UDACITY

PROJECT

Creating Customer Segments

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Great job with the report! You seem to have grasped all the main concepts of the project. For an excellent guide to approaching almost any machine learning problem, check out this blog post by a Kaggle grandmaster.

Congrats on meeting all the specs, 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, and nice job including some visualizations to make it easier to see how the samples' spending compares. Kudos!

Getting a closeup view of some of our customers should be helpful as we apply machine learning to optimize food delivery.

✓

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

✓

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

In this section we only use the graphical representation of the data to visually inspect for normality, but you could also run a chi-sq test for normality. You can also read more on some of the issues with these tests.

```
from scipy import stats
feature = data['Fresh']
norm_results = stats.normaltest(feature, axis=0)

# low p-value indicates data not likely to come from normal distribution
print norm_results
```

```
NormaltestResult(statistic=274.34162662040899, pvalue=2.6759173855882678e-60)
```

Data Preprocessing

✓

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

BOX-COX TRANSFORMATION

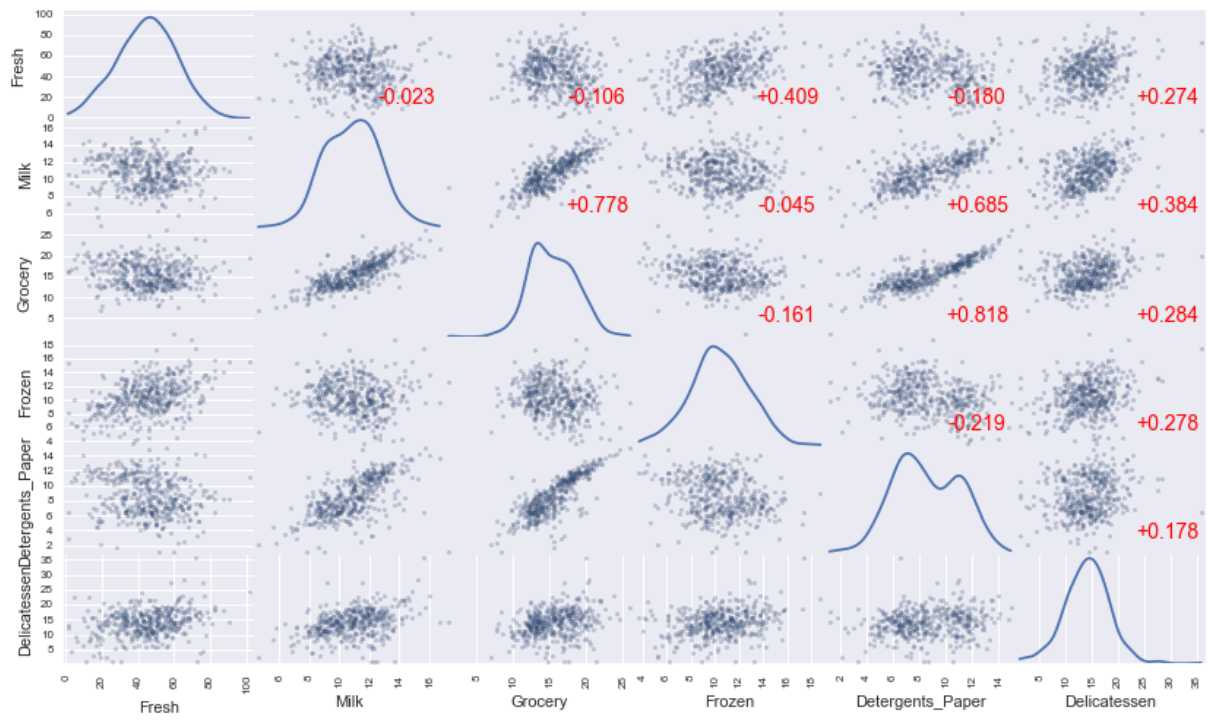
As mentioned in the implementation guidelines, we can also scale features by performing a Box-Cox transformation. You can see an example below and compare the results to our log transformed data (feature correlation values are in red text)...

```
from scipy.stats import boxcox

bc_df = data.copy()
for col in bc_df.columns:
    bc_df[col], _ = boxcox(bc_df[col])

axes = pd.scatter_matrix(bc_df, alpha = 0.3, figsize = (14,8), diagonal = 'kde');
corr = bc_df.corr().as_matrix()

for i, j in zip(*plt.np.triu_indices_from(axes, k=1)):
    axes[i, j].annotate("%+.3f" %corr[i,j], (0.8, 0.2), xycoords='axes fraction', ha='center', va='center',color="red", fontsize=14)
```



✓

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

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.

Excellent work here reporting the cumulative variance and identifying what the category weights in each dimension represent!

The only thing you might do is edit the **"type of customer"** portions -- the dimensions represent *patterns of spending* in the categories, rather than actual *customers* (or segments of customers).

For example, with the 3rd dimension you could adjust slightly to...

> **types of customers that can be observed**: looks like we can distinguish an open market to me (farmers market for example) as it sells Fresh products more than anything else vs customers who **aren't** markets (don't sell Fresh foods, sell mostly pre-cooked Deli foods).

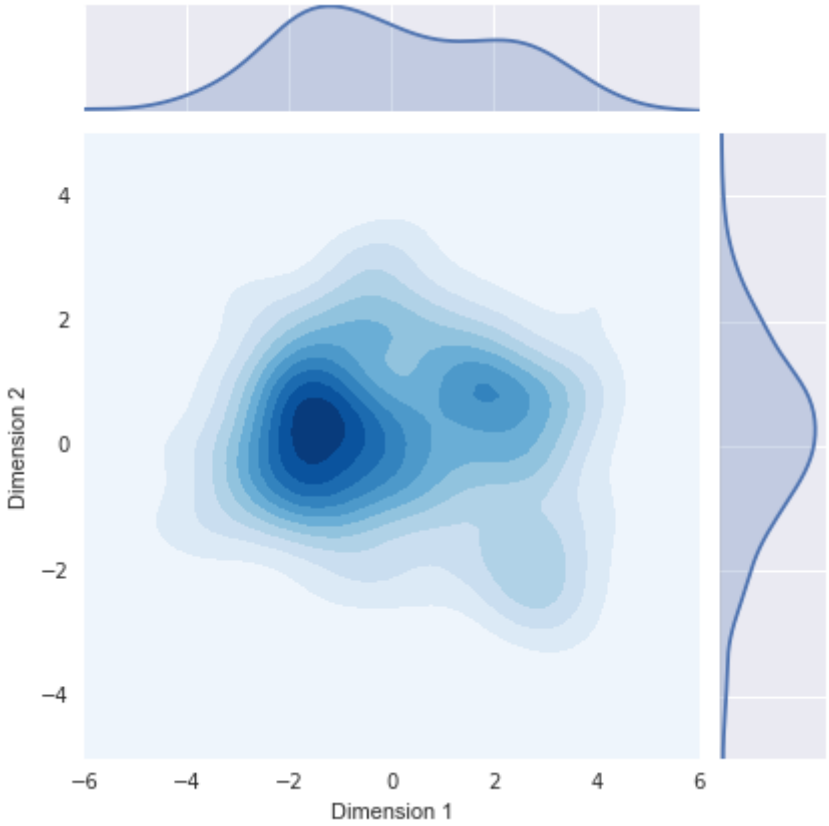
One last thing to note with PCA is that we could actually reverse the signs of the weights and still be capturing the variance in the data.

✓

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

For another look at the pca-reduced data, we can also use a joint grid and see 2 "peaks" in the kernel density plot:

```
import seaborn as sns
g = sns.JointGrid("Dimension 1", "Dimension 2", reduced_data, xlim=(-6,6),
 ylim=(-5,5))
g = g.plot_joint(sns.kdeplot, cmap="Blues", shade=True)
g = g.plot_marginals(sns.kdeplot, shade=True)
```



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.

Note that a big drawback with KMeans is that it assumes the groups come in spherical (globular) shapes that are symmetrical, which don't always occur with real data. With GMM the groups are assumed to be elliptical.

For further reading, here are more thoughts on the best clustering approaches.

✓

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.

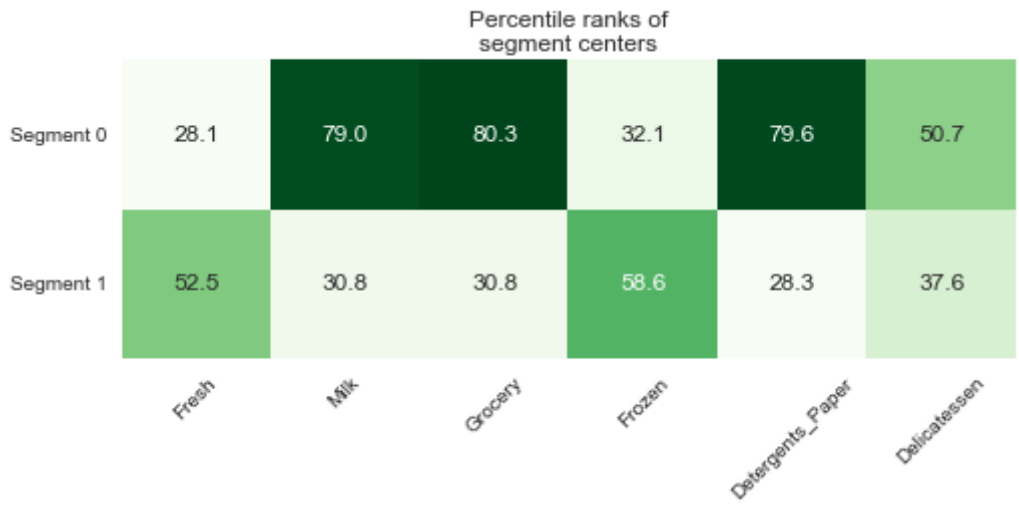
The establishments represented by each customer segment are proposed based on the statistical description of the dataset. The inverse transformation and inverse scaling has been properly implemented and applied to the cluster centers in code.

Similar to the above section looking at the 3 sample points, we can also look at where the centers would theoretically rank in percentile terms...

```
import seaborn as sns
import matplotlib.pyplot as plt
# add the true centers as rows to our original data
newdata = data.append(true_centers)

# show the percentiles of the centers
ctr_pcts = 100. * newdata.rank(axis=0, pct=True).loc[['Segment 0', 'Segmen
t 1']].round(decimals=3)
print ctr_pcts

# visualize percentiles with heatmap
_ = sns.heatmap(ctr_pcts, annot=True, cmap='Greens', fmt='.1f', square=Tru
e, cbar=False)
plt.xticks(rotation=45, ha='center')
plt.yticks(rotation=0)
plt.title('Percentile ranks of\nsegment centers');
```



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.

For more reading on A/B testing, you can learn about how Netflix selected the best artwork for videos through A/B testing, and also how to determine the necessary sample size to detect a statistically significant effect.

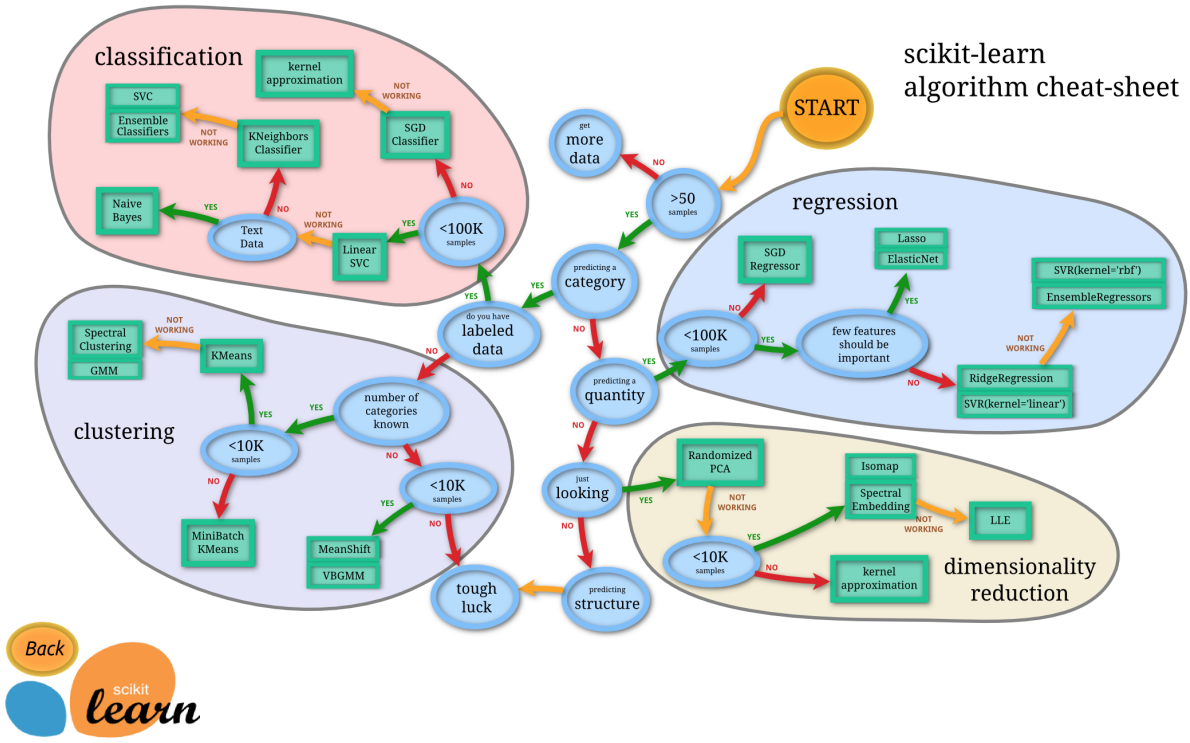
Student discusses with justification how the clustering data can be used in a supervised learner for new predictions.

You may already be doing this, but as you go through the projects it can also be useful to look in the forums for guidance.

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.

It's outside the scope of this project, but another approach you might experiment with is density based clustering (DBSCAN), which can be effective when the number of clusters is unknown.

DBSCAN isn't included in the below diagram of selecting an algorithm, but the flowchart is still a handy rough guide for choosing a predictive model technique.



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