Unsupervised Learning

Eric Gu
November 20, 2018

For full code and notes, see "(CS 4641) Assignment 3 – Unsupervised Learning.ipynb".

Written in Jupyter Notebook using Python's scikit-learn library and documentation, and with additional reference to Python Data Science Handbook. Data sourced from Kaggle.com.

Dataset Recap

We will skip the rudimentary data exploration and preprocessing that has been explained in past assignments.

Avocado Prices

Our first dataset is on Hass avocado prices from 2015–2018, downloaded from the Hass Avocado Board website in May 2018 and compiled on Kaggle.com. We'll try to predict the average price of avocados by clustering on features like sales numbers by PLU, date & time, and location.

	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	month	day	year	region
0	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	12	27	2015	Albany
1	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	12	20	2015	Albany
2	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	12	13	2015	Albany
3	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	12	6	2015	Albany
4	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	11	29	2015	Albany

The dataset contains 18249 rows of weekly retail scan data for national retail volume (units) and price. It is missing no values, and is well-balanced across type, region, and datetime.

This dataset is relatively simple: it has few real features, some of which are likely already correlated. Furthermore, we will encode the categorical variables and discretize the average prices to turn it into a multiclass classification problem, increasing the dimensionality of the problem and making the number of correlated columns even higher. Because it comes from real-world sales, the dataset is likely messy and will provide an interesting comparison between the two clustering algorithms' sensitivity to noise (before and after dimensionality reduction). Perhaps most importantly, this dataset contains only a posteriori information on average prices of avocado sales in relation to the times and locations they were sold at. Without feature engineering additional data like weather, transportation costs (e.g. oil prices), and cost of living, we can only rely on PCA to infer the most important axes of variance that may reflect those underlying causal relationships.

Breast Cancer in Wisconsin

For our second classification problem, we'll explore a smaller, less balanced dataset on breast cancer diagnostics published in a paper from the University of Wisconsin. It was donated to the UCI Machine Learning repository in 1995 and is available on Kaggle.com.

id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry
842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. Ten real-valued features are computed for each cell nucleus. The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features.

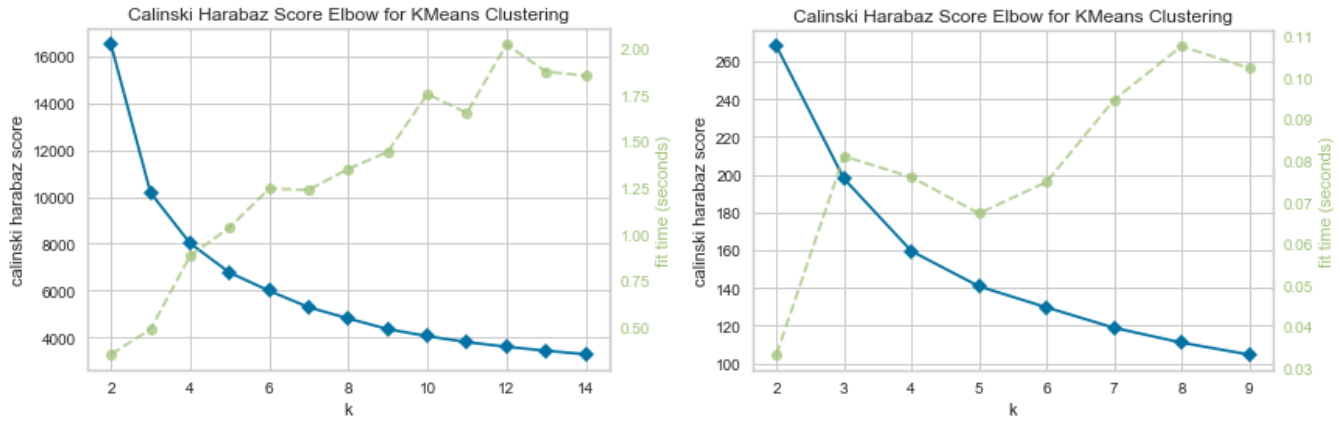
We expect that many of the real and engineered features will be correlated, and not all need to be included as inputs into the clustering algorithms. Dimensionality reduction will certainly help reduce a lot of noise and improve the test performance of those algorithms.

Clustering Without PCA

K-Means Clustering

We will be using the KMeans and PCA algorithms from scikit-learn library and comparing the results from running them on each dataset using several clustering performance evaluators. This implementation uses flat geometry (distances between points) for its distance metric. An important observation for *K*-means is that the clusters must be circular and are nonprobabilistic, which means that it cannot account for oblong or overlapping clusters very well. Both our datasets are noisy (the avocado dataset more so than the cancer dataset), and they likely overlap and do not fit into neat circular clusters in hyperspace. Some of these issues can be alleviated with PCA, but I predict we may have more success with GMMs.

First, we need to find an appropriate *K* (the number of clusters) to run the algorithm on. We'll graph the algorithm's average performance for k = (2,15) for Avocado and k = (2,10) for Cancer—our performance measure being the Calinski Harabaz score, defined as the ratio between within-cluster dispersion and between-cluster dispersion.



Seeing as there is no clear point of inflection in either of the line charts (they start at their peaks, k=2), we will rely on our domain knowledge of the true number of classes: the avocado dataset has 5 clusters by construction (corresponding to 5 price ranges), and the cancer dataset has 2 clusters (corresponding to Benign or Malignant).

Now we can run *K*-means. We want to focus more on adjusted Rand Index (ARI) for the Avocado model because its ground truth clustering has large, balanced classes; we'll focus on Adjusted Mutual Information (AMI) for the Cancer model because its ground truth clustering is unbalanced (fewer Malignant than Benign cases).

```
In [15]: k_means(avocado_X, avocado_Y.values.ravel(), n_init=50)
```

```
n_digits: 5,     n_samples 18249,        n_features 72

init            time    inertia homo    compl   v-meas  ARI     AMI     silhouette
k-means++       4.90s   87749   0.149   0.255   0.188   0.150   0.148   0.119
random          5.55s   90410   0.157   0.198   0.175   0.134   0.157   0.092
```

```
In [16]: k_means(cancer_X, cancer_Y.values.ravel(), k=2)
```

```
n_digits: 2,     n_samples 569,          n_features 30

init            time    inertia homo    compl   v-meas  ARI     AMI     silhouette
k-means++       0.05s   11590   0.544   0.563   0.553   0.671   0.543   0.353
random          0.04s   11590   0.537   0.555   0.546   0.665   0.537   0.353
```

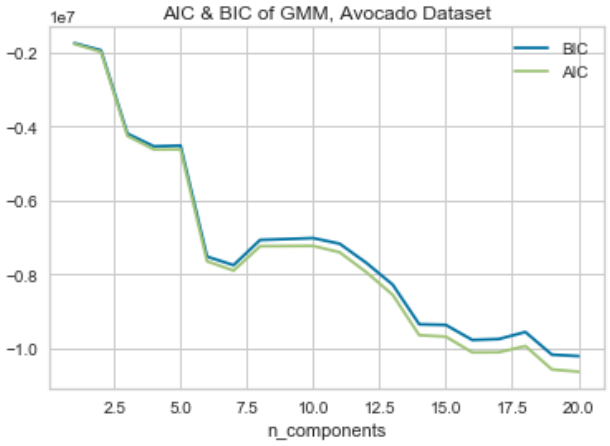
Results

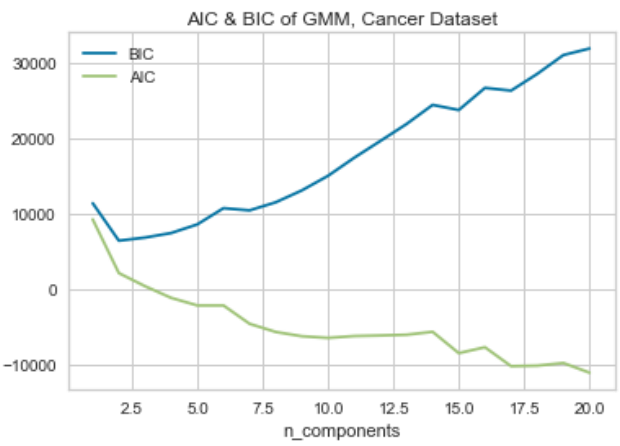
The results are lackluster for the avocado data. While it's hard to visualize what these clusters might look like in hyperspace, the silhouette score is close to 0, indicating that many points are very close to the decision boundaries and the clusters heavily overlap. The ARI is 0.150, which indicates very low agreement, resembling something closer to random label assignments. This is not surprising, seeing as the dataset has had no dimensionality reduction and *K*-means is sensitive to outliers and noise. Additionally, we ran the algorithm for 50 initializations using k-means++ seeding (cluster centers far away from each other) to ameliorate the algorithm's proneness to local minima, but we saw negligible gains on our evaluation measures over random seeding. The state space is still very large and may hide more optimal clusterings, but we will continue to run *K*-means for 50 initializations for the remainder of the report.

K-means worked better for the cancer data, which was much smaller and had fewer features. The silhouette score of 0.353 suggests that the clusters have less overlap. The AMI was 0.543, which indicates clusterings quite similar to the ground truth, verging on significant agreement.

GMMs via Expectation Maximization

As mentioned earlier, GMMs are more flexible because they allow for probabilistic cluster assignments. We'll be using GaussianMixture from scikit-learn, which uses Mahalanobis distance.





From the graphs of the Akaike information criterion (AIC) and Bayesian information criterion (BIC), the Avocado dataset could benefit from more than 5 components, while the Cancer dataset performs best with 2 components. For the sake of comparison, we will choose to use the same number of components for our GMMs as the number of clusters we used for *K*-means, but we will allow for the model to use full covariance (which corresponds to the true generative model) and seed instances using both *K*-means and random assignment.

```
GMMs(avocado_X, avocado_Y.values.ravel(), n_init=50)
```

```
n_digits: 5,     n_samples 18249,        n_features 72
```

init	time	homo	compl	v-meas	ARI	AMI	silhouette
kmeans	147s	0.029	0.035	0.032	0.013	0.029	0.009
random	121s	0.007	0.007	0.007	0.005	0.007	-0.084

```
GMMs(cancer_X, cancer_Y.values.ravel(), k=2)
```

```
n_digits: 2,     n_samples 569,          n_features 30
```

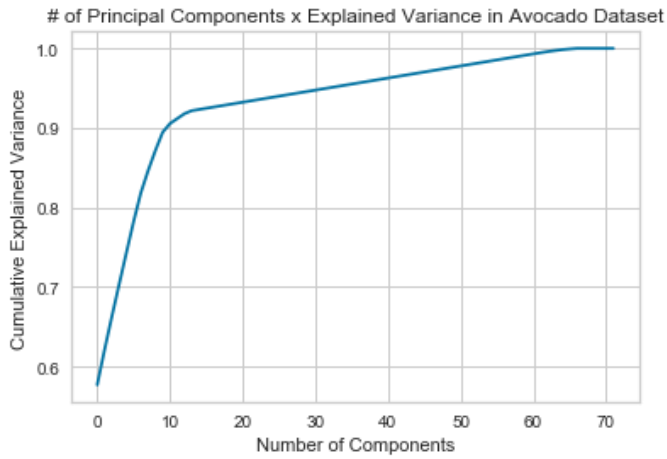
init	time	homo	compl	v-meas	ARI	AMI	silhouette
kmeans	00s	0.612	0.606	0.609	0.725	0.605	0.313
random	00s	0.562	0.549	0.556	0.667	0.549	0.282

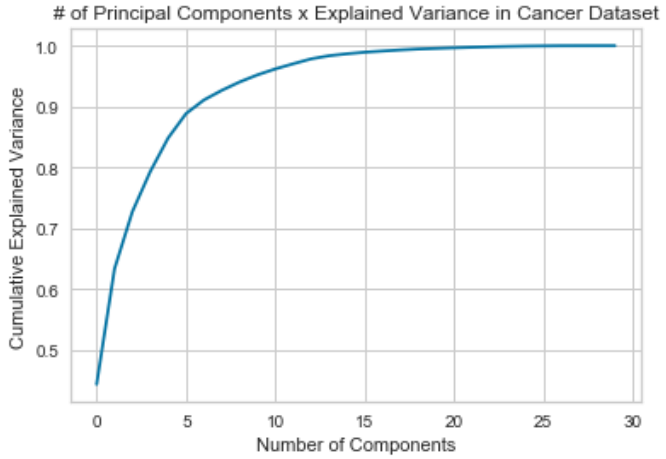
The GMMs performed with mixed results. Surprisingly, it took about 30 times longer to run than *K*-means in wall clock time and resulted in an adjusted Rand Index of 0.013 for the Avocado dataset, roughly equivalent to completely random (uniform) label assignments. The silhouette score is next to 0, meaning the model's clusters largely overlap. We can conclude that this model failed.

The Cancer dataset fared better on the other hand, with an AMI of 0.605. In hindsight, the long running time for the Avocado dataset was not unexpected, but the low performance reflects the difficulty in initializing Gaussian components with very high dimensional data and many parameters. The Cancer model is small, and must have benefited from the additional flexibility of non-spherical cluster geometry and "soft" assignment, since the silhouette suggests that the clusters have about the same amount of overlap as the *K*-means model.

PCA

Now, let's perform some dimensionality reduction on our datasets. Graphing cumulative explained variance against the first *n* components:





The graphs are quite striking. The Avocado dataset hits rapidly diminishing returns after around 10 or 11 components, while the Cancer dataset has most of its gains in the first 5 components.

After PCA, there are now 11 and 5 columns in the reduced Avocado and Cancer datasets, respectively. The transformed datasets are otherwise not very meaningful to human inspection and are difficult to graph in 2D.

Clustering with PCA

We now run all the previous code for clustering algorithms on the newly transformed columns for each of our problems. The elbow method and BIC graphs again do not give us a clear inflection point, so we'll rely on our domain knowledge and keep the number of clusters and components the same as before.

```
k_means(avocado_X_pca, avocado_Y.values.ravel(), n_init=50)
```

```
n_digits: 5,     n_samples 18249,        n_features 11

init            time    inertia homo    compl   v-meas  ARI     AMI     silhouette
k-means++       2.11s   66877   0.149   0.256   0.188   0.150   0.148   0.153
random          3.14s   69538   0.157   0.199   0.175   0.134   0.157   0.137
```

```
k_means(cancer_X_pca, cancer_Y.values.ravel(), k=2)
```

```
n_digits: 2,     n_samples 569,          n_features 5

init            time    inertia homo    compl   v-meas  ARI     AMI     silhouette
k-means++       0.05s   9003    0.537   0.555   0.546   0.665   0.537   0.362
random          0.03s   9003    0.537   0.555   0.546   0.665   0.537   0.401
```

```
GMMs(avocado_X_pca, avocado_Y.values.ravel(), n_init=50)
```

```
n_digits: 5,     n_samples 18249,        n_features 11

init            time    homo    compl   v-meas  ARI     AMI     silhouette
kmeans          31s     0.139   0.149   0.144   0.120   0.139   -0.012
random          47s     0.152   0.172   0.162   0.120   0.152   0.071
```

```
GMMs(cancer_X_pca, cancer_Y.values.ravel(), k=2)
```

```
n_digits: 2,     n_samples 569,          n_features 5

init            time    homo    compl   v-meas  ARI     AMI     silhouette
kmeans          00s     0.408   0.390   0.399   0.466   0.389   0.296
random          00s     0.402   0.383   0.392   0.457   0.383   0.284
```

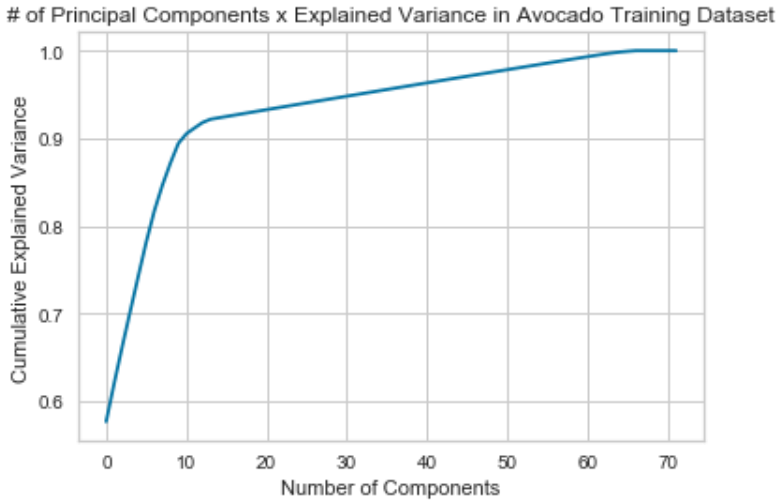
The results are not too exciting. Avocado *K*-means model has ARI of 0.150 (exact same as before) and the Cancer *K*-means model has AMI of 0.537 (slightly lower than before). Wall clock run times for the Avocado model halved and theoretically the clustering models for both should be more generalizable to new out-of-sample data, so it's impressive that in-sample performance stayed the same.

If anything, the performance for GMMs deteriorated significantly. The ARI and AMI scores of the models for both datasets are lower than the models fitted on the original, full datasets. This is because the extra information discarded during dimensionality reduction was likely incorporated into the Gaussian mixture models to tweak shape and density. In general, I would hypothesize that probabilistic models suffer more from dimensionality reduction than deterministic algorithms.

It appears that both of these problems are somewhat intractable for clustering algorithms that rely on ordinary diagonal distance measures in higher dimensional Euclidean space, even with dimensionality reduction.

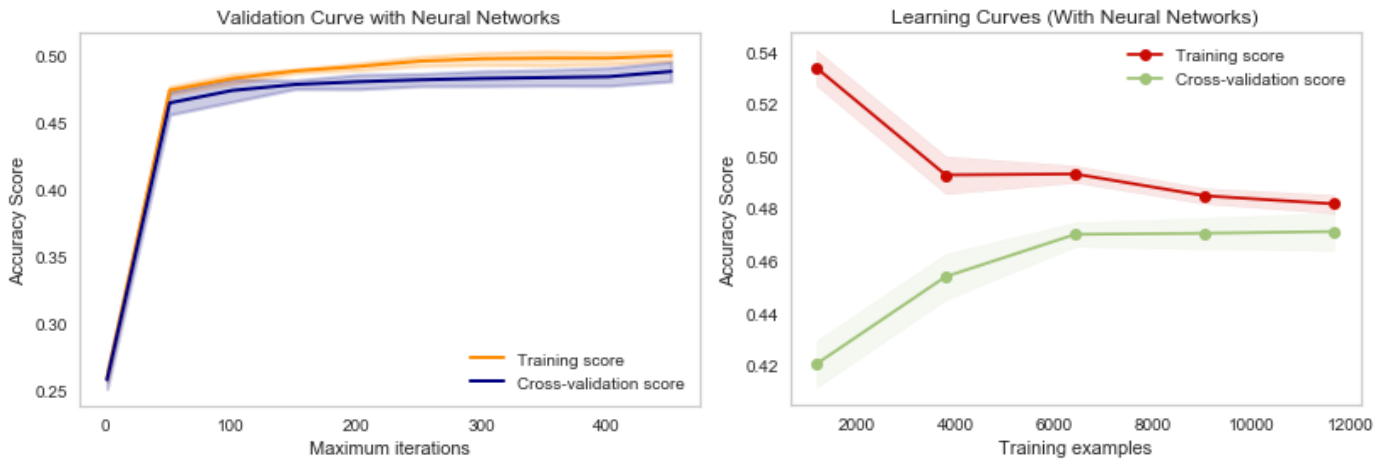
Neural Network with PCA

What if we had more success with applying PCA to the training set of one of our datasets, running a neural network learner on the newly projected data, and testing against the projected test set? For this purpose, we will apply PCA to our Avocado training set we split off at the beginning of the notebook.



It looks like we can still explain >90% of variance with 11 components. Copying our code from Assignment 1, we can use GridSearchCV to refit hyperparameters for our neural network and calculate validation and learning curves.

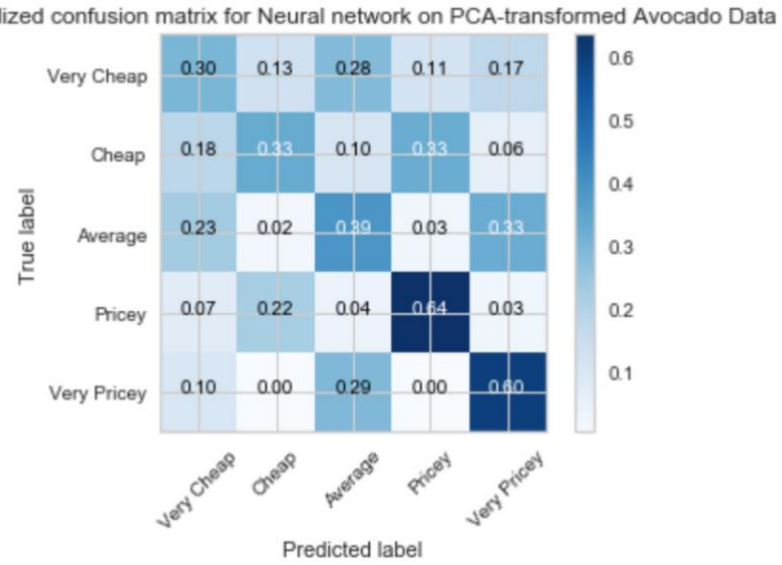
GridSearchCV recommended alpha = 0.001, a hidden layer of 15 nodes, using solver 'lbfgs.' We calculate a validation curve using those hyperparameters:



At this point, it's clear that this neural network does not perform as well as on the PCA-transformed dataset as one fit on all the original Avocado dataset features from Assignment 1, but the algorithm's accuracy score on 5-fold CV is an impressive improvement over its unsupervised clustering algorithm counterparts. The training and CV accuracy scores converge at around 48%, which means we are not overfitting. If anything, the neural network was high bias, low variance, meaning our model is too simple (underfitting). Regardless, predicting roughly half of the CV dataset's

labels correctly is a significant improvement over the ARI of 0.150 for *K*-means and ARI of 0.013 for GMM on PCA-transformed data, which was little better than random assignment.

Training the neural network on the entire PCA-transformed dataset and testing it:



	precision	recall	f1-score	support
Very Cheap	0.34	0.30	0.32	744
Cheap	0.46	0.33	0.39	725
Average	0.34	0.39	0.37	712
Pricey	0.58	0.64	0.61	745
Very Pricey	0.51	0.60	0.55	724
micro avg	0.45	0.45	0.45	3650
macro avg	0.45	0.45	0.45	3650
weighted avg	0.45	0.45	0.45	3650

Curiously, the neural network confusion matrix had a checkerboard-like appearance, where the algorithm opted to make label predictions either on the true price range or 1 removed from the true label's neighbors, but rarely on the true label's neighbors themselves. This may be because of the nature of the dataset, where true average prices of avocados were discretized into 5 classes (price ranges), which were sometimes very narrow price ranges because the dataset was skewed right. It may be that the neural network performed worse than in Assignment 1 because it has too few features to learn from after dimensionality reduction. It seems more confident about categorizing avocados into more extreme or median labels but is not as sure about the in-between labels.

Overall, the performance on our PCA-transformed Avocado test dataset is 45% (both precision and recall), quite similar to the predicted performance based on cross-validation. It runs in <5 seconds, so given computational constraints, it seems preferable to reduce complex datasets using PCA before fitting a neural network rather than fitting on clustering algorithms.

Neural Networks, using Clustering Algorithms as Dimensionality Reduction

Finally, we can try substituting PCA with both of our clustering algorithms on the same Avocado problem, using the clustering algorithms' outputs as new features to input to the neural network instead of PCA's. We've already determined *k* earlier and need to provide input features to the neural network that can suggest the correct number of possible output classes, so we will use the same number of clusters / components as classes.

Our *K*-means will transform the raw input features for each data point into cluster-distance space as new features for input into the neural network, while the GMM will provide the posterior probability of belonging to each cluster.

```
print(avocadoKMeans_clusterdist[:20])
```

```
[[  2.405  21.808   1.913   4.362  29.545]
 [  1.916  21.257   2.428   3.658  28.979]
 [  2.402  21.801   1.915   4.37   29.546]
 [  1.899  21.664   2.353   4.037  29.45 ]
 [  2.426  21.798   1.93    4.336  29.536]
 [  1.931  21.593   2.378   3.918  29.313]
 [  2.421  21.785   1.944   4.397  29.563]
 [  1.903  21.552   2.372   3.914  29.277]
 [  1.918  21.572   2.394   4.004  29.351]
 [  2.003  21.478   2.475   3.901  29.159]
```

```
print(avocadoGMM_componentprob[:20])
```

```
[[ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.]
```

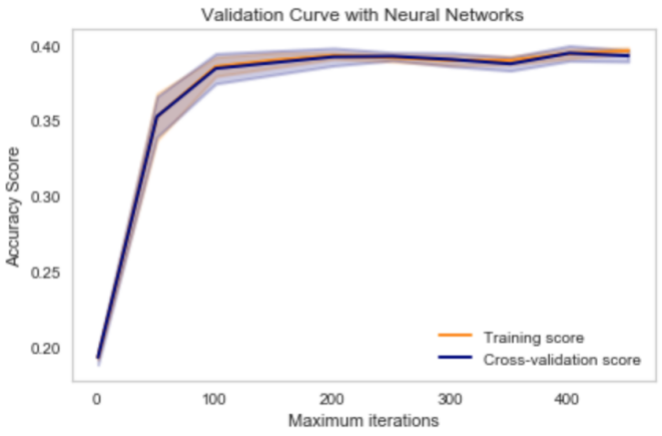
Cluster distance seems to be a straightforward transformation: how far away from each cluster center is a given data point? Though the clusters may be overlapping and have poor accuracy, the neural networks have access to the ground truth and might be able to map the clustering algorithms' outputs to the correct label.

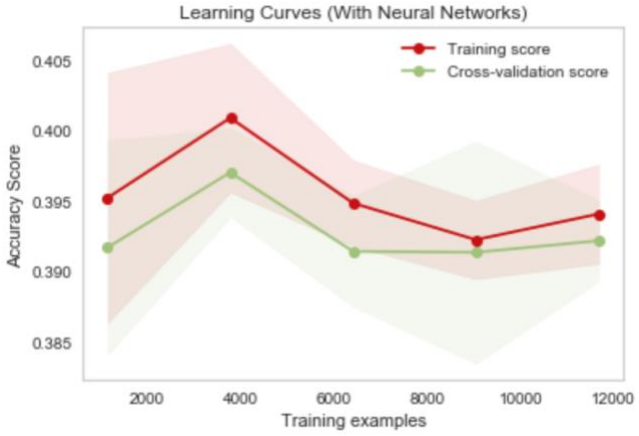
It seems that the GMM is very certain about the posterior probabilities of each component for the data it has already trained on—I suspect that this may translate to less nuanced information for the neural network to sift through.

We'll go ahead and just use GridSearchCV to fit hyperparameters for the neural networks on the respective sets of clustering-transformed input features.

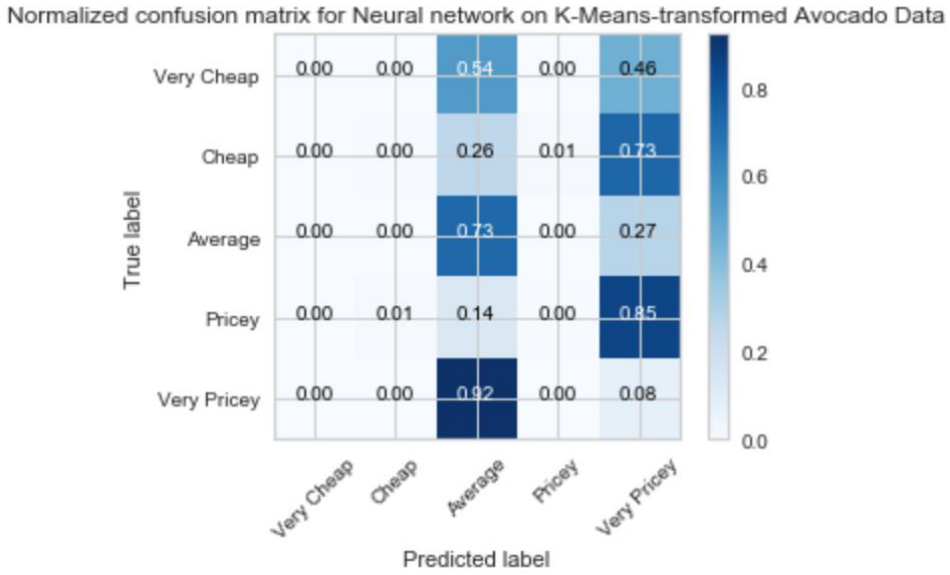
K-means

GridSearchCV and validation curves lead us to use the same neural networks as in the past (ReLU activation, single layer 15 hidden nodes, 400 max iterations, 'lbfgs' solver). The cross-validation accuracy hovers at around 39%.





There’s not much we can do but go ahead and test it.

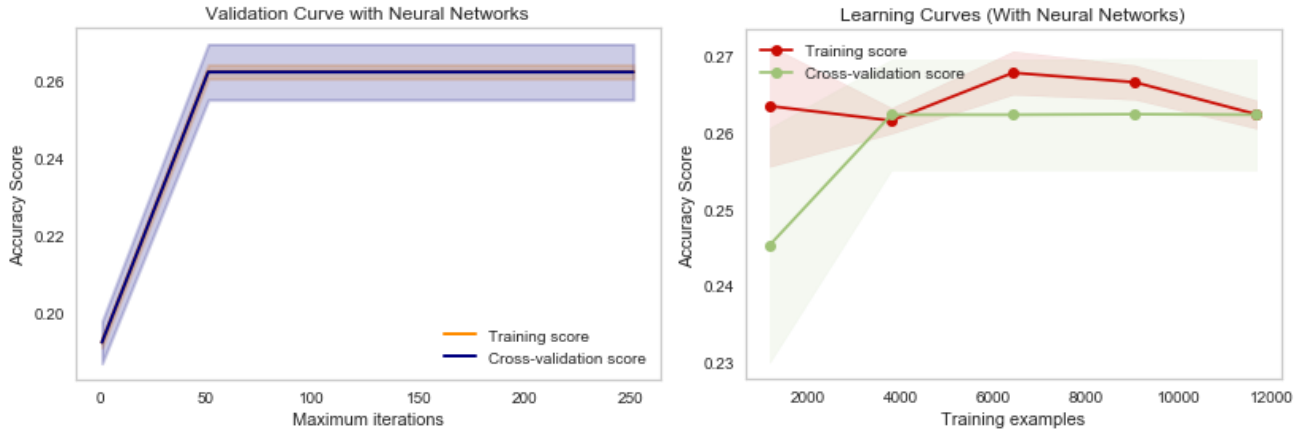


	precision	recall	f1-score	support
Very Cheap	0.00	0.00	0.00	744
Cheap	0.33	0.00	0.01	725
Average	0.28	0.73	0.40	712
Pricey	0.33	0.00	0.01	745
Very Pricey	0.03	0.08	0.05	724
micro avg	0.16	0.16	0.16	3650
macro avg	0.19	0.16	0.09	3650
weighted avg	0.19	0.16	0.09	3650

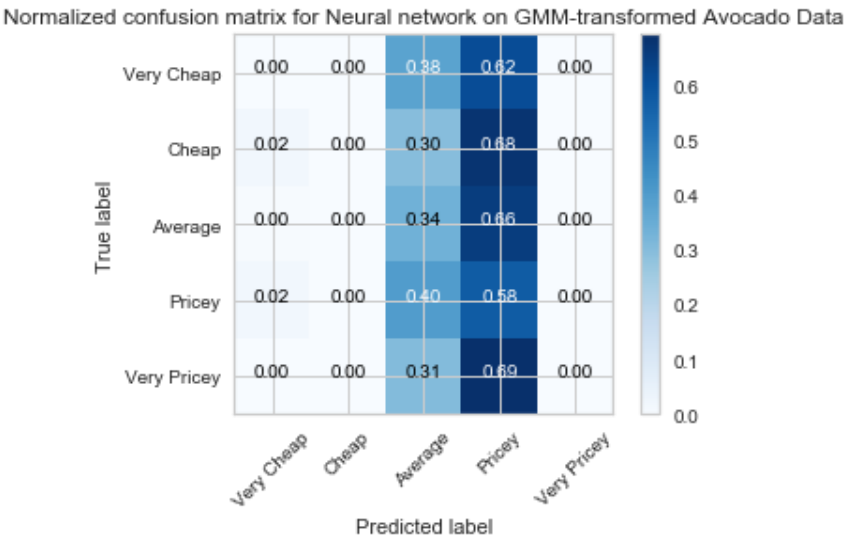
The results are bad. The neural network essentially predicts only one of two labels: 'Average' or 'Very Pricey.' This makes sense seeing as how much difficulty both clustering algorithms had in sussing out the different labels—a given point had similar distances to the centers of several clusters because of how much the clusters overlapped. Our model was both high in variance and high in bias. The overall out-of-sample accuracy is <20%, meaning the classifier performed worse than guessing. It doesn't even run any faster than the PCA neural network.

GMMs

Same activation function and solver for the neural network as above.



This looks just as ugly as *K*-means if not worse. I double checked that I hadn't made a typo, but it looks like our cross-validation performance is already in the gutter.



	precision	recall	f1-score	support
Very Cheap	0.03	0.00	0.00	744
Cheap	0.00	0.00	0.00	725
Average	0.19	0.34	0.24	712
Pricey	0.18	0.58	0.28	745
Very Pricey	0.00	0.00	0.00	724
micro avg	0.18	0.18	0.18	3650
macro avg	0.08	0.18	0.10	3650
weighted avg	0.08	0.18	0.10	3650

Failure. Unfortunately, much of the lackluster performance in the clustering algorithms carried over to the neural network. Similar to the neural network fed from a *K*-means transformed dataset, the one built on GMM classified nearly everything in one of two classes: either 'Average' or 'Pricey.' My hypothesis regarding the GMM's extremely high posterior probabilities was proven correct: the trained model never had any nuanced, diverse samples to learn from, and thus never diverged from guessing 2 principal classifications. The accuracy is <20%, worse than guessing.