UDACITY

PROJECT

Creating Customer Segments

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

1 SPECIFICATION REQUIRES CHANGES

Hello Student,
You almost get it done, keep it UP!
There is only one minor mistake you need to amend, I believe you already understand most of the concept in 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.

Well done commenting on the different types of establishments the three customers could represent.

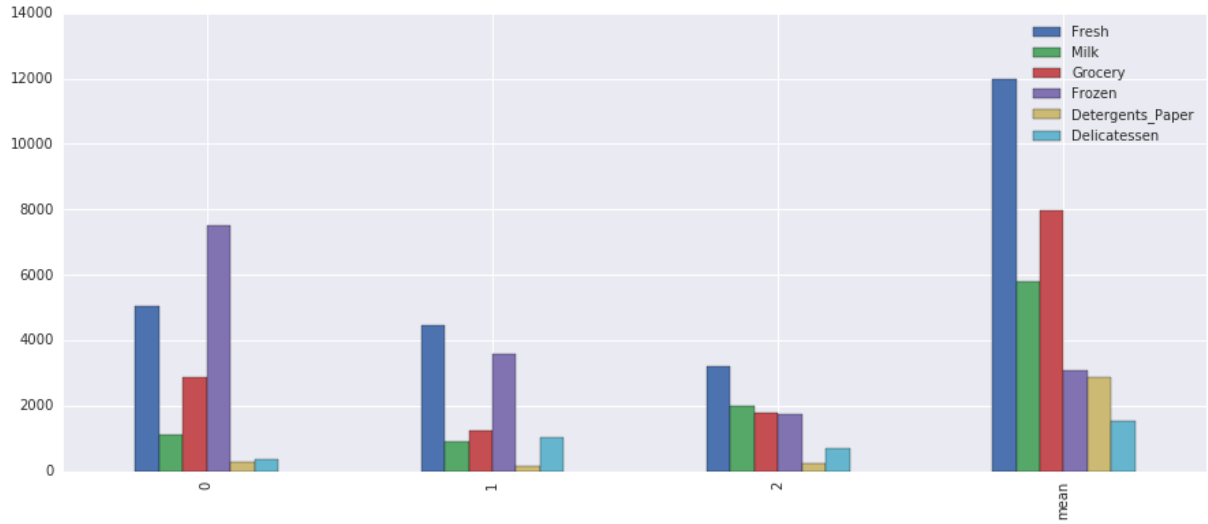
Pro Tips:

COMPARING TO DATASET AVERAGE

You could quickly draw a bar plot to visualise the amount of each product purchased for each sample, together with the dataset mean.

```
# Import Seaborn, a very powerful library for Data Visualisation
import seaborn as sns
samples_bar = samples.append(data.describe().loc['mean'])
```

```
samples_bar.index = indices + ['mean']
_ = samples_bar.plot(kind='bar', figsize=(14,6))
```



This will make comparing the three different sample points with each other much easier.

✓

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

Great analysis to identify that the amount of `Detergents_Paper` purchased is not necessary to identify specific customers!

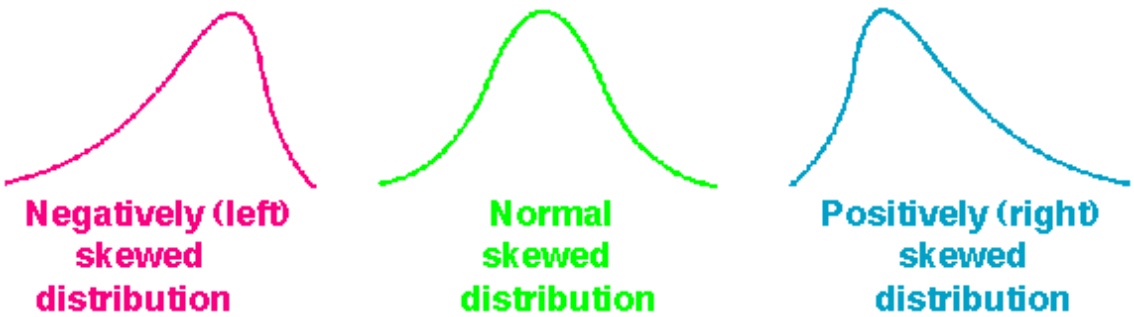
✓

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

You did great identifying the features that have correlation from the dataset.

Suggestions and Comments:

- Kindly note that the distribution is skewed to the right (most points lie to the left of the graph). The graph below interprets how to quickly judge the skewness of a distribution:



- As you can see from the scatter plot, there are a number of outliers for most of the features.
- Also, the median falls below the mean, and there are a large number of data points near 0.

Pro Tips:

```
import seaborn as sns
corr = data.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("white"):
ax = sns.heatmap(corr, mask=mask, square=True, annot=True, cmap='RdBu')
```

Data Preprocessing

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

Nice work implementing feature scaling :)

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

Great job catching all of the outliers, and giving justifications on why you think they are outliers and why they should be removed.

Pro Tip:

- Below are some code snippets which can help you identify some outliers. You can use these code snippets together with the starter code provided in the notebook to catch outliers
- This code should be fast and efficient because it is vectorised and should scale well if the size of the dataset increases.

1. Create a numpy array to catch all of the outliers which occur for any feature in the loop.

```
# Create an array of all outliers
all_outliers = np.array([], dtype='int64')
```

2. Catch all of the outliers inside the loop using the `np.append` function. The reason `numpy` is used is that, `numpy` is much faster than pure Python, especially for very large datasets. You may not notice the performance improvements at the moment, but as you progress and deal with larger dataset, you'll realise `numpy` is much faster than pure Python.

```
all_outliers = np.append(all_outliers, outlier_points.index.valu
es.astype('int64'))
```

3. Outside of the loop, and after looping, count all of the unique elements in the `all_outliers` array.

```
# Count the unique elements in the all_outliers array
all_outlier, indices = np.unique(all_outliers, return_inverse=Tr
```

```
ue)
counts = np.bincount(indices)
```

4. Finally, obtain all of the outliers using the counts

```
# Obtain outliers using the counts
outliers = all_outlier[counts>1]
print outliers
```

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.

Nice attempt improving your answer using the previous reviewer's feedback! Below are some suggestions and comments for you:

Required:

Nice discussion on the consumption weight of the first four dimensions, however, could you suggest which type of customer would they be?

- For example, for a particular dimension, you could add:

  > This dimension is best categorized by customer spending on retail goods.

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

Nice job here!

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.

Great work elaborating on GMMs and K-means, and giving a solid reason as to why you chose [K-means] algorithm for this particular problem.
You might want to provide some citations and reference for your work to make it more credible.
Below are some of my comments, feedback, and suggested reading:

Gaussian Mixture Models

- You did well on giving the advantages of Gaussian Mixture Model.

SUGGESTED READING:

- If you feel as going deeper with regards to Gaussian Mixture Models, check out the following links:
  - http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/mixture.html
  - http://www.nickgillian.com/wiki/pmwiki.php/GRT/GMMClassifier
  - http://scikit-learn.org/stable/modules/mixture.html#gmm-classifier

K-Means Clustering

- Your description on the advantages of K-means is very explicit

SUGGESTED READING:

- Check out these links for even more thorough explanations of K-Means Clustering:
  - http://playwidtech.blogspot.hk/2013/02/k-means-clustering-advantages-and.html
  - https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algorithm
  - 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

Reason for choice of Algorithm:

- Well done here again! Please see the links below for some more information on how to compare the two algorithms:

SUGGESTED READING:

- https://www.quora.com/What-is-the-difference-between-K-means-and-the-mixture-model-of-Gaussian

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.

Great job identifying the optimal cluster score as 2, and identifying its associated silhouette score

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.

Well done recovering the true centres, and proposing establishments using guidance from the statistical description of the dataset!

Suggestions and Comments:

COMPARING TO DATASET AVERAGE

You could use the following code to quickly print a comparison of your sample points to the dataset central values:

```
display(true_centers - np.around("central_values".values))
display(true_centers - np.around("central_values".values))
```

Please replace the "central_values" in the code with the appropriate value, and you should be able to have an unbiased comparison on which to base your judgement.

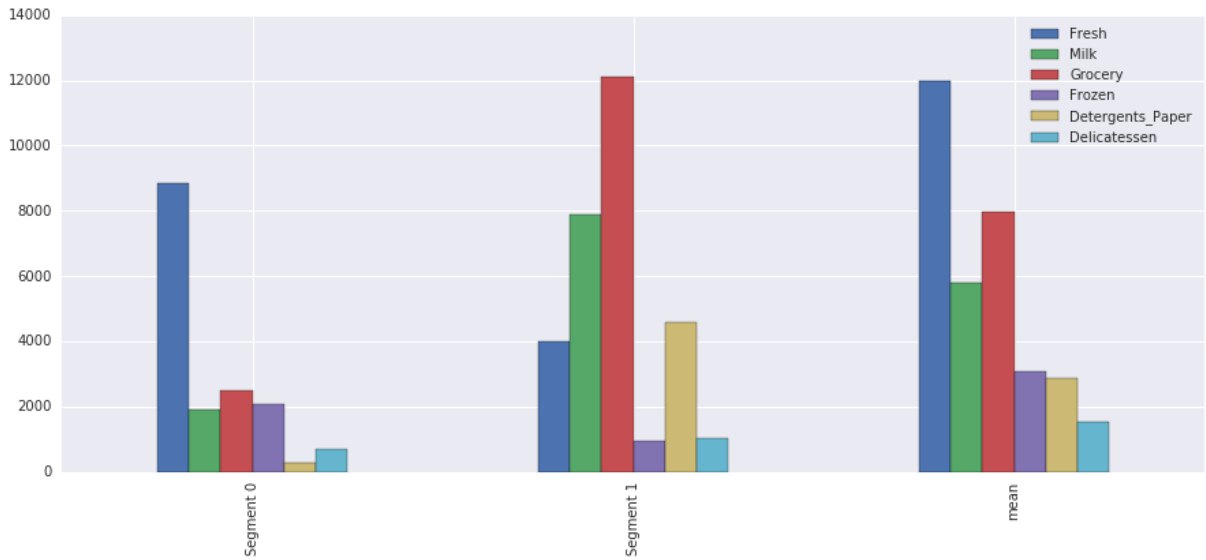
Please see some pro tips for your reference below:

Pro Tips:

COMPARING TO DATASET AVERAGE

You could quickly draw a bar plot to visualise the amount of each product purchased for each `true_center`, together with the dataset mean.

```
# Import Seaborn, a very powerful library for Data Visualisation
import seaborn as sns
true_centers = true_centers.append(data.describe().loc['mean'])
_ = true_centers.plot(kind='bar', figsize=(15,6))
```



This will make comparing the three different sample points with each other much easier.

COMPARING TO DATASET AVERAGE

You could use the following code to quickly print a comparison of your sample points to the dataset central values:

```
display(true_centers - np.around(data.mean().values))
display(true_centers - np.around(data.median().values))
```

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

Nice work identifying the sample points by customer segment, and discussing the predicted cluster for each sample point.

Conclusion

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

Great explanation here!
The key is to conduct the A/B test on only one segment at the time, rather than on the whole dtaset. I like your design of using 2 A/B tests, one per segment.

Suggestions and Comments:

- You can find more information about A/B testing from this link. You can also refer to the Udacity course to get an overview on how A/B testing is conducted (please note that you don't need to take the whole course, just skimming through the lecture videos and overviews should be alright)
- These links were also really helpful to me when picking up A/B testing. Hope they help you too!
  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/

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

Nice job stating that the cluster labels can be used as an **input feature** to a supervised learning algorithm!

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.

Well done comparing the clusters from your algorithm to the customer 'Channel' data!

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

Have a question about your review? Email us at review-support@udacity.com and include the link to this review.

RETURN TO PATH

Rate this review

Student FAQ