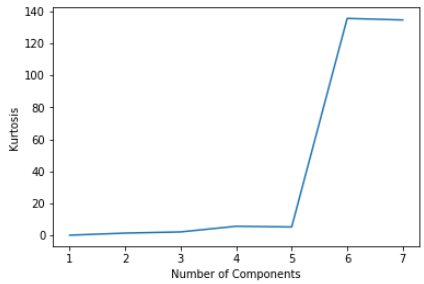
Wine Data –

For our wine data, our Kurtosis value increases are we add more components. Based on this graph, I would generally pick around the first 8 or 9 components.



Abalone Data –

Unlike our wine dataset, kurtosis of abalone dataset jumps at just 5 components –



Random Projection –

Random Projection is differing from PCA where PCA sorts and choose the components based on the direction of maximum variance, random projection finds the direction of projection randomly. The advantage of RP is that it is faster than PCA with the disadvantage of a small loss in accuracy. RP is particularly useful when –

- When our dataset is huge (in both row and column terms)
- When the dataset if sparse
- When the dataset doesn't conform to linear subspace

RP is also known to preserve the pairwise distance at relatively high probability.

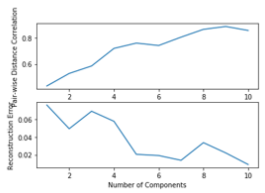
So, we will be using two metrics to find how well Random Projection is performing –

1. Pairwise distance correlation
2. Reconstruction error

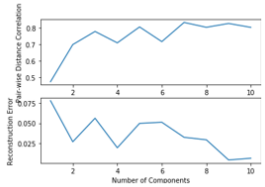
Wine Data –

As can be seen, at around 7 components, the reconstruction error is lower than 2% with high pairwise distance correlation**. Even running the algorithm with different seeds (to check if starting the algorithm under different conditions moves the graph drastically) show that the numbers are the same**.

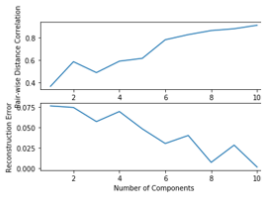
Seed = 5



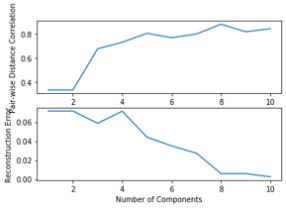
Seed = 30



Seed = 45



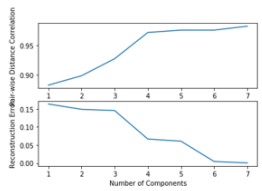
Seed = 60



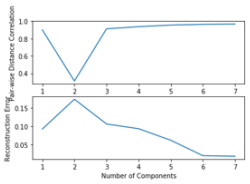
Abalone Data –

Compared to wine data it takes bit more components to reconstruct the data with under 5% error. On average it has a good reconstruction rate by 6 components.

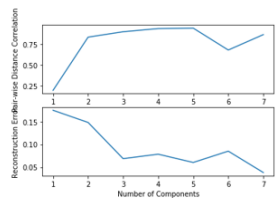
Seed = 5



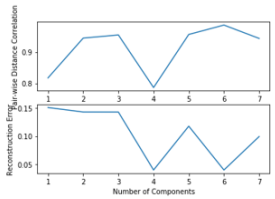
Seed = 30



Seed = 45



ed = 60



Random Forest –

Random Forest is among the most popular ML method. This is because of their good robustness, accuracy, and ease of use. It provides 2 methods for feature selections –

1. Mean Decrease Impurity – which calculates feature importance on how much entropy it decreases when in a tree. We will be using this method
2. Mean Decrease Accuracy – which calculates feature importance by permuting the value of each feature and measuring how much the permutation decreases the accuracy of the model

Wine Data –

Using Random Forest, we get the features importance as –

Feature	Importance
alcohol	15.0%
density	11.7%
volatile acidity	11.4%
free sulfur dioxide	9.6%
chlorides	8.4%
total sulfur dioxide	8.3%
residual sugar	8.0%
citric acid	7.5%
pH	7.3%
fixed acidity	6.4%
sulphates	6.4%

Abalone Data –

Feature	Importance
shuckedweight	16.2%
viscera weight	15.4%
wholeweight	15.1%
shellweight	14.2%
length	11.1%
diameter	10.6%
height	9.6%
rings	7.8%

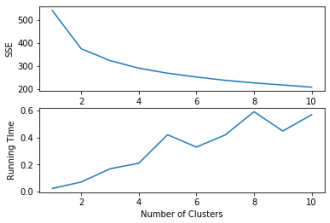
K-Means after PCA –

Now we perform K-means on the PCA transformed data. We see some improvement in the Silhouette coefficient after PCA, but the story remains the same.

Wine Data –

```
For n_clusters=2, The Silhouette Coefficient is 0.27668586812253854
For n_clusters=3, The Silhouette Coefficient is 0.223917194264677 27
For n_clusters=4, The Silhouette Coefficient is 0.201905338774565 76
For n_clusters=5, The Silhouette Coefficient is 0.202688950364407 23
For n_clusters=6, The Silhouette Coefficient is 0.16648618022681 08
For n_clusters=7, The Silhouette Coefficient is 0.169672937844440 54
For n_clusters=8, The Silhouette Coefficient is 0.170048200826744 63
For n_clusters=9, The Silhouette Coefficient is 0.160001573071857 28
For n_clusters=10, The Silhouette Coefficient is 0.15841869360685093

<module 'matplotlib.pyplot' from 'C:\\ProgramData\\Anaconda3\\lib\\sit
```

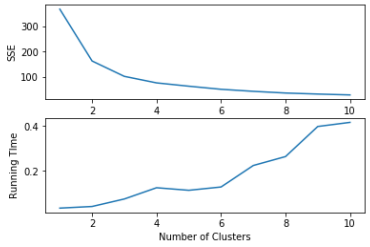


Abalone Data –

Similarly, the Silhouette coefficient sees improvement in this case too after PCA. The story is a little better, where we the elbow is now either 3 or 2 clusters.

```
For n_clusters=2, The Silhouette Coefficient is 0.46571443365652637
For n_clusters=3, The Silhouette Coefficient is 0.41804463130860686
For n_clusters=4, The Silhouette Coefficient is 0.38525761059028557
For n_clusters=5, The Silhouette Coefficient is 0.4015121268488733
For n_clusters=6, The Silhouette Coefficient is 0.3829685457497266
For n_clusters=7, The Silhouette Coefficient is 0.377082833570374
For n_clusters=8, The Silhouette Coefficient is 0.39692058517187045
For n_clusters=9, The Silhouette Coefficient is 0.37344394108144807
For n_clusters=10, The Silhouette Coefficient is 0.37079170258515814

<module 'matplotlib.pyplot' from 'C:\\ProgramData\\Anaconda3\\lib\\si
```

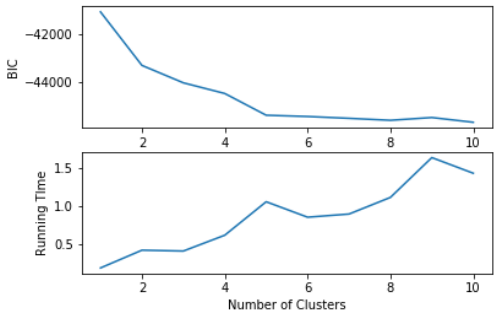


GMM after PCA –

Let's see if GMM (since it is not constrained to spherical cluster shape) perform better.

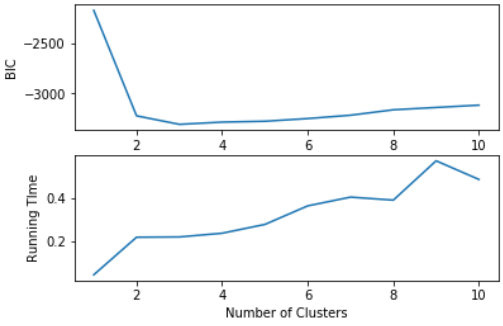
Wine Data –

BIC sees a big drop after using PCA (it was in the -130K range without PCA)



Abalone Data –

We are seeing the elbow at number of clusters = 2. This is equivalent to the actual classification.

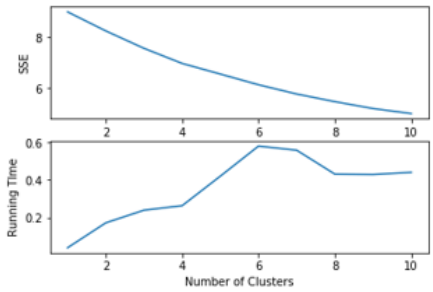


K Means & GMM after ICA –

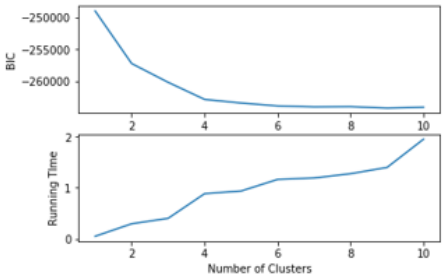
Since we are not trying to separate any blind source problem here, ICA doesn't perform as great compared to the other feature transformation algorithms.

Wine Data –

K-Means –

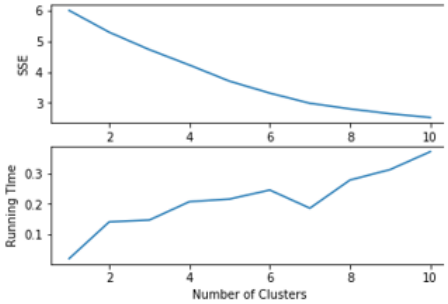


GMM –

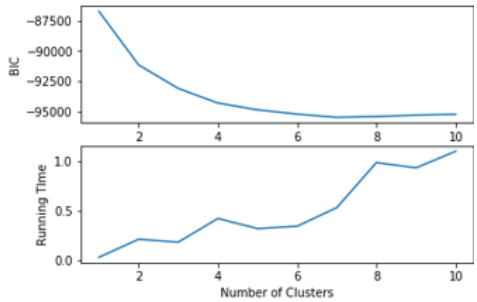


Abalone Data –

K-Means –



GMM –

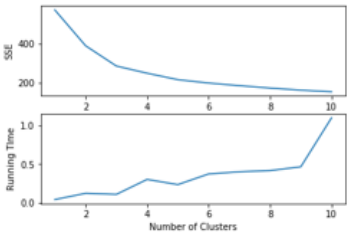


K Means & GMM after Random Projection –

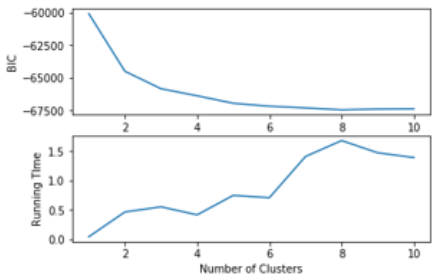
Neither K Means or GMM perform as good with RP as they did with PCA.

Wine Data –

K means –

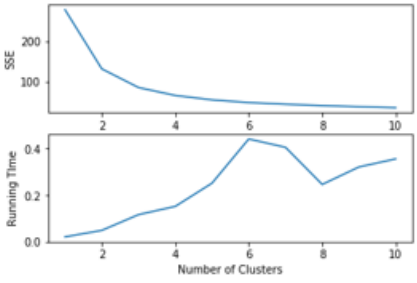


GMM –

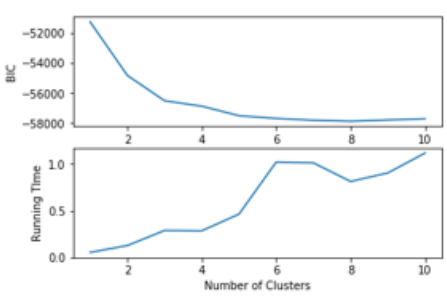


Abalone Data –

K means –



GMM –



MLP after PCA –

Wine Data –

When we ran MLP the last time for Assignment #1 on the Wines dataset, we got –

Validation Accuracy = 0.773

Validation Sensitivity = 0.595

Validation Specificity = 0.867

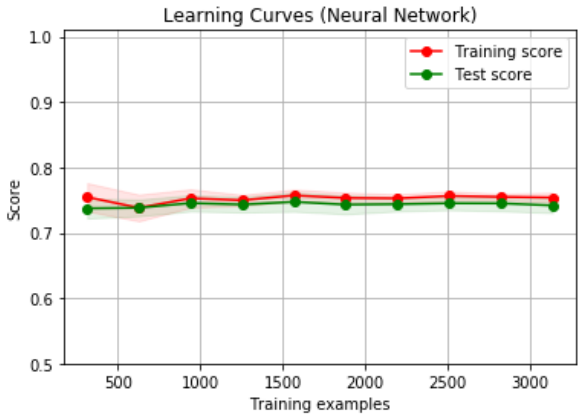
Now when we run the MLP (and hyper parameter tune) after passing the data through feature transformation, we get –

Validation Accuracy = 0.756

Validation Sensitivity = 0.617

Validation Specificity = 0.825

I did mention in Assignment #1 that identifying Bad wines might be more important as a business than identifying Good wine. With that in mind, our sensitivity does perform better after PCA. But not by that much. Given that PCA takes more time and is slightly less accurate overall, I don't see much benefits from PCA in this case. The learning curve also shows that the train and validation scores are almost the same – meaning the model is a good fit.



MLP after K-Mean Clustered and PCA Transformed –

Here we use the PCA transformed features on K-Means to get the clusters for each data point. We will use this as a feature into the MLP algorithm.

Wine Data –

Using the new cluster information from K-Means, our MLP performs even better in finding bad wines. Thus, we can say adding the clustering information does add value to our MLP. As for the number of Clusters, I chose 2 since I know this problem has 2 classes. In terms of accuracy. Here is the confusion matrix –

Actual/Prediction	No	Yes	Grand Total
No	359	1881	2240
Yes	1281	1377	2658
Grand Total	**1640**	**3258**	**4898**

Can we can see, the clustering accuracy is very low – around 35.5%. That is almost as bad a random guess for this skewed dataset.

Validation Accuracy = 0.773

Validation Sensitivity = 0.626

Validation Specificity = 0.836

Abalone Data –

Interestingly, though the data is evenly split into Male and Female we see that the model has a harder time accurately predicting Female. Like the above problem, we used K-Means with 2 Clusters.

The clustering accuracy is around 53%. Seeing that the MLP accuracy is around 54.1% I can be pretty sure that the clustering class contributed heavily to the final output. Here is the Clustering Confucian matrix –

Actual/Prediction	F	M	Grand Total
F	777	809	1586
M	530	719	1249
Grand Total	**1307**	**1528**	**2835**

Validation Accuracy = 0.541

Validation Sensitivity = 0.705

Validation Specificity = 0.334

Few other things we can try –

1. Use **Mahalanobis distance** instead of Euclidian distance. Euclidian distance is particularly well suited for spherical clusters. If the cluster shape is something like elliptical, Mahalanobis performs better
2. Check if the MLP with cluster information and un-transformed data perform better (as PCA seem to reduce the algorithm performance a little). This could be because, like we mentioned before Multi-layer perceptrons themselves perform some form of component analysis when training the model. Feeding it a PCA is probably unnecessary or redundant. The transformation of the data by PCA (centering and scaling) may improve the convergence time for NN. But at least in this case we don't see any improvements to the classification itself.