

## PROJECT

## Creating Customer Segments

A part of the Machine Learning Engineer Nanodegree Program

## PROJECT REVIEW

## NOTES


SHARE YOUR ACCOMPLISHMENT!  

## Meets Specifications

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, so please do go through the remarks and the reading material provided to further improve your understanding.

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.

Excellent work predicting the establishments represented by the sample points based on the offset of their features from 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.

Frozen is necessary feature to identify the customer habits.

This is 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.

On the other hand, a feature that can be predicted from other features would not really give us much additional information and thus, would be a fit candidate for removal, if we ever need it to make the dataset more manageable.

## Suggestions:

- Good job fixing the `random_state` while splitting the dataset. It would nice to do the same for `Regressor` as well, so that we obtain the same score for every run of the program.
- Alternatively, you can average the prediction scores over many iterations, without setting any of the `random_state(s)`, so as to mitigate the impact of a particular choice of `random_state(s)` on the score obtained.

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

## 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 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 also important 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.

## 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:

- You can use the following code to compute the cumulative explained variance for the first two and four dimensions:

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

- Nice work elaborating on the first four dimensions and interpreting them as a representation of customer spending. Remark, however, that any PCA dimension, in itself, does not represent a particular type of customer, but a high/low 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.
- Also note 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, 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.

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/>

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.

Your decision to use KMeans is absolutely fine. In my opinion, it is a good strategy 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.

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.

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.

However, your statement,

To perform the A/B Test wholesale dealer can split each cluster in two sub sets.

implies that you intend to implement the change to half the population of each segment, which is not at all a good idea. A/B testing is actually an experiment performed on small samples from the population, just large enough to get statistically significant results.

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.**

Indeed, KMeans does a pretty decent job here :) Even when the data is not linearly separable, as is generally the case in real world, KMeans can still give surprisingly good results, and is therefore, a good first algorithm to use for a lot of clustering problems.

GMM could also have been a good choice here as the scalability is not an issue and the clusters do have a fair amount of overlap in reality. Although a perfect classification is not possible to achieve even with GMM, 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(drop
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](#)

---

[Student FAQ](#)