UDACITY

Return to Classroom

Creating Customer Segments

REVIEW

HISTORY

Meets Specifications

Awesome job! I left some tips and comments but you're definitely ready to move on in the program. If you feel that this review was valuable, please consider leaving a rating and feedback - it makes a big difference to us as reviewers 😄
Anyway, keep up the great work! I hope you enjoy the rest of the nanodegree.

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 job here!
Your intuitions are backed up with statistical descriptions of the data +1:

TIP

In general, I find it really helpful to visualize sample points when I'm trying to figure out what they represent. You can do this quite simply with the following code 😄

```
import matplotlib.pyplot as plt
import seaborn as sns

samples_for_plot = samples.copy()
samples_for_plot.loc[3] = data.median()
```

```
labels = ['Sample 1','Sample 2','Sample 3','Median']
samples_for_plot.plot(kind='bar')
plt.xticks(range(4),labels)
plt.show()
```

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

Great!
You nailed the key point here - if we can reliably reconstruct a feature from other features, it probably doesn't contain a whole lot of unique information. :+1:

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

Good job addressing the distributions! Often, when we have right-skewed data it approximates a log-normal distribution. It's for this reason that we normalize it in the next section with a log-transformation. However, there are cases where the log-transform doesn't get us as close to normal as we'd like. Another option in that case is the BoxCox transformation.
We can apply it here like so

```
from scipy.stats import boxcox
boxcox_data = data.apply(lambda x: boxcox(x)[0])
pd.scatter_matrix(boxcox_data, alpha = 0.3, figsize = (14,10), diagonal = 'kde');
```

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.

Great job!
You identified the points which are outliers in more than 1 feature :+1:

TIPS ON HANDLING OUTLIERS
Deciding what to do with outliers is never a simple choice. Simply removing them can often delete useful structure in our data, but leaving them in can cause problems in many machine learning algorithms

(including clustering and PCA!). There's no one-size-fits-all solution, so I definitely recommend taking a look at the links below. They do a great job of covering some common cases and how to handle them 😄
http://www.theanalysisfactor.com/outliers-to-drop-or-not-to-drop/
http://unilytics.com/how-to-handle-outliers-in-your-data/

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.

Really fantastic work!
Good job finding the cumulative explained variance. As a tip, we can do this programmatically like so

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

Really good description of the PCA components. It looks like you identified what they mean in terms of correlation as well as how they might represent the degree to which a customer is like a certain kind of customer segment.
It seems like you have a solid handle on PCA, but if you're interested in reading more I encourage you to check out the post below. It really helped me wrap my head around it when I first started
https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

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.

Great work!
In general, K-Means offers better performance if we care about

- Speed
- Scaleability
- Simplicity

Whereas GMM provides more

- Flexibility
- Robustness

The fact that K-Means assumes that all clusters is globular is a pretty enormous assumption, and is always

something we have to take into consideration. GMM is far less rigid in this - it allows these spheres to be stretched and compressed.

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.

Good work!
It looks like you've made some solid statistical arguments to support your choices

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.

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.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review