Unsupervised Learning and Dimensionality Reduction

Jun Wang
CS7641 – Assignment 3
jwang3316@gatech.edu

Abstract

In this assignment, firstly, I implemented two clustering algorithms (k-means clustering and Expectation Maximization) and four Dimensionality Reduction algorithms (PCA, ICA, Randomized Projections, and SelectFromModel which I desired to use) on two different classification datasets with different properties and then optimized them. Secondly, I applied the Neural Network to the results and analyzed them.

1. Introduction to Datasets

Two different datasets are used in this assignment as other two previous assignments (Supervised Learning and Randomized Optimization), the UFC dataset from Kaggle and the Wine dataset from OpenML.

1.1. The UFC Fight Dataset

The UFC Fight Dataset.contains fights of each fighter's round by round record since 2013. Totally, 3592 entries and 156 features. In each game, there are 3 important features, 2 of them introduce some detail about the two fighters (red and blue) and 1 feature represents that whether the match will decide a title. The target is the **winner** of each game. For a more powerful prediction model, I removed the *number_of_rounds*, which display the rounds of game.

- Features: 156, Some attributes are continuous and some are binary (OneHotEncoded).
- Classification: binary.
- Entry: 3592.

Because of the property of UFC corner, BLUE winners' prediction is more important. f1_score (macro) is used in the UFC Fight dataset. It calculates the average value of the RED class and BLUE class. So, the class with less representative data (BLUE), gets more importance due to the results that 66% of the winners are RED because the imbalance that almost all the popular fighters are arranged at RED corner

1.2. The Wine Dataset

The second dataset is the Wine Dataset from OpenML.org datasets [3], which introduces the wines, three different cultivars produced from one region in Italy. The analysis determined the quantities of 13 chemical constituents found in each of the three types of wines.

- Features: 13, continuous
- Classification: 3-class classification
- Entry: select the first 5000 entries of all the 1 million entries of the dataset due to the speed and time of the process of classification and optimization. What's more, finding out the differences among different classifiers is also important.

We use the algorithms to predict the right cultivar of each wine based on those 13 features. Unlike the UFC dataset, this one is not imbalanced. f1_score(weighted) is used in this dataset. It calculates the weighted average value of all 3 classes according to data count of each class.

1.3. Difference

So, there are some differences between the two datasets.

Dataset	balanced	features	Classification	predict
UFC	no	156	Binary	Difficult
Wine	yes	13	3-class	easy

2. Clustering Algorithms

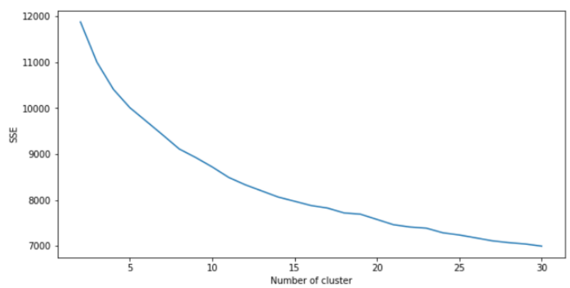
2.1. The UFC Fight Dataset

The UFC Dataset is a pretty complicated one and it gave poor results with simpler algorithms in the first assignment. So that, my expectations on clustering algorithms were low in the beginning. Discovering the classes by clustering seemed to be a long shot.

2.1.1 K-Means Clustering

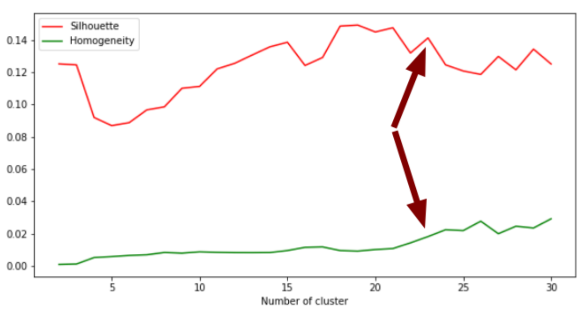
Since the dataset consists of so many floating-point data fields, I have directly used Euclidean distance to make the clustering.

First, I have tried to come up with the best k on clustering using Error Sum of Squares (SSE). I have tried k=2 to 30 and got this graph.



As it can be seen, it is difficult to find out an elbow point so that I have proceeded to other methods.

Next, I have used Silhouette and Homogeneity Scores



The homogeneity score increased as expected. On the other hand, the silhouette score reached some peaks that infroms a good clustering. It can be seen between 18 and 21 seems to be good for Silhouette scores. But k=23 is better for the combination of Silhouette and Homogeneity scores. So I have chosen k=23.

This dataset includes information about two different fighters in a row, mainly. So, it can be expected to have clusters that describes the fighters with different styles of fighting and some types might have advantage on others. So, n=23 seems to be a good number of clusters since it includes both fighters and some information regarding the match (such as if it is a title-decider or not).

Comparing Clusters with Classes

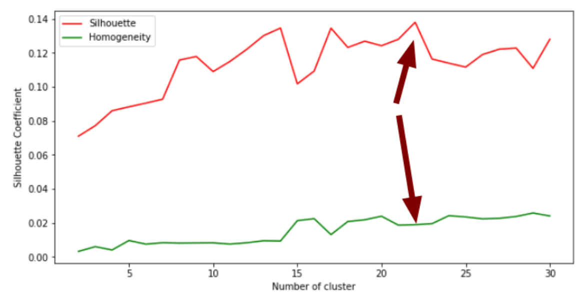
In this section, I compared 2 clusters with K-Means with classes because I needed to examine if they correspond to the classes of the dataset.

The accuracy of score of correspondence between the K-Means labels and the real classes was only 0.57. This indicates that these clusters didn't make good representations on the classes. In short, The complexity of the dataset (over 100 columns) was the reason.

2.1.2 EM Algorithm

The EM algorithm might result in different numbers of clusters because of the Gaussian Mixtures' variances. It can capture data in different shapes so it is expected to have a different optimal number of clusters comparing K-Means algorithm.

The SSE doesn't have influence on the results (they have different covariance matrices). So, I have directly used the used Silhouette and Homogeneity Scores on EM Algorithm.



k=22 seems to be a candidate on EM Algorithm because there is a peak on Silhouette score on 22 and its homogeneity score increased slightly. Interestingly, for the number of clusters, both EM and K-Means come up with so similar results. Even the Silhouette scores are close to 0.14 on both. I believe Since EM gives labels in the end (according to instances probabilities), it brought similar results with K-Means.

Comparing Clusters with Classes

In this section, I compared 2 clusters with EM with classes because I needed to examine if they correspond to the classes of the dataset.

The accuracy score representing the correspondence between the EM labels and the classes was only 0.5019. This indicates that the EM labels didn't fit, too. Its accuracy is lower but this is probably because of the randomness.

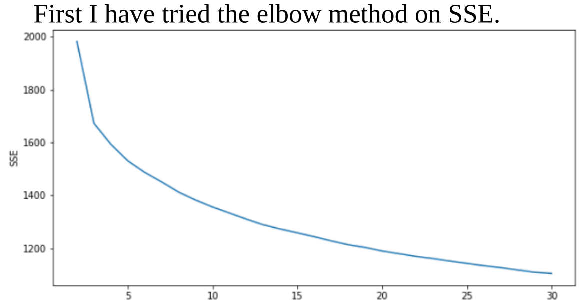
2.2. The Wine Dataset

Comparing to the UFC Dataset, the Wine Dataset was easier to discover using simpler algorithms. So, I have higher expectation from clustering algorithms in the beginning such as discovering the classes.

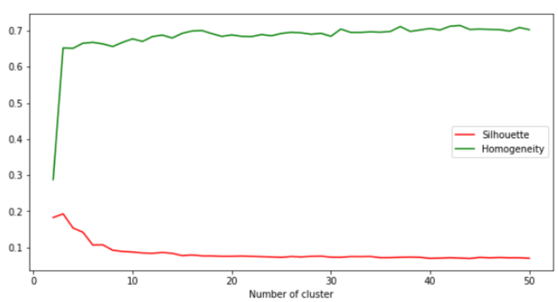
2.2.1 K-Means Clustering

The Wine Dataset consists of floating-point fields so I have used Euclidean distance metric on this dataset, as well.

First I have tried the elbow method on SSE.



Again this method was not so helpful e on deciding the number of clusters but k=3 seems to be a candidate so I have used Silhouette and Homogeneity Scores for more help.



This graph also points at k=3, clearly. So, I have used k=3 on K-Means.

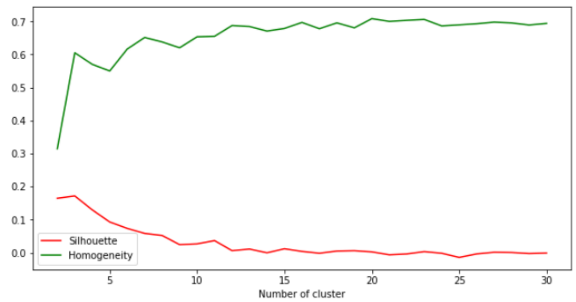
Comparing Clusters with Classes

The wine dataset has 3 classes and K-Means also got 3 clusters as optimum. When I examined the correspondence between the clusters and labels, it got a very interesting 0.9034 accuracy, which means the clusters captured the label, amazingly almost as good as a supervised learning algorithm.

I think this is mainly because the dataset consists of only 13 columns and probably all of them affects the labels closely to each other.

2.2.2 EM Algorithm

The SSE doesn't have influence on the results (they have different covariance matrices). So, I have directly used the used Silhouette and Homogeneity Scores on EM Algorithm



This graph also points out k=3, which was expected after the K-Means has an optimum k=3. This means the dataset is distributed into three different clusters.

Comparing Clusters with Classes

Unexpectedly, the EM didn't give a good result when compared with labels even though its Homogeneity score was almost as high as K-Means. It captured only 0.5648 of the labels correctly and when analyzed manually, I have realized that one of the labels are detected quite good but two others are not and this is probably because EM has more flexibility and it captured some different clusters that doesn't fit the labels.

3. Dimensionality Reduction Algorithms

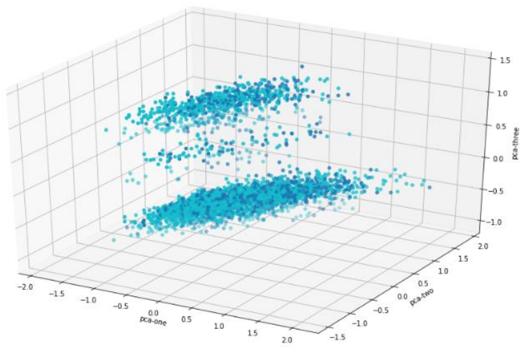
3.1. PCA Algorithm

3.1.1. UFC Dataset

UFC Dataset has 156 features, so a dimensionality reduction could get rid of a great deal of them and still capture most of the information in the dataset.

I have tried from 2 to 150 dimensions on PCA and captured the reconstruction errors along with the explained variance ratio. PCA could get 0.9 of the explained variances with 30 dimensions and 0.98 of it using 72 dimensions. I have used the 72 dimensions, which reduces the feature numbers by half and capturing 98% of the explained variance.

.

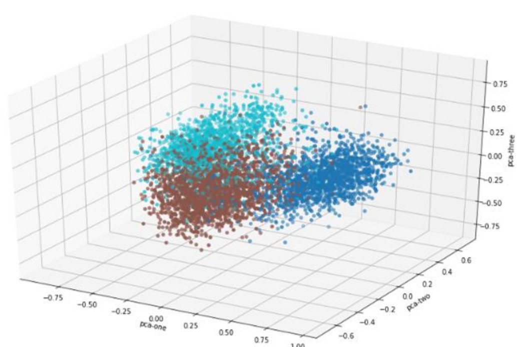


When drawn first 3 components of PCA, it can be seen that the points are not separated in 3 dimensions

3.1.2. Wine Dataset

The Wine Dataset has only 13 features and I have tried from 2 to 13 dimensions in PCA and captured the reconstruction errors along with the explained variance ratio.

8 dimensions could get the 0.93 of the explained variances ratio and it uses almost half as many features.

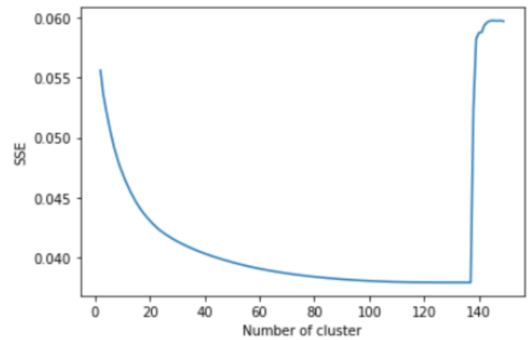


As it can be seen, the first three components of the PCA makes a quite good separation among the labels.

3.2. ICA Algorithm

3.2.1. UFC Dataset

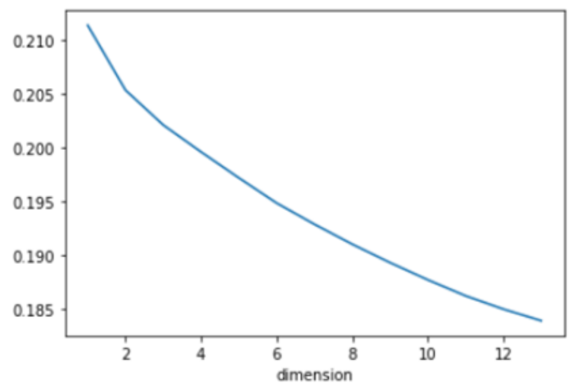
For ICA algorithm, I have used the reconstruction error to determine on the number of dimensions. After testing 1 to 150 dimensions on UFC Dataset, I got this graph of reconstruction error.



I have chosen 80 components because it gets flat after there and it uses quite less features comparing to the original dataset.

3.2.2. Wine Dataset

I have checked 1 to 13 (all) features on Wine Dataset and got this graph on reconstruction error.



It seems like a linear line that goes down but 3 seems like a good candidate and the reconstruction error seems like not changing a lot. So, I wanted to try 3 dimensions to see the ability of ICA when it comes to compress the information to such a low number of dimensions.

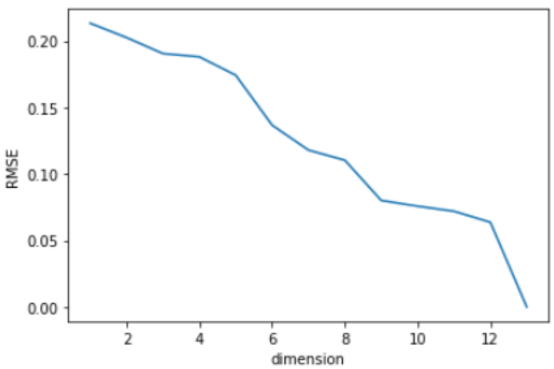
3.3. Randomized Projection

3.3.1. UFC Dataset

The reconstruction error of randomized projection was quite linear. So, I made an arbitrary choice of 40 dimensions on this dataset.

3.3.2. Wine Dataset

I have checked 1 to 13 (all) features on Wine Dataset and got this graph on reconstruction error.



After 9 dimensions, the RMSE doesn't drop down until 13, so I have chosen 9 dimensions on randomized projection.

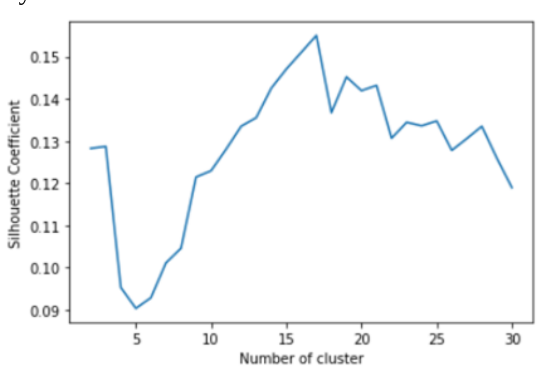
3.4. Feature Selection

As last, I have used SelectKBest from the model and selected the best 40 from UFC Dataset and only 5 from Wine Dataset to observe in next steps.

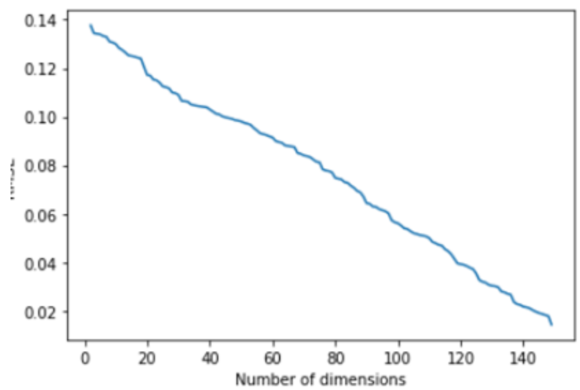
Using the best parameters (ccp_alpha=0.003 and criterion=`gini`)

4. Clustering on Dimensionality Reduction Algorithms

I have done all 16 clustering algorithms on Dimensionality Reduction algorithms. At the same time, I will write down the interesting results instead of analyzing one by one because there are 16 of them and they can be found in the code.



First of all, in general the clustering algorithms had

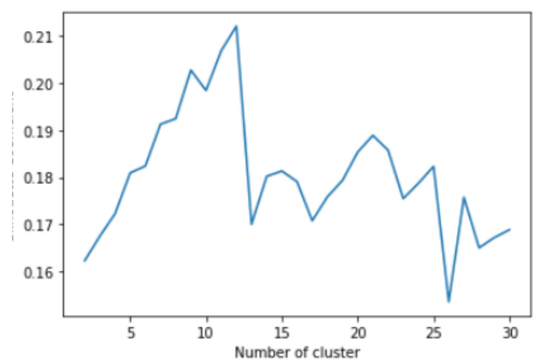


similar results on processed (dimensionality reduction) data, comparing to the original data.

When I ran K-Means on PCA algorithm, I have realized that, it got a better result on Silhouette score comparing to the original data at 16 clusters, which was unexpected.

I think this is because the data became easier to cluster after projected by PCA algorithm.

However, by using selectKBest algorithm, we can capture the largest Silhouette score.



We can see from the graph to find the highest score, resulting in over 0.21 Silhouette score. I think this is because the less important features are not interfering with the clustering algorithms so that it makes a better job. So that we can say that feature selection can increase the performance of Clustering if done correctly.

The resulting clusters differ a bit comparing to the original clusters and this is because some of the information is lost and the data is simply projected to some other dimension. But the difference is not so much because most of the information is kept in the dataset.

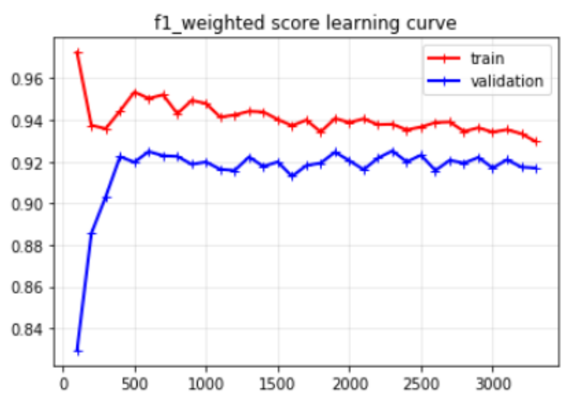
5. Neural Networks on Wine Dataset after Dimensionality Reduction

I have chosen the wine dataset because it was very promising with clustering vs label and it has over 0.9 accuracy with original dataset Neural Network setup. So, it will be better to make comparisons.

5.1. PCA Algorithm

As stated in 3rd section I have used 8 dimensions in the PCA algorithm, which covered 93% of explained variance ratio.

Firstly, I have used the sklearn's built-in Neural Network setup and got a 0.92 f1-weighted score.



After plotting the learning curve, it can be seen that the bias and variance are relatively balanced. I have used 2 hidden layers with 60 nodes each in Assignment 1. So, I wanted to explore the number of nodes using complexity analysis.

As it can be seen in the graph, fewer nodes (10 in this case) worked better for the validation set. This is probably because less features caused a simpler state space that can be captured with less nodes.

After this step, I have tested on the test set to see the results.

	precision	recall	f1-score	support
1	0.93359	0.91747	0.92546	521
2	0.92125	0.93373	0.92745	664
3	0.92043	0.92043	0.92043	465
accuracy			0.92485	1650
macro avg	0.92509	0.92388	0.92445	1650
weighted avg	0.92492	0.92485	0.92484	1650

8 features created by PCA algorithm could get 0.9248 weighted f1 score, comparing to original dataset's 0.9439 weighted f1-score.

Considering using almost half as many features, this seems to be a good result but it is still a significant lose for a supervised learning algorithm.

5.2. ICA Algorithm

Comparing to PCA's 8 features, I have used only 3 features in ICA Algorithm. The neural network (with 60,60 hidden-layers), which performed best in the Supervised Learning assignment, resulted in a high bias - low variance model on the 3 features of ICA.

So, I have made a complexity analysis with more nodes in hidden layers (up to 100) but it didn't help in a great deal. Using only 3 features from ICA caused a model that under-fits the data.

In the end, when tested against the test data:

	precision	recall	f1-score	support
1	0.92816	0.91747	0.92278	521
2	0.92694	0.91717	0.92203	664
3	0.90377	0.92903	0.91622	465
accuracy			0.92061	1650
macro avg	0.91962	0.92122	0.92034	1650
weighted avg	0.92079	0.92061	0.92063	1650

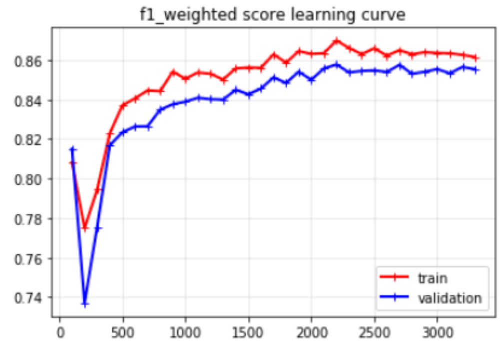
3 features of ICA gave a 0.92063 f1-weighted score, which is pretty close to PCA's 8 features. This means ICA has done an exceptional job at capturing the information out of original data only using 3 features.

5.3. Randomized Projections

I have used Gaussian Random Projection because the data I have used is a dense dataset, where there's no null or empty fields.

Since the performance of Randomized Projection was lower comparing to other algorithms, I have used 9 dimensions on this algorithm.

The built-in neural network's cross-validation only gave 0.86 f1-weighted score, which is super low.



After exploring different hidden layers, the Randomized Projection didn't give a very good result and couldn't go over 0.86 on cross-validation data.

When used against testing dataset:

	precision	recall	f1-score	support
1	0.86882	0.87716	0.87297	521
2	0.82840	0.84337	0.83582	664
3	0.86607	0.83441	0.84995	465
accuracy			0.85152	1650
macro avg	0.85443	0.85165	0.85291	1650
weighted avg	0.85178	0.85152	0.85153	1650

Random Projection could only get 0.85 f1-weighted score. This explained that random projection couldn't capture enough information out of the original dataset even though I have used 9 dimensions.

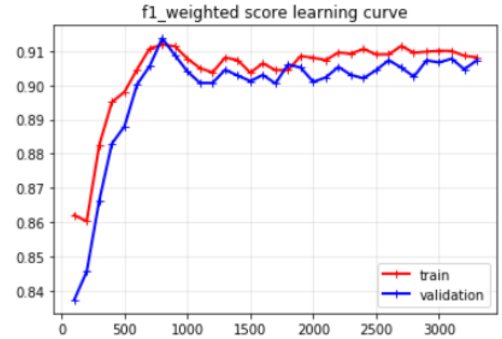
5.4. Select K Best Using Lasso

For the last one, I wanted to use Feature Selection (which is allowed by Prof.Isbell on Piazza). I have used only 5 features out of 13 in the wine dataset.

Using Lasso classifier, I have selected the most important features out of the dataset and used Neural Networks on them.

I have used the best neural network from Assignment 1 and the learning curve of this network was:

It can be seen that this algorithm suffers from high-bias, as well. So, I have made a complexity analysis on number of nodes in hidden layers and even 100 nodes didn't make a significant increase in cross-validation score.



I have used this neural network against the testing dataset and got 0.9 f1-weighted score

	precision	recall	f1-score	support
1	0.90286	0.90979	0.90631	521
2	0.89833	0.89157	0.89494	664
3	0.89914	0.90108	0.90011	465
accuracy			0.90000	1650
macro avg	0.90011	0.90081	0.90045	1650
weighted avg	0.89999	0.90000	0.89998	1650

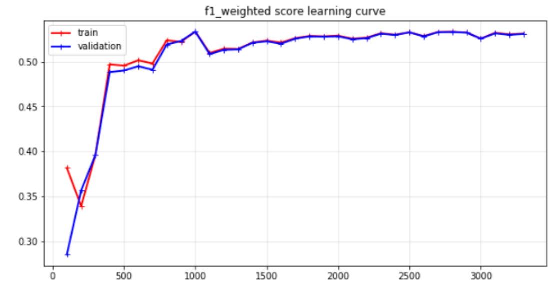
Even though the result is not so bad, it can be seen that the model under-fits the dataset. This means some valuable information is left out.

6. Neural Networks on Wine Dataset after Clustering

In this chapter, due to the complexity of Neural Networks, I set the clusters with 1 feature to decide on the labels. (not adding the clusters to the original dataset)

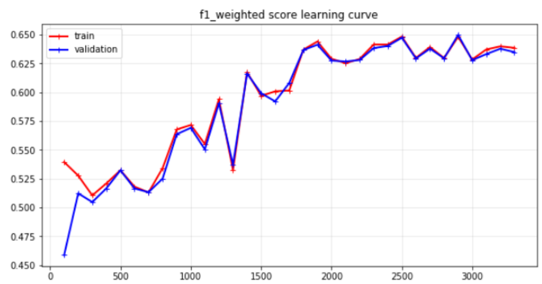
6.1. K-Means Clustering

I have used 15 clusters out of K-Means (since it gave a better Homogeneity score than optimum k=3 and it will make more sense to put more clusters in to the neural dataset) and put this inputs in the neural network. The built-in configuration of Neural Network gave 0.63 f1-weighted score on cross-validation.

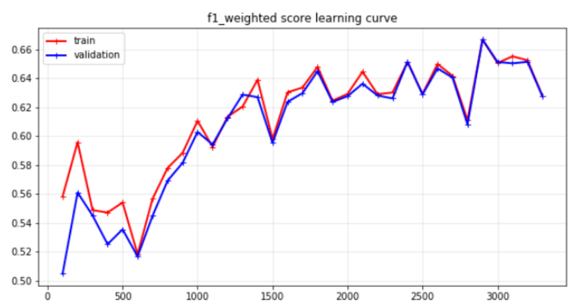


When draw the learning curve, it can be seen that the model suffers from high-bias and this is because there is not enough data to make the discovery.

To reduce bias, I have used two hidden layers with 60 nodes each and as a result the bias is reduced.



To furthermore reduce the bias, I have used two 100 nodes hidden layer and the bias didn't reduce more.



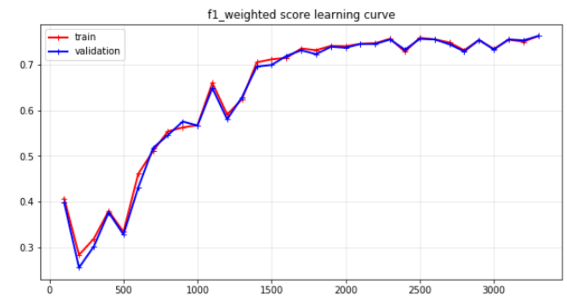
So, I have used this setup on the testing set and

	precision	recall	f1-score	support
1	0.79779	0.41651	0.54729	521
2	0.58302	0.94127	0.72005	664
3	0.93464	0.61505	0.74189	465
accuracy			0.68364	1650
macro avg	0.77182	0.65761	0.66974	1650
weighted avg	0.74993	0.68364	0.67165	1650

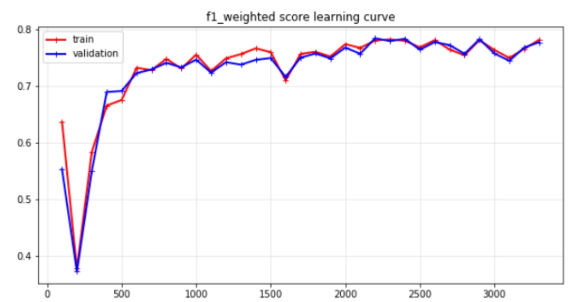
The model performed 0,67 f1-weighted score on the testing set. Only using clusters, the neural network guessed 2/3 (accuracy 0.68) of the testing set. Even though this seems good, the clustering (k=3 K Means) could get 0.9 accuracy score. So, we can say using multiple clusters in a neural network itself doesn't work well comparing to using only clusters and using the dataset in the neural network.

6.2. Expectation Maximization

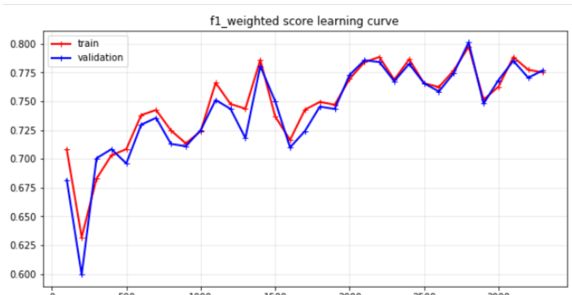
For Expectation Maximization algorithm, I have used 15 clusters, as well to make healthy comparisons between K-Means. The built-in configuration of Neural Network gave 0.75 f1-weighted score on cross-validation. Even this one is higher than K-Means's best result itself.



This model has high-bias as well. So, I use two hidden layers with 60 nodes each and draw another learning curve.



It can be seen that the model reduced its bias in a great deal by adding more complexity to the neural network. Now, I have increased the number of nodes from 60 to 100 in each hidden layer to increase the complexity more.



In the graph above, the bias didn't reduce as expected. The model underfits and still can't explain the dataset only using its clusters.

With the testing dataset below:

	precision	recall	f1-score	support
1	0.72495	0.76392	0.74393	521
2	0.78924	0.79518	0.79220	664
3	0.93519	0.86882	0.90078	465
accuracy			0.80606	1650
macro avg	0.81646	0.80930	0.81230	1650
weighted avg	0.81007	0.80606	0.80756	1650

The weighted-f1 score is 0.80. Comparing to K-Means' neural network, this is a much better result. As I have mentioned in the Clustering section, this is because EM acts more flexible and captures clusters. The Neural Network are able to make better prediction by using these clusters.

7. Conclusion

I had two datasets; UFC Dataset was very complex with a over 150 features. Dimension Reduction algorithms bring out more performance on this dataset, even using less than half PCA explained 0.98 of the variance, which is great.

On the other hand, less complex Wine Dataset was clustered in correspondence to the labels such that K-Means got 0.9 accuracy score with the true labels.

Neural Network on dimension reduced datasets worked really good except for the RP algorithm and the selectKBest. I think ICA worked really well on this section such that it bring over 0.9 f1-weighted score using only 3 features.

8. Clock Time

Neural network algorithms were much faster when run on clusters or even reduced dimensions. This is because they run on lower space state.

8. References

[1] "Ultimate Fighting Championship," Wikipedia, 09-Feb-2020.[Online].Available:https://en.wikipedia.org/wiki/Ultimate_Fighting_Championship. [Accessed: 10-Feb-2020].

[2] K. Aggarwal, "UFC Fight Data," Kaggle, 11-Dec-2018. [Online].Available:https://www.kaggle.com/calmdownkarm/ufcdataset. [Accessed: 10-Feb-2020].

[3] J. Vanschoren, "wine," OpenML. [Online]. Available: https://www.openml.org/d/187. [Accessed: 10-Feb-2020].