Unsupervised Learning - Project 3

A. (Re)Introduction to the Data

In *Project 1* I set out to put wine sommeliers out of business for good, while also convincing every last cretin of fine wine's merit. Having succeeded, I now return for a victory lap in Project 3. You will recall that I accomplished these amazing feats by analyzing two datasets of wine-related data: the first was a very large (60,000 instances after much pruning, and over 100 features after pre-processing) dataset of wine reviews with textual description of the wines. I classified these wines based on the variety of grape used to produce them. This is the "Big Dataset". The second dataset is much smaller (1600 instances with 11 features) and focused only on objective measurement of the wines, with features like "residual sugar" or "pH". This "Small Dataset" has labeled every wine with a quality ranging from one to ten. These datasets interest me even beyond their technical differences: the Big Dataset has passed the Small Dataset through a human filter - taste. This might add a significant amount of noise, but it might also add information through that human's long experience of tasting wine. As a quick note, the clustering graphs I will provide throughout this assignment are laid out so that the y-axis is the target variable (variety or quality) and the x-axis and color both correspond to the cluster of those instances.

Table 0 - Dataset 1 Features

description	country of origin	price	review score	grape varietal
Pineapple rind, lemon pith and orange blossom start off the aromas. The palate is a bit more opulent, with notes of honey-drizzled guava and mango giving way to a slightly astringent, semidry finish.	*USA*	*13*	*87*	*Riesling*

Table 1 - Dataset 2 Features

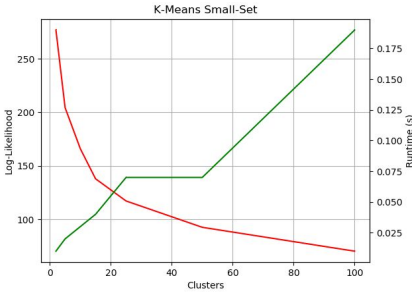
	fixed acidity	volatile acid	citric acid	Residual sugar	chlorides	Free sulfur	Total sulfur	density	pH	sulphates	alcohol	quality
Examples	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5

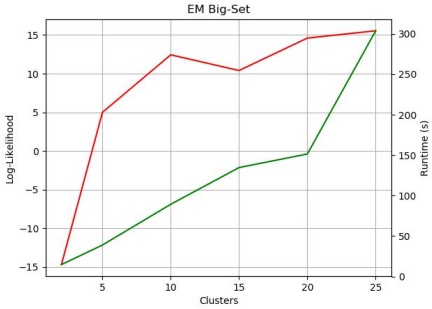
B. Simple Clustering

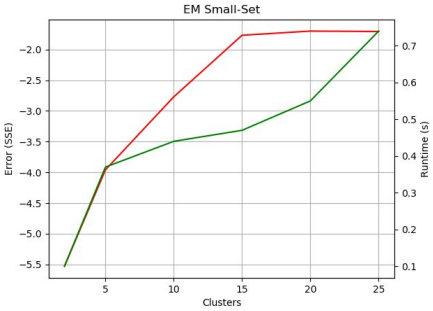
Before I get to the fun parts of the analysis, I will briefly explain how I arrived on the parameters for the datasets on each clustering algorithm. The green lines are runtime, the red lines are error or its equivalent. For all, I chose to measure distance in the euclidean fashion. It performed better than any other measure, while also having a

generally above-average runtime. Many of the metrics (e.g. Minkowski) didn't make much sense for the Big Dataset, since it is relatively sparse.

Determining "k" (number of clusters) required balancing a few factors. Of course as k increases, runtime will also increase (remembering that the possible total number of configurations is k^n). For k-means, this was noticeable but not a huge hassle - at 100 clusters it only took three minutes. But for Expectation Maximization this quickly got out of hand: at 25 clusters it took over five minutes to run. Also important is performance: the more clusters the better performance (you could have one cluster for every instance and get zero error). This held fairly true, except for a strange dip for EM on the Big Dataset. Finally, you don't want to "overfit" your data. If we make too many clusters, than each cluster isn't really saying anything useful about the data. We also tend to see a diminishing returns effect as we increase k. With all this in mind, I decided on the k values in the table below.





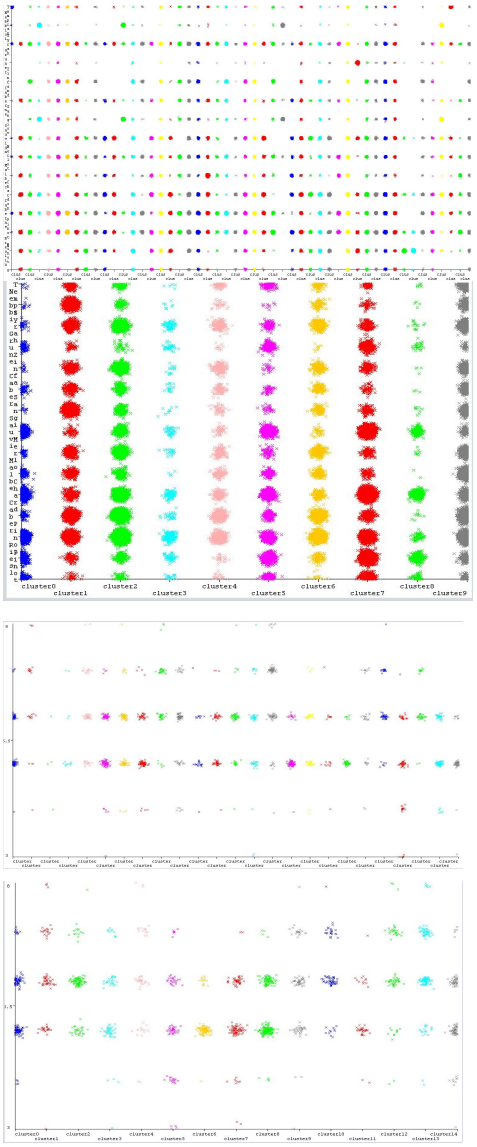


K-Values Table		
k-values	Big Dataset	Small Dataset
K-Means	50	25
EM	10	15

I was fairly surprised by clustering results, since we seem to have received more coherent clustering with the Big Dataset than the Small Dataset. Given the sparse and mostly nominal (instead of numerical) aspect of the Big Dataset, I did not expect clustering algorithms to be effective on it. Conversely, I expected the Small Dataset, whose attributes are all (seemingly) normally distributed**,** to lend itself quite well to clustering. But, as you can see (the first two clustering graphs are for K-Means and EM for the Big Dataset), even though we excluded the varietal attribute from clustering, we can see pretty clearly that the clusters correlate with varietals for the Big Dataset. The graph isn't included, but there was also a pretty nice lining up between Country of Origin and Clusters. This tells me that, even though only 3-4 features of the 100 were "active" or hot for any given instance, each feature was powerful enough to find themselves into a group. We also see that simple k-means actually seems to have performed slightly better than EM, and it ran faster. I can't think of a great

reason why k-means hard cluster assignment would perform better than EM's probability based assignment, but with further thought there might be something there.

The Small Dataset was probably hampered by two main issues. First, a theoretical 5.4 quality wine is assigned to the 5 bucket, where a theoretical 5.5 is assigned to the 6 bucket. So with theoretically perfect clustering, a cluster that straddles the line between to (arbitrary) quality buckets would have relatively poor likelihood or error. With this in mind, (it is a little hard to see), you can see some clusters that cross over the quality lines (i.e. between 4 and 5, 5 and 6, etc.). You would expect EM to do better here (because of the normally distributed variables, and the fact that a wine with a score of 5.5 could theoretically belong to either 5 or 6, and EM doesn't force you to make a choice). In fact it does, but we did use more clusters for it, and it did take longer to run.



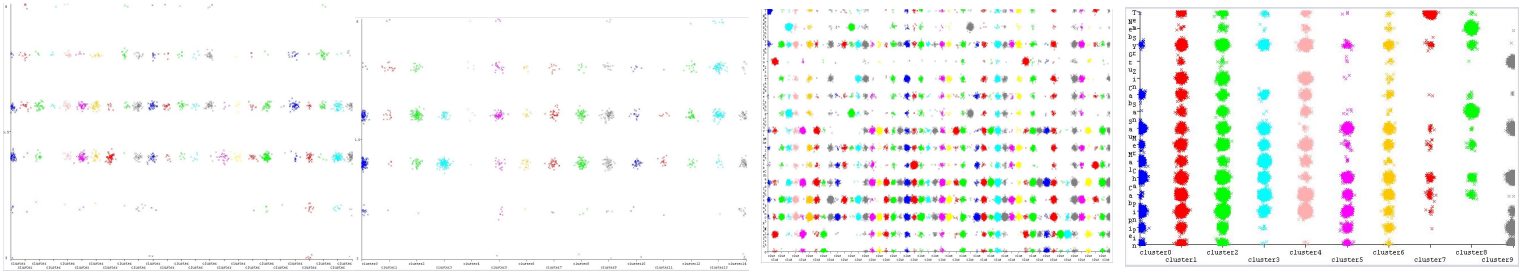
For both datasets, some more creative pre-processing could probably improve our performance here. In particular I think incorporating word2vec into my text processing for the Big Dataset could really improve performance, since I think it is a more effective numerical representation of text than a simple bag of words. In general though the relatively messy clusters is probably due to the highly dependent nature of the attributes of each dataset. That is, an instance with the word "crisp" is also quite likely to have the word "tart", or a wine with low pH likely also has high citric acidity. Thus it is hard to find a neat cluster around a feature.

C. Dimensionality Reduction

We start with PCA. For the small dataset, I somewhat arbitrarily decided that since we have four features dealing with acidity, and three features dealing with sulfur, we could reasonably turn those seven features into two, so I adjusted our target variance until it hit that number (in the end we had six total features). For the Big Dataset I didn't really have a heuristic and just played around until I felt we reached a more manageable number of features (we had 30 in the end, down from over 100).

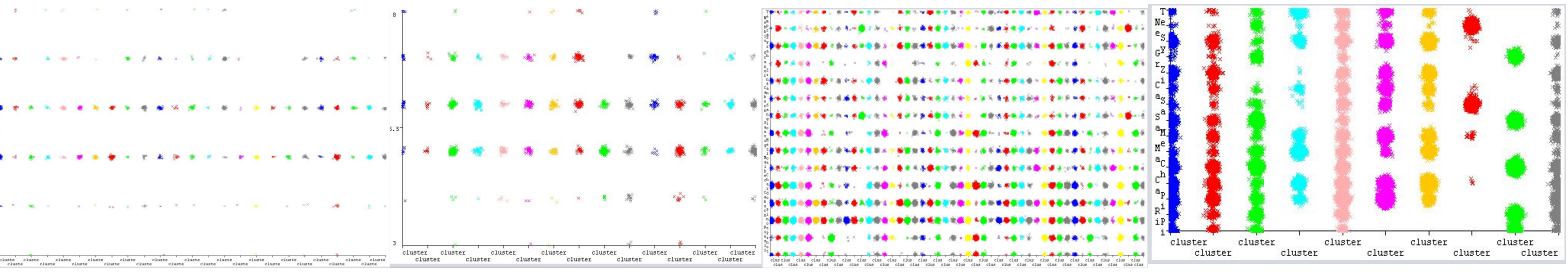
Surprisingly enough my intuition for the small dataset was correct and our “best” two new features were primarily a combination of the acidity features and sulfur features respectively. You can see the eigenvalues below, we have one very valuable feature at 3.09 (comprised largely of acidity features), some middle values from just below two (the sulfur features) to just below one, and one slightly smaller value at .66. This is about what you can expect, and in fact we could probably trim that last feature without seeing too much of a performance decrease. Clustering for the small set looks pretty similar but has better error/likelihood. The larger set did not fare as well: although it also had smaller error the clustering seems to have lost some of its relation to our desired output variable (variety or quality). We should expect error for PCA (and every other algorithm for that matter) to decrease, since we are removing and combining variables. We have fewer variables to contribute to sum error and the variables are, despite our best wishes, going to carry at least *a bit* less information (since there is no free lunch for dimensionality reduction). Unless the error *doesn't* go down with the other algorithms, I'll leave the error discussion out to avoid redundancy. The graphs, and all others for this section, are in the order: Small Dataset K-Means, Small Dataset EM, Big Dataset K-Means, Big Dataset EM.

Small Dataset PCA Eigenvalues					
3.09	1.93	1.55	1.21	0.96	0.66

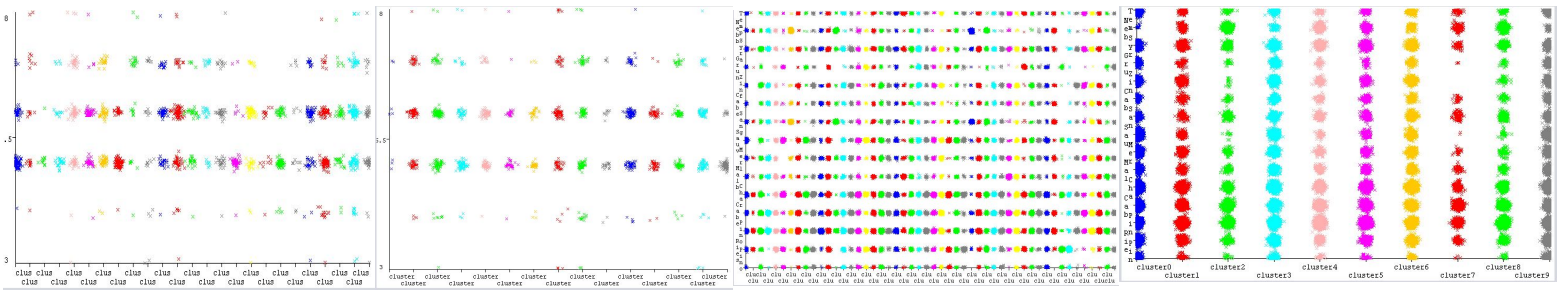


The Big Dataset, probably unsurprisingly, did not take well to feature selection. Although the words “crisp” and “tart” may be related and jointly applicable to a wine, the frequency with which they are both used (giving us good basis to combine them via PCA) is so low (because of the great number of synonyms in English) that PCA just can't get a good handle the dataset. This is a particular example of where I think word2vec could help, since two words of similar meanings have similar vectors, which would provide an avenue for PCA, and other algorithms, to assess the similarity of a feature representing a word. I won't list every eigenvalue, but most values fell between one and two, with only one value above three. Most of the eigenvectors are dominated by two or three base features.

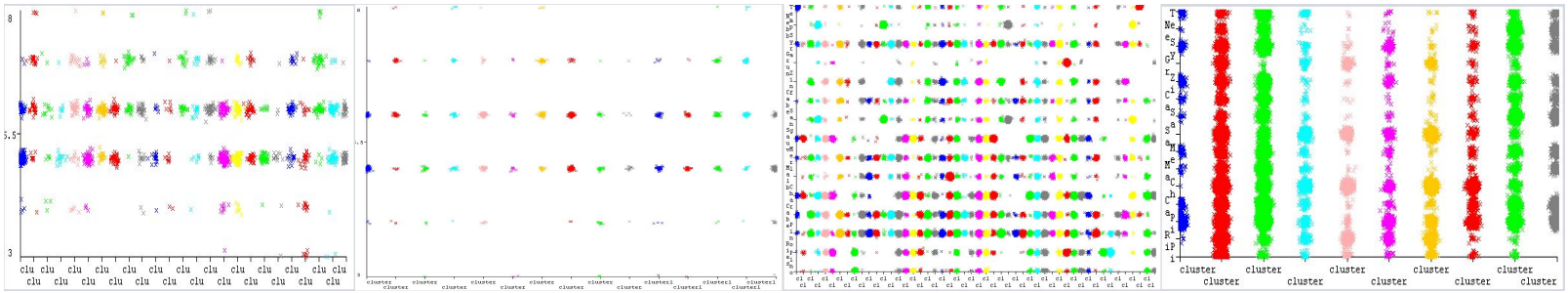
I ran ICA four times, twice times for each dataset: once without reducing the number of features, and once for the number of features identified in PCA. I don't have a good way of displaying kurtosis, and I assume you don't want to scroll through a table of length 30 or 100, so I'll just describe the general trends. For the Big Dataset, we see a very large increase in kurtosis. Since that data is mostly from a bag of words, the vast majority of values are 0 or 1, so we really don't need to do that well to show kurtosis improvement. A couple features had very high kurtosis, hitting 36 and 44. Most hover in the zero to five range. The ICA run with less features for the Big Dataset had nearly identical performance, which given the paucity of information from each individual feature, cutting it down to a third really shouldn't give us that much more. The Small Dataset has much more interesting performance with ICA. Overall kurtosis increased, with two features breaking 50. We found similar relationships as in PCA, with ICA projection axes showing some relationship between the acidity features and sulfate features. Clustering seems to have gotten slightly worse for the small dataset but a few of the clusters on the Big Dataset are interesting. In the EM clustering graph, the cyan, second red, and second green clusters show particular sensitivity to varietal (again varietal *was not* included in clustering) which is exactly what we're looking for.



Randomized Projections seems to have worked just as well as ICA and PCA. As I ran it multiple times, the variance of each individual variable got bigger and bigger (I would guess exponentially). This was for both datasets. The Big Dataset RP attributes were either discrete seeming approximations of a normal function or resembling the curve of the original price feature (which makes sense whenever the price feature was used in projection). The Small Dataset new variables were all normal-looking curves. Performance wise I didn't see much difference after multiple runs (3 for the clustering below). As you can see in our clustering results below, the Small Dataset clusters showed pretty good sensitivity to quality. The Big Dataset might look a little worse than its ICA and PCA performance but given the naivety of the approach to dimensionality reduction I am pretty impressed.



My final dimensionality reduction I chose separate algorithms. For the Big Dataset I ran an Information Gain based algorithm. I like info gain because it is the most simple and intuitive algorithm for me: just pick the features that help the most. It should be noted that it is a feature selection algorithm rather than a feature combiner like the previous algorithms, so intuitively we are losing more information. For the Big Dataset it picked out (keeping with 30 attributes) the country and price as the two most useful features. This tracks, since *every* instance has a price and country, whereas most attributes go unused for most instances. Info Gain had pretty good results for clustering - the pink and yellow clusters for EM show pretty good variability for Varietal. I tried to pick a similar algorithm for the Small Dataset (infogain wouldn't work for it), and I ended up with Random Subset, which is also a feature selection algorithm. Random Subset randomly picks a number of features. This really doesn't have any advantages, other than a quick runtime, but it should be a good baseline for other algorithms. Theoretically this could give us identical performance to Information Gain, if we randomly select the same features info gain would select. In practice though this is going to depend entirely on which features I roll. Given we just threw away a random half of our information, it didn't do too bad on clustering.



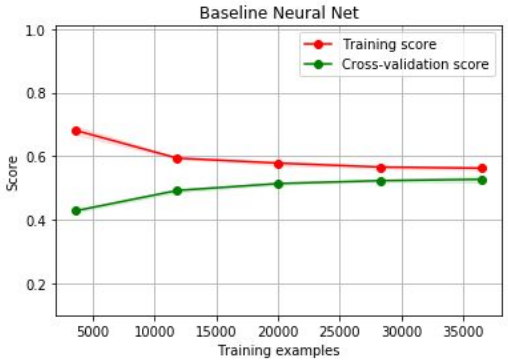
Overall, we got seemingly pretty different clusters after applying dimensionality reduction. We were able to see clustering around varietals a little bit more than before, which is pointing in the right direction I think the main reason this is the case is because of the complexity of the data. If this were a simple binary classification problem, we might have been able to see the trends (or lack of trends) more clearly. But since we have so many features and so many labels, the trends are obscured, and small changes can cause a large change in the behavior of clustering.

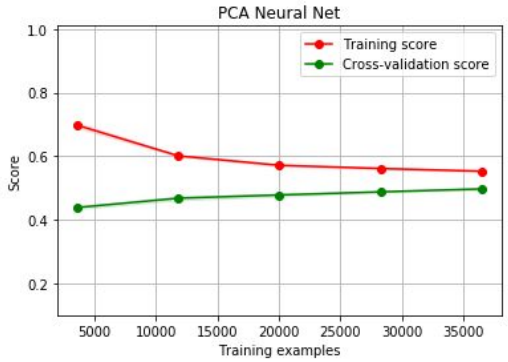
C. Fun With Neural Networks

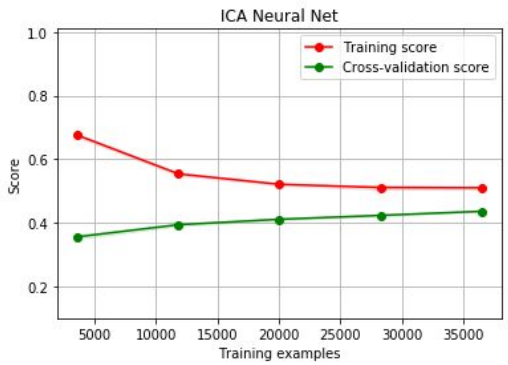
Everything for Neural Networks will be ran with the Big Dataset for a few reasons. First, I think it is the more interesting dataset with juicier applications. Second there are a number of issues it faces with Neural Networks that dimensionality reduction and clustering could help solve. The runtime here is a big factor, it would be hard to

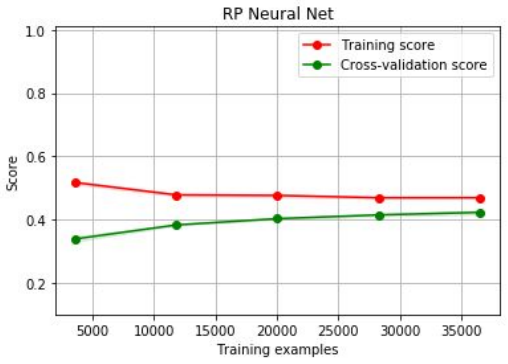
differentiate between the .002 and.001 seconds that the Small Dataset takes to train. The Big Dataset takes 46 seconds to train without reduction, so we have a lot of room for improvement. Also of interest is the sparsity of the instances of the Big Dataset - Neural Networks traditionally have issues on sparse datasets. Hopefully our dimensionality reduction will solve this problem (although I will note, since we have so much data (45,000 instances in the training set) this effect was reduced.

As you can see from the graphs and results table (below), our dimensionally reduced data performed very nearly as well as un-reduced. PCA performed the best of our reduction algorithms, and probably could have beaten the baseline if it had a good run and the baseline had a bad run. Since we reduced our number of features from over 100 down to just 30, you could probably expect better runtimes, and faster convergence. In fact we do see better run times when we have less features. We see that RP and InfoGain have the best runtimes, but I don't think there is a mathematical reason why, rather I think I stepped away from my computer, and the reduced usage (YouTube, video games, etc.) may have given the training more resources than before. Most surprising is InfoGain's performance, coming pretty close to PCA. This tells me we probably have a few very powerful features and many not so helpful features. None of these seemed to converge any faster. We might have gotten faster results with different hyperparameters, but my grid search for hyperparameters looked to maximize F1 Score and not convergence speed. Also of Interest: I ran PCA again and cut it down to 10 features rather than 30. As you can see below, it (predictably) didn't do too well as we cut out too much information.

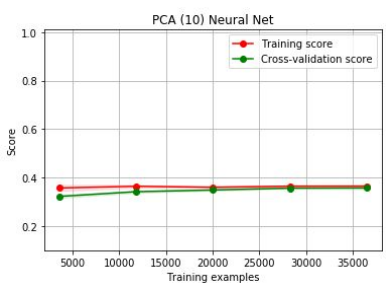


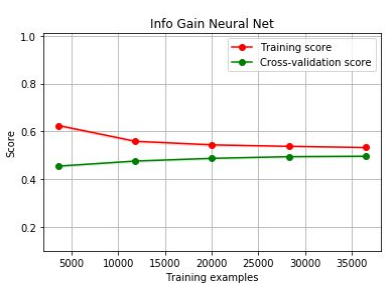




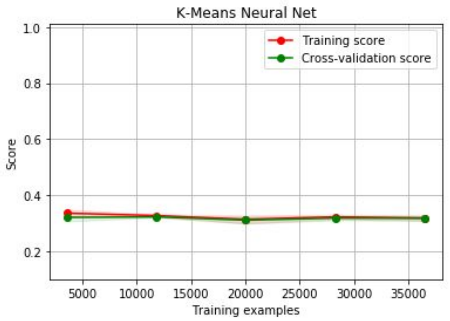


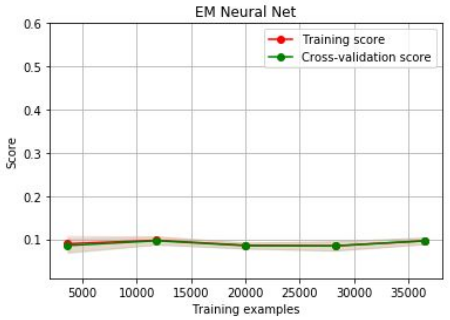
Neural Net Performance - Dimensionality Reduction					
	Baseline	PCA	ICA	RP	Info Gain
Accuracy (%)	60.1	56.7	54.4	50.9	56.3
F1 Score	.527	.493	.447	.427	.501
Runtime (s)	84.9	65.4	67.3	47.4	49.2





Clustering Neural Nets		
	K-Means	EM
Accuracy (%)	*38.5*	*28.1*
F1 Score	*.310*	*.082*
Runtime (s)	*7.65*	*8.63*





Our results using the clusters as features was significantly less impressive. To some extent, I understand this from the EM results because we took it down to so few clusters (10) - it would be impossible to get a perfect score here because we simply don't have the information to guess (with only 10 different possibilities of input and 15 possibilities of output). If we could somehow encapsulate the probabilistic aspects of EM (i.e. second or third choices of a cluster), then we might get further in that direction. Similarly, although we have 50 clusters with out k-means attempt, we don't have a lot of flexibility when it comes to prediction. We are essentially forced to assign a cluster to a wine varietal, and allow for no other input. They both ran quite quickly, which makes sense because the feature set is so small, but this really doesn't help since the performance is poor.

D. Final Thoughts

Having run all the experiments, I think it's clear that my datasets are not a picture perfect example of unsupervised learning. I've talked about this in prior assignments so I won't go too in depth, but the subjective nature datasets makes it difficult to get really clean results. The inputs of my Big Dataset (just descriptions of taste) and the subjective nature of outputs for the small dataset (a quality score) really muddy the waters. I could have spent more time tuning the number of clusters or some other parameters but I really wasn't seeing too much real variation on first attempts. I considered playing with the output of the Big Dataset to be a simpler binary classifier (Red Wine or White Wine). But decided that, although this would probably clean things up quite a bit, it really wasn't an interesting problem to solve after all. As for the Small Dataset, it is again handicapped by its lack of examples for qualities outside the range 4-7. I think the extreme examples would tell us more than the middling examples, and give us some clearer grounds clustering.