UDACITY

PROJECT

# Creating Customer Segments

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

## Requires Changes

3 SPECIFICATIONS REQUIRE CHANGES

### 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 work here but please make sure to also compare each of the samples expenditures to the statistical description of the data (e.g. you could compare each sample against the mean expenditure).

Try looking at their normalized expenditures:

```
import seaborn as sns

sns.heatmap((samples-data.mean())/data.std(ddof=0), annot=True, cbar=False, square=True)
```

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

A feature with a high R^2 score i.e that is well modeled by all the other features is less useful when attempting to identify customer spending habits. In terms of information gain from its inclusion in a model, it provides less. Its value can be inferred easily from the rest of the features!

I used the code below to study all of the features over 100 randomly generated trials. Such an analysis can tell you a lot about the correlation between all of the features. Are there others with higher R^2 scores?

```
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeRegressor

def calculate_r_2_for_feature(data,feature):
    new_data = data.drop(feature, axis=1)

    X_train, \n    X_test,  \n    y_train, \n    y_test = train_test_split(
        new_data,data[feature],test_size=0.25
    )

    regressor = DecisionTreeRegressor()
    regressor.fit(X_train,y_train)

    score = regressor.score(X_test,y_test)
    return score
```

```
def r_2_mean(data,feature,runs=200):
    return np.array([calculate_r_2_for_feature(data,feature)
                     for _ in range(200) ]).mean().round(4)

print "{0:17} {1}".format("Fresh: ", r_2_mean(data,'Fresh'))
print "{0:17} {1}".format("Milk: ", r_2_mean(data,'Milk'))
print "{0:17} {1}".format("Grocery: ", r_2_mean(data,'Grocery'))
print "{0:17} {1}".format("Frozen: ", r_2_mean(data,'Frozen'))
print "{0:17} {1}".format("Detergents_Paper: ", r_2_mean(data,'Detergents_Paper'))
print "{0:17} {1}".format("Delicatessen: ", r_2_mean(data,'Delicatessen'))
```

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

I used this to visualize correlations.

```
corr = data.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask, 1)] = True
with sns.axes_style("white"):
    ax = sns.heatmap(corr, mask=mask, square=True, annot=True,
                     cmap='RdBu', fmt='+.3f')
    plt.xticks(rotation=45, ha='center')
```

## Data Preprocessing

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

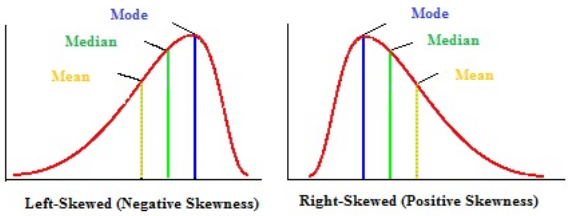
This helped me to see the distribution of data before and after scaling.

```
fig, axes = plt.subplots(2, 3)
axes = axes.flatten()
fig.set_size_inches(18, 6)
fig.suptitle('Distribution of Features')

for i, col in enumerate(product_data_by_customer_df.columns):
    feature = product_data_by_customer_df[col]
    sns.distplot(feature, label=col, ax=axes[i]).set(xlim=(-1000, 20000),)
    axes[i].axvline(feature.mean(),linewidth=1)
    axes[i].axvline(feature.median(),linewidth=1, color='r')
```

Then after preparing the log data:

```
fig, axes = plt.subplots(2, 3)
axes = axes.flatten()
fig.set_size_inches(18, 6)
fig.suptitle('Distribution of Features for Log Data')

for i, col in enumerate(log_product_data_df.columns):
    feature = log_product_data_df[col]
    sns.distplot(feature, label=col, ax=axes[i])
    axes[i].axvline(feature.mean(),linewidth=1)
    axes[i].axvline(feature.median(),linewidth=1, color='r')
```

From these you can think about skewness in data:




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

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

Some helpful links on outliers:

- http://www.theanalysisfactor.com/outliers-to-drop-or-not-to-drop/
- http://graphpad.com/guides/prism/6/statistics/index.htm?stat_checklist_identifying_outliers.htm

You can also find the double counted outliers programatically using a Counter:

```
from collections import Counter
```

## 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.**

A principal component is an engineered feature made from the original features. The first dimension has 4 features correlated together and to features correlated in the other direction. The sign of the features can not be interpreted. The signs are actually reversible, and if you run it multiple times on your computer you may have noticed this. Have a look at
http://stats.stackexchange.com/questions/30348/is-it-acceptable-to-reverse-a-sign-of-a-principal-component-scorecorrelated
What we are looking for here is the largest absolute value magnitude features. These are the features that are most heavily represented.

Further reading:
You can read more about how to interpret the dimensions here:

- https://onlinecourses.science.psu.edu/stat505/node/54
- http://setosa.io/ev/principal-component-analysis/
- https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues
- https://www.quora.com/What-is-an-intuitive-explanation-for-PCA

I used this to show the cumulative variance:

```
# create an x-axis variable for each pca component
x = np.arange(1,7)

# plot the cumulative variance
plt.plot(x, np.cumsum(pca.explained_variance_ratio_), '-o', color='black')

# plot the components' variance
plt.bar(x, pca.explained_variance_ratio_, align='center', alpha=0.5)
```

```
# plot styling
plt.ylim(0, 1.05)
plt.annotate('Cumulative
explained
variance',
             xy=(3.7, .88), arrowprops=dict(arrowstyle='->'), xytext=(4.5, .6))
for i,j in zip(x, np.cumsum(pca.explained_variance_ratio_)):
    plt.annotate(str(j.round(4)),xy=(i+.2,j-.02))
plt.xticks(range(1,7))
plt.xlabel('PCA components')
plt.ylabel('Explained Variance')
plt.show()
```

If you really want to understand PCA, I recommend MIT's 1806 course, especially eigenvalues and vectors:

https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/index.htm

**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.**

Some helpful links:

http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/mixture.html
http://www.nickgillian.com/wiki/pmwiki.php/GRT/GMMClassifier
http://playwidtech.blogspot.hk/2013/02/k-means-clustering-advantages-and.html
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
http://www.r-bloggers.com/k-means-clustering-is-not-a-free-lunch/
http://www.r-bloggers.com/pca-and-k-means-clustering-of-delta-aircraft/
https://shapeofdata.wordpress.com/2013/07/30/k-means/
http://mlg.eng.cam.ac.uk/tutorials/06/cb.pdf
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.**

Suggestion:
The choice of outliers is very important for the silhouette scores of your outliers. What if you ran your analysis again, this time without removing any outliers? How would the number of clusters and silhouette scores vary?

**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.**

Note that it is required that you compare the identified customer segments to the statistical descrition of the dataset.

Try plotting the normalized cluster expenditures:

```
import seaborn as sns

sns.heatmap((true_centers-data.mean())/data.std(ddof=1), annot=True, cbar=False, square=True)
```

You could also visualize each segment in terms of the underlying features:

```
centers = true_centers.copy()
centers[true_centers.shape[0]] = data.median()
```

```
plt.style.use('ggplot')
centers(kind='bar')
labels = true_centers.index.values.tolist()
labels.append("Data Median")
plt.xticks(range(centers[0]),labels)
```

**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.**

Note that you are required to provide the sketch of what an A/B test on this scenario would look like based upon you analysis.

How can you use the knowledge you have just gained about clusters to design a test against each of these clusters? Think in terms of the two clusters and a test group and a control group.

Here are a few links for further reading on A/B testing:
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.**

Here is an online discussion about using unsupervised learning to generate a target: https://datascience.stackexchange.com/questions/985/can-i-use-unsupervised-learning-followed-by-supervised-learning

**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.**

RESUBMIT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH