UDACITY

PROJECT

Creating Customer Segments

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW | NOTES

Meets Specifications

SHARE YOUR ACCOMPLISHMENT

Perfect submission! 🏆
Exceptional coding work, and analysis demonstrates a pretty fine understanding of clustering in general 😄

Note that I have been a bit lenient at a few places (in particular, Q5), so please do go through the remarks and the reading material provided to further improve your understanding.

Good luck for the next project! 👍

PS: `seaborn` *module makes for some awesome visualizations, but is in no way obligatory for this project* 😉

Data Exploration

Three separate samples of the data are chosen and their establishment representations are proposed based on the statistical description of the dataset.

Awesome work predicting the establishments represented by the sample points by taking quartiles as a reference!
As we see later, the features' distribution is highly *right-skewed*, therefore, the quartiles serve as a pretty appropriate reference for comparison of sample points.

Code tip:

You can use the following code to plot the percentile values for sample points:

```
percentiles_data = 100*data.rank(pct=True)
percentiles_samples = percentiles_data.iloc[indices]
display(percentiles_samples)
```

A prediction score for the removed feature is accurately reported. Justification is made for whether the removed feature is relevant.

Your conclusion regarding the relevance of `Delicatessen`, based on the r^2-score obtained, is absolutely correct! The low/negative prediction score for a feature means that the values of that feature cannot be predicted well by the other features in the dataset and therefore, the feature is not redundant and may contain useful information not contained in other features.

Suggestion:

Your choice of random states can have a huge influence on the R^2-score obtained, which could, in turn, have an influence on your interpretation of the relevance of a feature. To mitigate this, you can average the prediction scores over many iterations, say 100, without setting any of the random states.

Student identifies features that are correlated and compares these features to the

predicted feature. Student further discusses the data distribution for those features.

Remarks:

- The most significant correlation is definitely between `Grocery` and `Detergents_Paper`. `Milk` is also correlated with both these features, but the correlation is relatively mild. For the exact values, you can use `data.corr()` to get a matrix of correlations for all feature pairs.
- The lack of significant correlation with `Delicatessen` nicely aligns with your interpretation of its relevance in the previous question.
- Well done remarking that the features' distribution is not normal, but positively skewed! Clustering algorithms discussed in this project work under the assumption that the data features are (roughly) normally distributed. Significant deviation from zero skewness indicates that we must apply some kind of normalisation to make the features normally distributed.

Data Preprocessing

Feature scaling for both the data and the sample data has been properly implemented in code.

Student identifies extreme outliers and discusses whether the outliers should be removed. Justification is made for any data points removed.

Remarks:

- Good work, correctly identifying the Tukey outliers for more than one features!
- You make an excellent point regarding the impact of outliers on clustering algorithms because of the distance averaging involved. In our context, `cluster_centers` turn out to be relatively insensitive to the choice of outliers, unless the outliers end up forming a different cluster by themselves, which could indeed happen if they are not removed at all.
- It is important here to achieve a compromise between removing outliers to get better clustering results, and not removing too much useful information. Removing all the Tukey outliers, even those for only one feature, effectively removes 10% of samples from our dataset and is generally not recommended without a strong justification. Therefore, one might choose to remove only the "extreme" outliers, where "extreme" is reasonably defined, for example, the outliers for more than one features, and/or outliers obtained by increasing the step size in Tukey's method.

Code tip:

You can also use the concept of counter to identify the multiple-feature Tukey outliers:

```
from collections import Counter
outliers_all = []
for feature in log_data.keys():
      ....
      feature_outliers = log_data[~((log_data[feature] >= Q1 - step) & (l
og_data[feature] <= Q3 + step))]
      outliers_all.extend(list(features_outliers.index.values))

# multiple feature outliers
outliers_mult = [item for item, count in
Counter(outliers_all).iteritems() if count > 1]
```

Feature Transformation

The total variance explained for two and four dimensions of the data from PCA is accurately reported. The first four dimensions are interpreted as a representation of customer spending with justification.

Remarks:

- Nice work elaborating on each dimension, but it would be better if after remarking the relative weights given to the different features in each dimension, you could surmise what kind of customers might be well-separated along this dimension. For example, a dimension giving relatively high (positive or negative) weights to `Fresh`, `Milk`, `Frozen` and `Delicatessen` would likely separate out the restaurants from the other types of customers.
- It is important to remark that `Fresh` and `Delicatessen` have opposing signs in Dimension 3. Such a dimension would differentiate very different set of customers compared to a dimension in which `Fresh` and `Delicatessen` have large absolute weights of the same sign.
  It doesn't matter, as explained in the next remark, whether `Fresh` has a positive sign and `Delicatessen` negative, or vice versa, but it matters that they have opposing signs.
  Similar logic applies to Dimension 4, as well.
- The sign of a PCA dimension itself is not important, only the relative signs of features forming the PCA dimension are important. In fact, if you run the PCA code again, you might get the PCA dimensions with the signs inversed. For an intuition about this, think about a vector and its negative in 3-D space - both are essentially representing the same direction in space. You might find this exchange informative in this context.

PCA has been properly implemented and applied to both the scaled data and scaled sample data for the two-dimensional case in code.

Clustering

The Gaussian Mixture Model and K-Means algorithms have been compared in detail. Student's choice of algorithm is justified based on the characteristics of the algorithm and data.

Good job comparing GMM and KMeans!
From a practical standpoint, the main criteria for deciding between these two algorithms are the speed v/s second order information (confidence levels) desired and the underlying structure of our data.

Regarding your choice of algorithm:

Both the algorithms will do fine here, although considering the fact that there are no visually separable clusters in the biplot, one might, indeed, prefer the soft-clustering approach of GMM, particularly since the dataset is quite small and scalability is not an issue. For large datasets, an alternative strategy could be to go with the faster KMeans for preliminary analysis, and if you later think that the results could be significantly improved, use GMM in the next step while using the cluster assignments and centres obtained from KMeans as the initialisation for GMM. In fact, many implementations of GMM automatically perform this preliminary step for initialisation.

I provide below some citations which might prove useful, if you would like to go deeper into the dynamics of these algorithms:
http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/mixture.html
http://www.nickgillian.com/wiki/pmwiki.php/GRT/GMMClassifier
http://playwidtech.blogspot.hk/2013/02/k-means-clustering-advantages-and.html
http://www.improvedoutcomes.com/docs/WebSiteDocs/Clustering/K-Means_Clustering_Overview.htm
http://stats.stackexchange.com/questions/133656/how-to-understand-the-drawbacks-of-k-means
http://www.r-bloggers.com/k-means-clustering-is-not-a-free-lunch/
http://www.r-bloggers.com/pca-and-k-means-clustering-of-delta-aircraft/
https://shapeofdata.wordpress.com/2013/07/30/k-means/
http://mlg.eng.cam.ac.uk/tutorials/06/cb.pdf

Several silhouette scores are accurately reported, and the optimal number of clusters is chosen based on the best reported score. The cluster visualization provided produces the optimal number of clusters based on the clustering algorithm chosen.

Remarks:

- In your case, `number of clusters = 3` definitely gives the best silhouette score among the many considered, but this is where your choice of random state plays a huge role. Please re-run your code without setting the random state in GMM, or setting it to 1. What is the optimal number of clusters that you get? :)
- Your choice of outliers can also play a decisive role. For example, repeat the analysis without removing any outlier. For most of the random states, `number of clusters = 3` would give the best silhouette score.
- Due to this high variability of silhouette scores, it is recommended not to decide the optimal number of clusters solely on the basis of scores obtained. To verify if your choice based on the highest silhouette score is reasonable, you can check how ***balanced*** are the clusters obtained for different values of `number of clusters`, using the code given at this link.
  Remark that in certain cases, you can even choose a value for `number of clusters` which gives a sub-optimal score. For example, in the link provided, 2 is not considered optimal, despite having a better Silhouette score, because it doesn't result in ***balanced*** clusters, while 4 does.
- From sklearn documentation, the Silhouette Coefficient is calculated using the mean intra-cluster distance and the mean nearest-cluster distance for each sample. Therefore, it makes sense to use the same distance metric here as the one used in the clustering algorithm. This is `Euclidean` for KMeans (default metric for Silhouette score) and `Mahalanobis` for general GMM.
- For GMM, BIC could sometimes be a better criterion for deciding on the optimal number of clusters, since it takes into account the probability information provided by GMM. I leave you to experiment with this.

The establishments represented by each customer segment are proposed based on the statistical description of the dataset. The inverse transformation and inverse scaling has been properly implemented and applied to the cluster centers in code.

Awesome work comparing with the quartiles to propose the establishments represented by the two clusters! You can easily tweak the code provided in the comment to Q1, to get the percentile values for `true_centers`.

Sample points are correctly identified by customer segment, and the predicted cluster for each sample point is discussed.

Conclusion

Student correctly identifies how an A/B test can be performed on customers after a change in the wholesale distributor's service.

Excellent! You have correctly identified the key point here which is to test each segment independently, although to be technically correct, you must also form a control group for each segment, corresponding to the test group. For an A/B test to be effective, the experiment group (A) has to be highly similar to the control group (B), before the treatment is applied to the experiment group. If they are dissimilar to each other, then the result of the A/B test might be due to some variable other than the variable being tested.

I give below a few links which might help remove misconceptions on this topic, if any:
https://www.quora.com/When-should-A-B-testing-not-be-trusted-to-make-decisions/answer/Edwin-Chen-1
http://multithreaded.stitchfix.com/blog/2015/05/26/significant-sample/
http://techblog.netflix.com/2016/04/its-all-about-testing-netflix.html
https://vwo.com/ab-testing/
http://stats.stackexchange.com/questions/192752/clustering-and-a-b-testing

Student discusses with justification how the clustering data can be used in a supervised learner for new predictions.

Comparison is made between customer segments and customer 'Channel' data. Discussion of customer segments being identified by 'Channel' data is provided, including whether this representation is consistent with previous results.

Remark that the `channel_visualisation` validates, to some extent, the choice of using GMM, as the clusters do have a fair amount of overlap in reality. Although a perfect classification is not possible to achieve, soft clustering gives us confidence levels in our predictions, which would understandably be low at the boundary between two clusters.

As for the discrepancy in the number of clusters, again, your choice of random state has played a decisive role here.

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ