UDACITY

PROJECT

Creating Customer Segments

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

🏆 CONGRATULATIONS!

Outstanding job with the report, you seem to have grasped all the main concepts of the project.

We only looked at K-Means and GMM clustering here, but another approach you might experiment with is density based clustering (DBSCAN), which can be effective when the number of clusters is unknown.

Congrats again, and keep up the good work with 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.

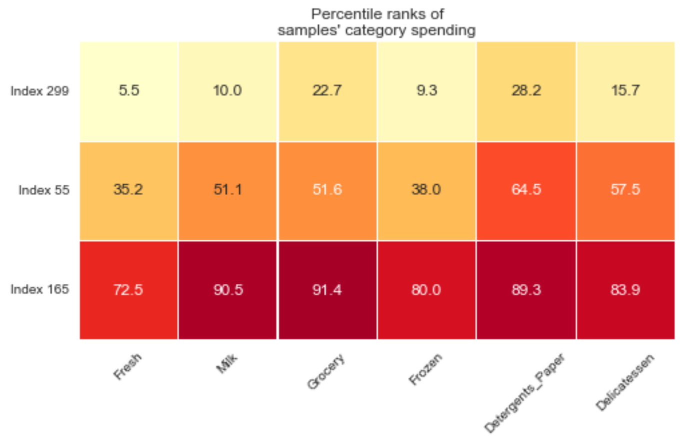
⭐ Terrific discussion of the samples in relation to the overall category spending stats! Getting a closeup view of customers should help as we apply machine learning to optimize food delivery.

Suggestion: look at percentile ranks
As we'll see later, the distribution of our customers' spending data has some large skew, so another thing you could try is looking at the category spending percentile ranks with a heatmap:

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

# look at percentile ranks
pcts = 100. * data.rank(axis=0, pct=True).iloc[indices].round(decimals=3)
print pcts

# visualize percentiles with heatmap
sns.heatmap(pcts, annot=True, linewidth=.1, vmax=99, fmt='.1f', cmap='YlOrRd', square=True, cbar=False)
plt.yticks([2.5,1.5,.5], ['Index '+str(x) for x in indices], rotation='horizontal')
plt.xticks(rotation=45, ha='center')
plt.title('Percentile ranks of\nsamples\' category spending');
```



	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
Index 299	5.5	10.0	22.7	9.3	28.2	15.7
Index 55	35.2	51.1	51.6	38.0	64.5	57.5
Index 165	72.5	90.5	91.4	80.0	89.3	83.9

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

DONE!

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

Great work spotting the correlations and describing the distributions! The feature distributions appear to be skewed Right / Positive (or Log Normal) with a mean greater than the median.

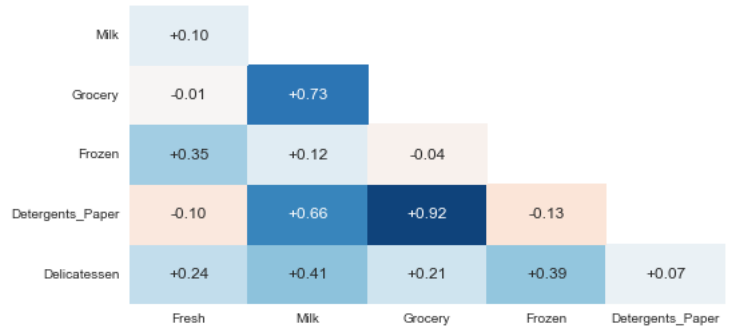
And for a closer look at the correlations we can also use `data.corr()`...

```
# get the feature correlations
corr = data.corr()

# remove first row and last column for a cleaner look
corr.drop(['Fresh'], axis=0, inplace=True)
corr.drop(['Delicatessen'], axis=1, inplace=True)

# create a mask so we only see the correlation values once
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask, 1)] = True

# plot the heatmap
with sns.axes_style("white"):
    sns.heatmap(corr, mask=mask, annot=True, cmap='RdBu', fmt='+.2f', cbar=False)
```



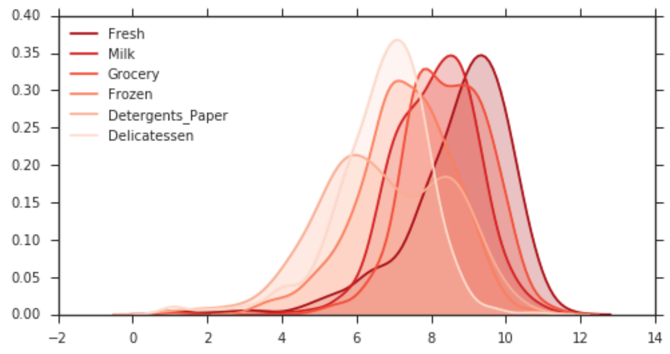
	Fresh	Milk	Grocery	Frozen	Detergents_Paper
Milk	+0.10				
Grocery	-0.01	+0.73			
Frozen	+0.35	+0.12	-0.04		
Detergents_Paper	-0.10	+0.66	+0.92	-0.13	
Delicatessen	+0.24	+0.41	+0.21	+0.39	+0.07

Data Preprocessing

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

To compare the log-transformed feature distributions, you can also plot them on top of each other with a seaborn KDE plot...

```
# set plot style & color scheme
sns.set_style('ticks')
with sns.color_palette("Reds_r"):
    # plot densities of log data
    plt.figure(figsize=(8,4))
    for col in data.columns:
        sns.kdeplot(log_data[col], shade=True)
    plt.legend(loc='best');
```



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

DONE!

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.

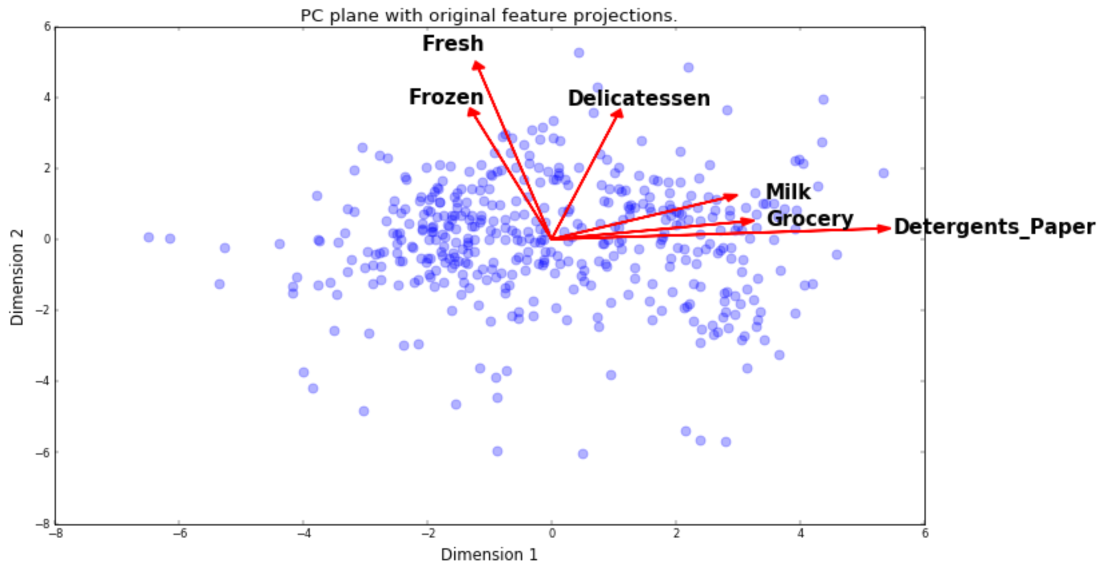
One other thing to note with PCA is that we could actually reverse the signs of the category weights and still be capturing the variance in the data.

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

⭐ Great job implementing the dimensionality reduction!

For another look at our pca-reduced data and the cofficients of the pca vectors, we can use a biplot.

- A biplot is an enhanced scatterplot with each data point represented by its scores along the principal components — the axes in our case are "Dimension 1" & "Dimension 2".
- The arrows show the projection of the original 6 features along the 2 components — this can help us interpret the reduced data, and find relationships between the components & original features.



```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize = (14,8))
# scatterplot of the reduced data
ax.scatter(x=reduced_data.loc[:, 'Dimension 1'],
           y=reduced_data.loc[:, 'Dimension 2'],
           facecolors='b', edgecolors='b', s=70, alpha=0.3)

feature_vectors = pca.components_.T

# we use scaling factors to make the arrows easier to see
arrow_size, text_pos = 7.0, 8.0,

# projections of the original features
for i, v in enumerate(feature_vectors):
        ax.arrow(0, 0, arrow_size*v[0], arrow_size*v[1],
                  head_width=0.2, head_length=0.2, linewidth=2, color='red')
        ax.text(v[0]*text_pos, v[1]*text_pos, good_data.columns[i], color='black',
                 ha='center', va='center', fontsize=18)

ax.set_xlabel("Dimension 1", fontsize=14)
ax.set_ylabel("Dimension 2", fontsize=14)
ax.set_title("PC plane with original feature projections.", fontsize=16);
```

NOTE: The above explanation and code can also be found in the most recently updated project template.

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.

DONE!

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.

Terrific work finding the scores and determining the best number of clusters!

If you wanted, you could also output the scores programmatically with a loop, and make the labelling of segments reproducible by setting a random state on the clusterer (below example uses K-Means)...

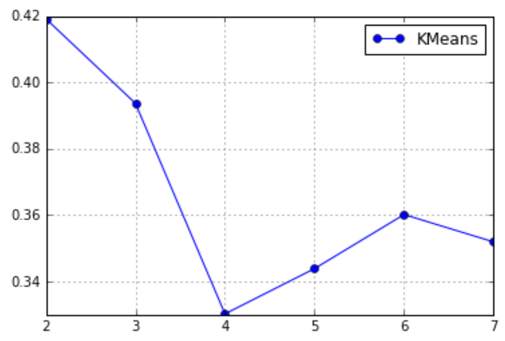
```
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans

# keep the scores for each cluster size
sil_scores = []

# choose range to loop over
sizes = range(10,1,-1)
for i in sizes:
    clusterer = KMeans(i, random_state=0).fit(reduced_data)
    preds = clusterer.predict(reduced_data)
    centers = clusterer.cluster_centers_
    sample_preds = clusterer.predict(pca_samples)
    score = silhouette_score(reduced_data, preds)

    # print the score and append to the list of scores
    print i, 'clusters:', score.round(5)
    sil_scores.append(score)

# plot the scores
import matplotlib.pyplot as plt
plt.plot(sizes, sil_scores, '-o', label=clusterer.__class__.__name__)
plt.grid(True)
plt.legend(loc='best');
```



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.

DONE!

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

DONE!

Conclusion

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

If interested, here's more reading on A/B testing in the real world...

- When A/B testing shouldn't be trusted
- Pitfalls of A/B testing
- Great Data Skeptic podcast on "p-hacking" and other shady practices to watch out for in A/B testing and Conversion Rate Optimization (CRO)

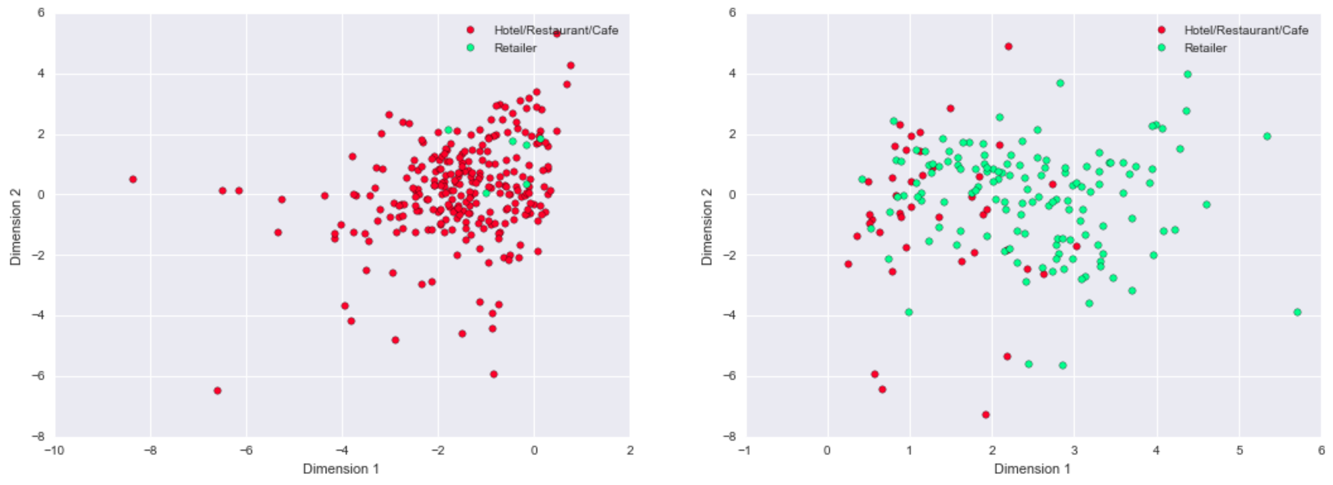
Student discusses with justification how the clustering data can be used in a supervised learner for new predictions.

DONE!

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.

For a look at how well the 'Channel' data and segments are aligned, you can see the 2 clusters from a K-Means implementation plotted separately below (no outliers removed from data).

Although there is disagreement with some data points, the overall alignment is actually pretty good....



DOWNLOAD PROJECT

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