UDACITY

‹ Return to "Machine Learning Engineer Nanodegree" in the classroom

DISCUSS ON STUDENT HUB

Creating Customer Segments

REVIEW

HISTORY

Meets Specifications

Perfect submission! 🏆

This is one of the most original attempts made for this project, and I really enjoyed reviewing it! In particular, the analysis demonstrates exceptional coding work, and a pretty fine understanding of clustering in general 😉

Good luck for the next 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.

Good work predicting the establishments represented by the sample points based on the comparison of their features to the dataset mean.

Suggestion:

As we see later, the features' distribution is highly *right-skewed*, therefore, the median would probably serve as a better reference than mean. In fact, I would recommend comparing to the quartiles to get a better idea of the nature of the establishments represented.

Code tip:

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

```
import seaborn as sns

percentiles_data = 100*data.rank(pct=True)
percentiles_samples = percentiles_data.iloc[indices]
sns.heatmap(percentiles_samples, annot=True)
```

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

Your interpretation of the relevance of `Grocery`, based on the prediction score obtained, is absolutely correct! The R^2 score obtained is high, but probably not high enough to justify its removal. However, if we ever need to drop a feature to make the dataset more manageable, `Grocery` could be a good candidate - a feature that can be predicted, at least partly, from other features would only give us a marginal 'information gain'.

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 R^2-scores over many iterations, say 100, without setting any of the random states.

In particular, such an averaging would lead to a much higher score for `Grocery` (>0.6), which would, in turn, have an influence on your interpretation of its relevance.

Student identifies features that are correlated and compares these features to the predicted feature. Student further discusses the data distribution for those features.

Significant correlations and interpretation:

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.

Note that the high correlation between `Grocery` and `Detergents_Paper` contradicts your interpretation from the previous question. We do get additional information if we keep both of `Grocery` and `Detergents_Paper` in the dataset, but we can drop one just in case we severely need to reduce the dimensionality of our feature space. Later, we will see a better way of reducing the dimensionality of our dataset - PCA.

Marginal distribution of features:

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:

- Awesome coding 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 the context of this project, apart from causing some deviation in the cluster boundaries and centers, one might even be persuaded by the subsequent silhouette analysis to consider an additional cluster just to accommodate the outliers.
- It is also important to achieve a compromise between removing outliers to get better clustering results, and not removing too much useful information. The set of all the Tukey outliers, forming around 10% of our dataset, is too huge to be removed without a particularly strong reason. 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.

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 the PCA dimensions and interpreting them as a representation of customer spending. Two main takeaways:

- A high/low (absolute) value along the PCA dimension can help differentiate between different types of customers. 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.

- A corollary of the above remark is that the sign of a PCA dimension itself is not important, only the relative signs of features forming the PCA dimension are important. In fact, on running the PCA code again, one might get the PCA dimensions with the signs inversed. For an intuition about this, it is helpful to 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.

The following links might be of interest in the context of this question:
https://onlinecourses.science.psu.edu/stat505/node/54
http://setosa.io/ev/principal-component-analysis/

Code tip:

Good work getting the correct values of cumulative explained variance for the first two and four dimensions. You could also use the following code to compute these values:

```
print pca_results['Explained Variance'].cumsum()
```

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.

Awesome coding work! Indeed, `number of clusters = 2` gives the best silhouette score among the many considered!

Important remark regarding the choice of outliers:

This is one place where your choice of outliers plays a huge role. For example, repeat the analysis without removing any outlier. What is the optimal number of clusters that you get?

Miscellaneous remarks:

- If you want to get more support for your results obtained using silhouette analysis, one way is to check how ***balanced*** are the clusters obtained from 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.

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

One interesting point to note from the `cluster_visualization` plot is that the two clusters are essentially separated by a value on the first PCA dimension, which we saw earlier is predominantly a combination of `Detergents_Paper`, `Grocery` and `Milk`. The rest of the features, which figure prominently only in the second PCA dimension, don't really matter!

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 conduct the A/B test on each segment independently, since in A/B testing, everything besides the testing parameter should remain as similar as possible for both the experiment (A) and the control (B) groups, so that we can study the change in behavior caused by the testing parameter.

Here are a few links for further reading on A/B testing:
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.

Important point to remark here is that the `channel_visualization` 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.

Code tip:

You can calculate the accuracy score for clustering using the following code:

```
channel_labels = pd.read_csv("customers.csv")["Channel"]
channel_labels = channel_labels.drop(channel_labels.index[outliers]).reset_index(dro
p = True) - 1
# channel_labels = abs(channel_labels -1)
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(channel_labels,preds)
```

Note that I've subtracted 1 from `channel_labels`, because the given `channel_labels` are `1` and `2`, while our cluster-labels are `0` and `1`.
Also, note that the assignment of labels - `0` and `1` - in the clustering algorithm is completely arbitrary. Therefore, you might have to keep or remove `channel_labels = abs(channel_labels -1)` in the above code, to ensure that the cluster and channel labels are "compatible".

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review