UDACITY

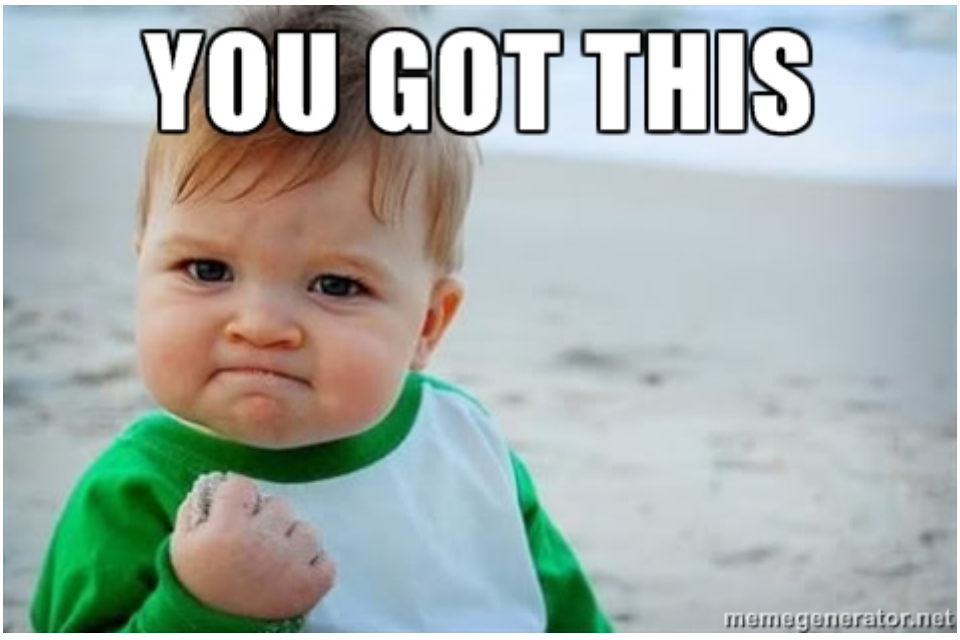
Back to Machine Learning Engineer Nanodegree

Creating Customer Segments

REVIEW

HISTORY

Meets Specifications



Congratulations on passing your exam! 👏👏

Data Exploration

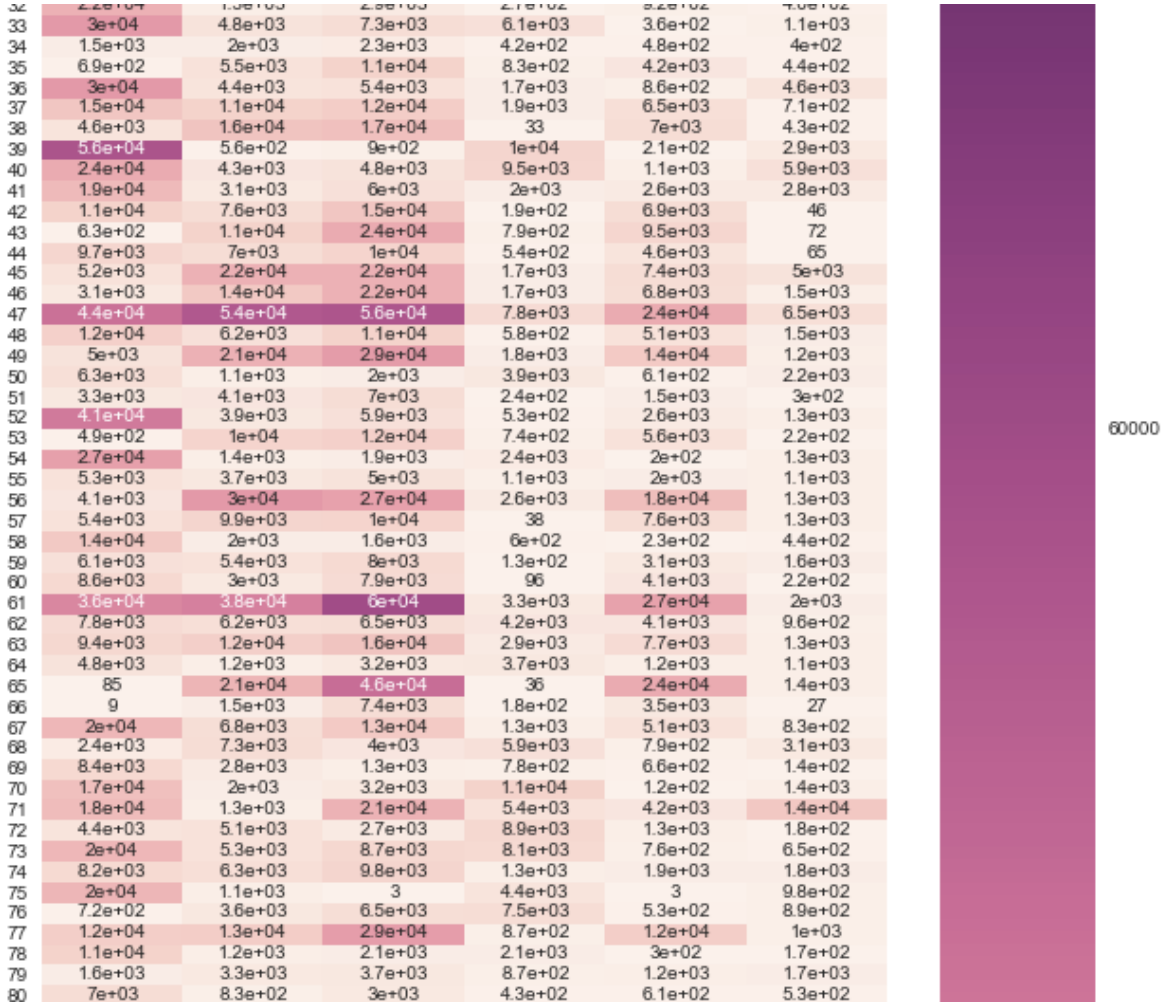
Three separate samples of the data are chosen and their establishment representations are proposed based on the statistical description of the dataset.

Bonus

You can also use this heatmap in order to check what are the most relevant indices in the dataset:

```
import matplotlib.pyplot as plt
import seaborn as sns # install by typing 'pip install seaborn' in the termin
al

%matplotlib inline

fig, ax = plt.subplots(figsize=(10,30))
sns.heatmap(data[:150], annot=True, ax=ax)
```

Output:



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

Comment

Correct! A negative R^2 score means that the model fits poorly the data. That means the given feature is relevant. This forum discusses the negative R score.

Bonus

The feature's scores can vary significantly when changing the random_state. Here's an average score over 1000 tests of random states for all categories:

In [7]:
```
categories = data.columns

for name in categories:
    new_data = data.drop(name, axis=1)
    scores = []

    for r in range(1000):
        X_train, X_test, y_train, y_test = \
            train_test_split(new_data, data[name], test_size=0.25, random_state=r)
        regressor = DecisionTreeRegressor(random_state=r).fit(X_train,y_train)
        scores.append(regressor.score(X_test, y_test))

    print("Average score over 1000 random states for {}: {}".format(name, np.mean(scores)))
```

```
Average score over 1000 random states for Fresh: -0.742560283417
Average score over 1000 random states for Milk: 0.120710240044
Average score over 1000 random states for Grocery: 0.674541932071
Average score over 1000 random states for Frozen: -1.28382318184
Average score over 1000 random states for Detergents_Paper: 0.679866636495
Average score over 1000 random states for Delicatessen: -3.29192600467
```

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

Comment

> the data and outliers are skewed to the left of the charts. What this means is the data needs to be normalized before processed as preferred by clustering.

That's correct! In fact, we have to apply "log-transform" in order to "normalize" the features. Besides skewed to the left, we can also say that the categories have a log-normal distribution

Data Preprocessing

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

Comment

Although very simple, a very important pre-processing was performed in this part to ensure that we are working with malleable data for the clustering section. The logarithmic transformation was applied due to the fact that the dataset has an approximately log-normal distribution. This transformation is not always the most appropriate and varies according to the distribution of the data. This sklearn module provides alternative preprocessing functions:

- http://scikit-learn.org/stable/modules/preprocessing.html

Suggestion

These two papers discuss the implications of the logarithmic transformations in the data:

- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4120293/
- https://www.r-statistics.com/2013/05/log-transformations-for-skewed-and-wide-distributions-from-practical-data-science-with-r/

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

Suggestion

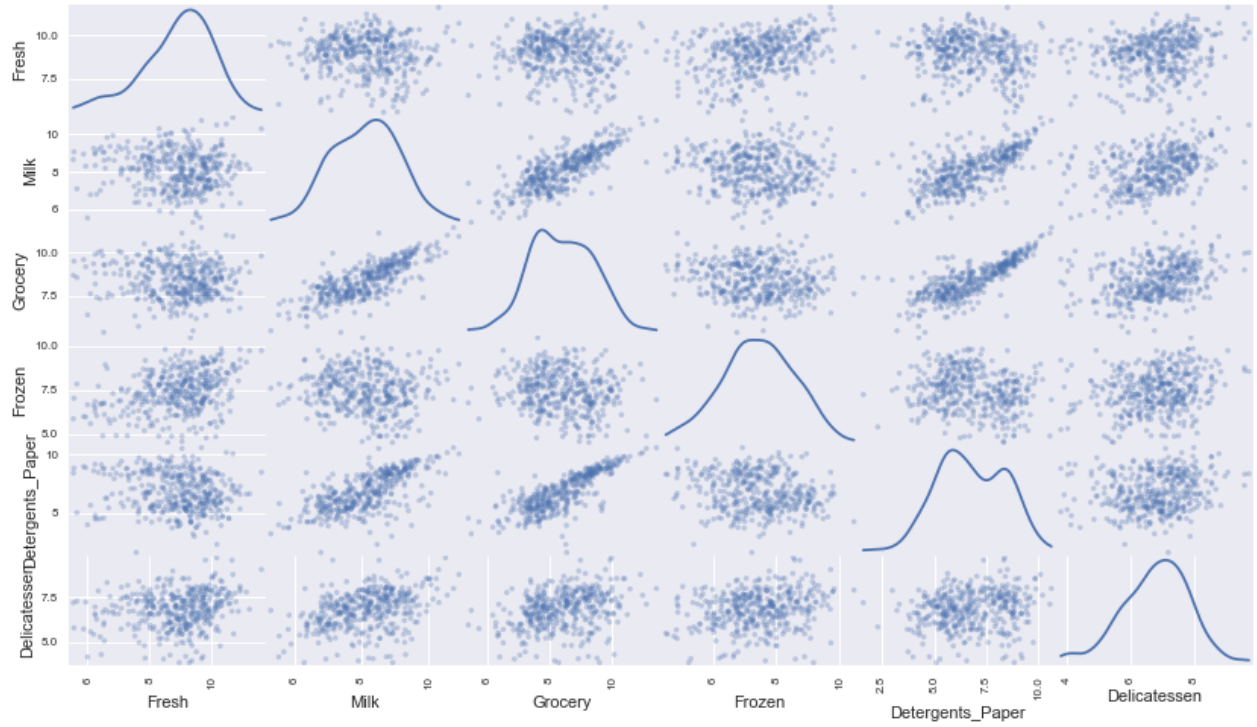
We can get the double counted outliers using this code:

```
outliers = list(set([x for x in outliers if outliers.count(x) > 1]))
```

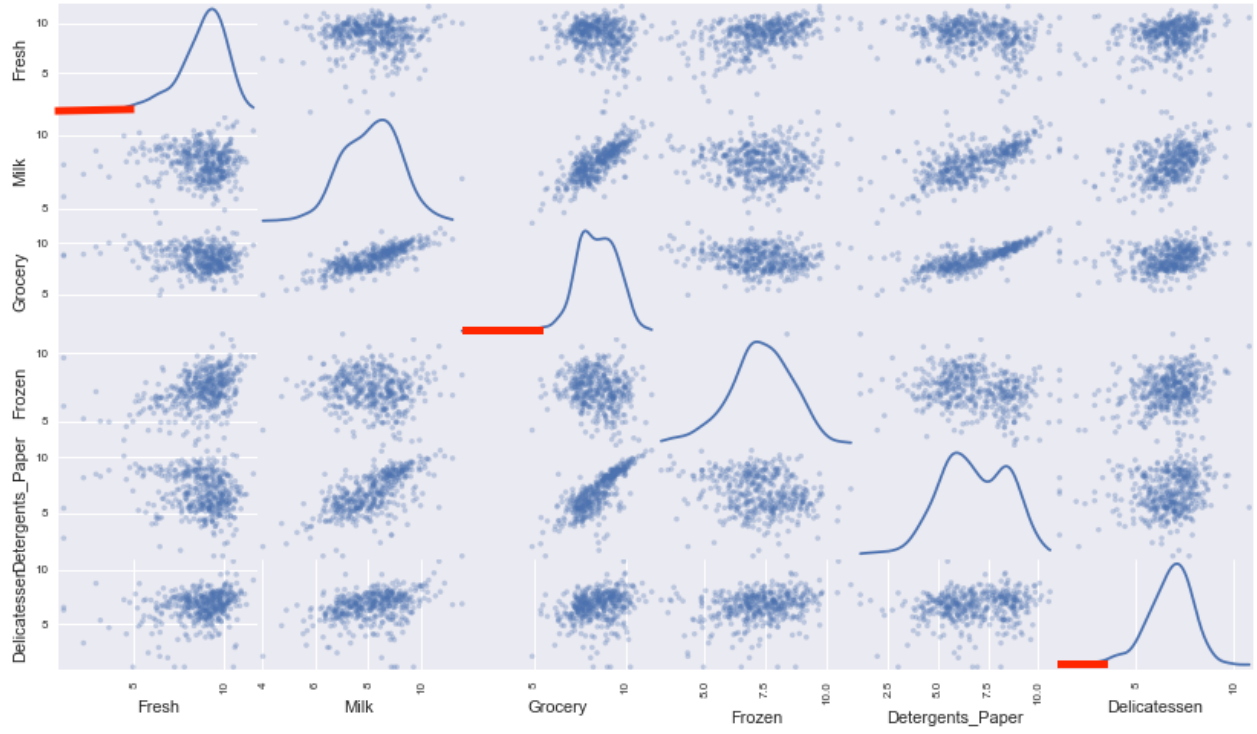
Comment

Here's how the distribution of each feature looks like if we remove all the outliers:

```
pd.scatter_matrix(good_data, alpha = 0.3, figsize = (14,8), diagonal = 'kde'
);
```



We can check that the tails considerably decreased, especially for grocery and fresh. Here's how it looked like before (tails in red):



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.

Comment

I consider this analysis the most difficult of this project. First of all, keep in mind that the signs of the transformations can be reversed. When interpreting those weights, what really matters is their amplitude, not their sign. In each component, the categories with high weight amplitude are going to define how the component's axis is split into positive and negative values. As a summary, the transformation that occurs in the data is the product of the weights that we see in the graph with the the prices minus their averages. The formulation is as follows:

```
Transformation PCA = Weights x (Features_Values - Features_Mean)
```

Positive weights for a given attribute make any value of this attribute that is *above average* also assume positive values in the transformation. Below average values are translated into negative values in the transformation. Conversely, negative weights cause values above the mean to assume negative values in the transformation and values below the mean assume positive values in the transformation.

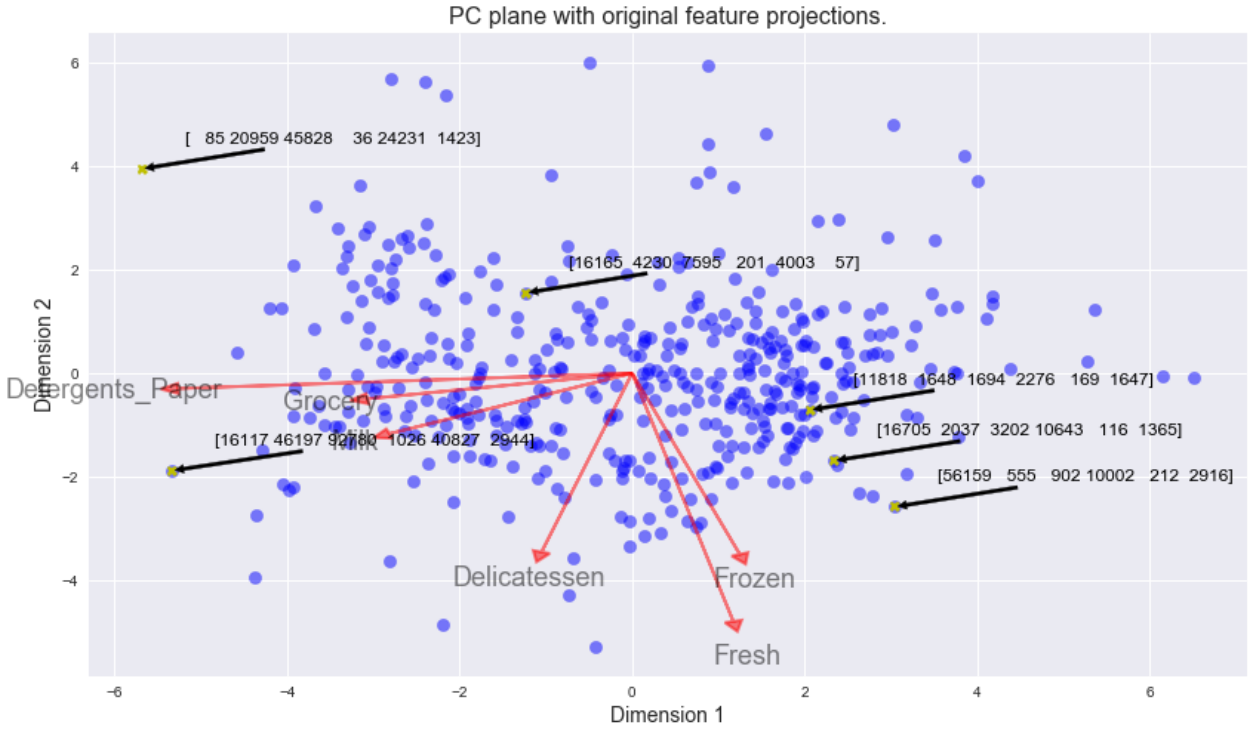
Bonus

We can visualize some of the transformed samples and compare with the original ones (pointed with arrows) using this biplot:

```
# Visualize Biplot with original samples
fig, ax = plt.subplots(figsize = (14,8))
# scatterplot of the reduced data
ax.scatter(x=reduced_data.loc[:, 'Dimension 1'], y=reduced_data.loc[:, 'Dimen
sion 2'],
    facecolors='b', edgecolors='b', s=70, alpha=0.5)

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], alpha=0.5,
              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'
, alpha=0.5,
             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);
```

```
for i,sample in enumerate(samples_.values):
    ax.plot(pca_samples_[i,0],pca_samples_[i,1],'yX')
    ax.annotate(sample,xy=(pca_samples_[i,0],pca_samples_[i,1]), xytext=(pca_
samples_[i,0]+0.5,pca_samples_[i,1]+0.5),
            arrowprops=dict(arrowstyle='simple', facecolor='black'), fontsize
=12, color='black')
```

Here's the output (where the values of each original sample are in the following order `[Fresh, Milk, Grocery, Frozen, Detergents_Paper, Delicatessen]`):



For example, we can clearly see that clients with very high spends in 'Detergents_Paper' are more predominant in the left (negative) side while clients spending very low in 'Detergents_Paper' are more predominant in the right (positive) side. This confirms the very high negative amplitude for Detergents_Paper in the first component.

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

Suggestion

There are other dimensionality reduction strategies besides PCA. You might want to check this sklearn's page:

- http://scikit-learn.org/stable/modules/unsupervised_reduction.html

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.

Bonus

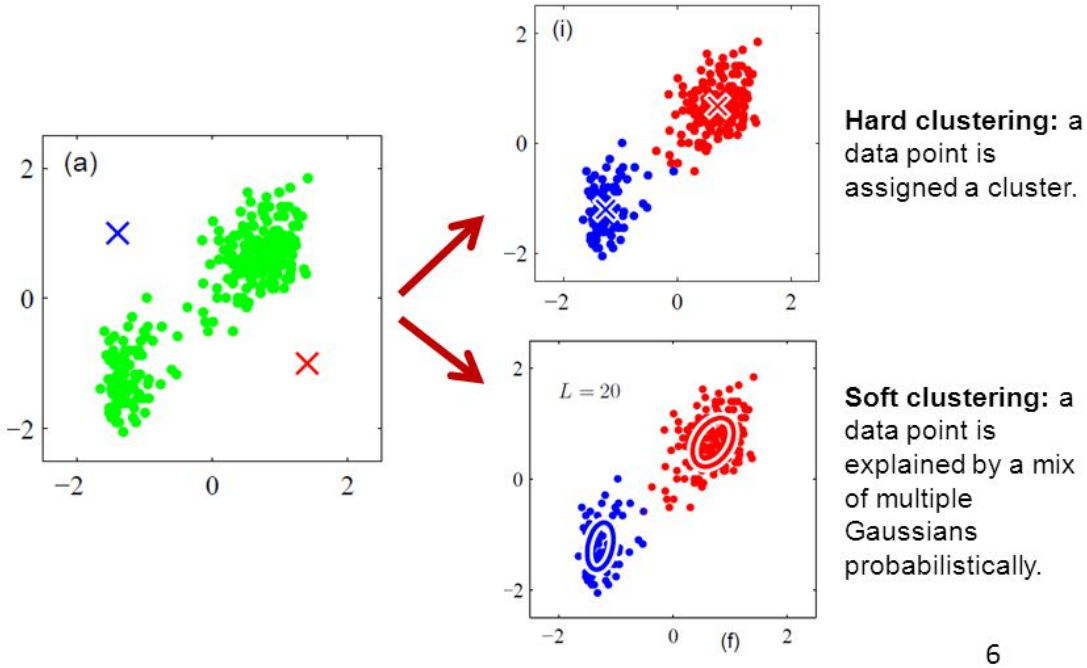
The following slide summarizes the difference between both GMM and K-Means:

Imperial College London

EE462 MLCV

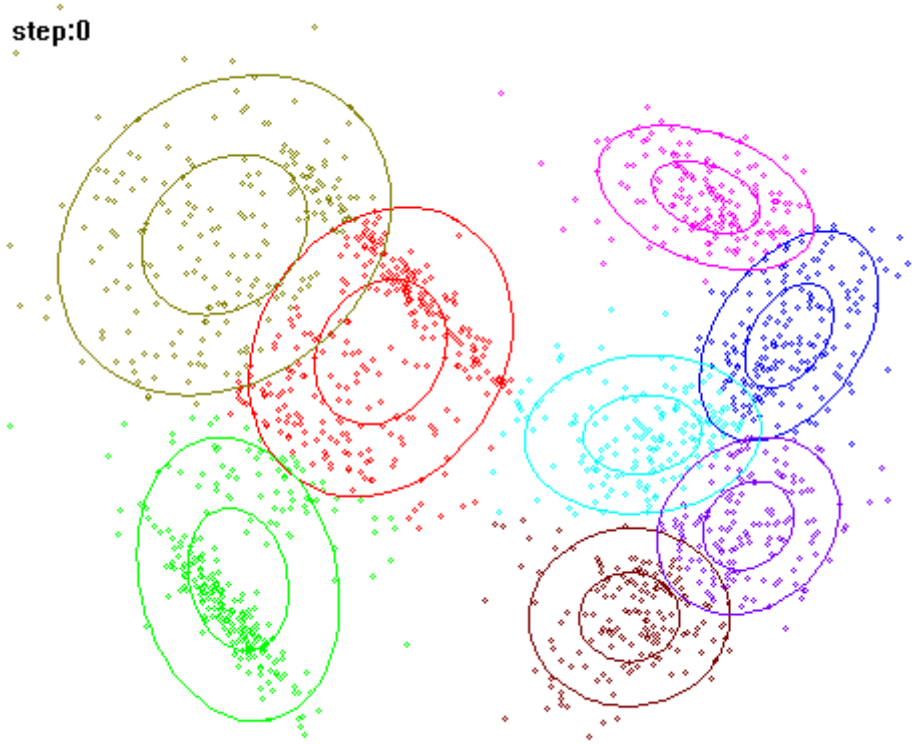
K-means vs GMM

Two standard methods are k-means and Gaussian Mixture Model (GMM).
K-means assigns data points to the nearest clusters, while GMM represents data by multiple Gaussian densities.

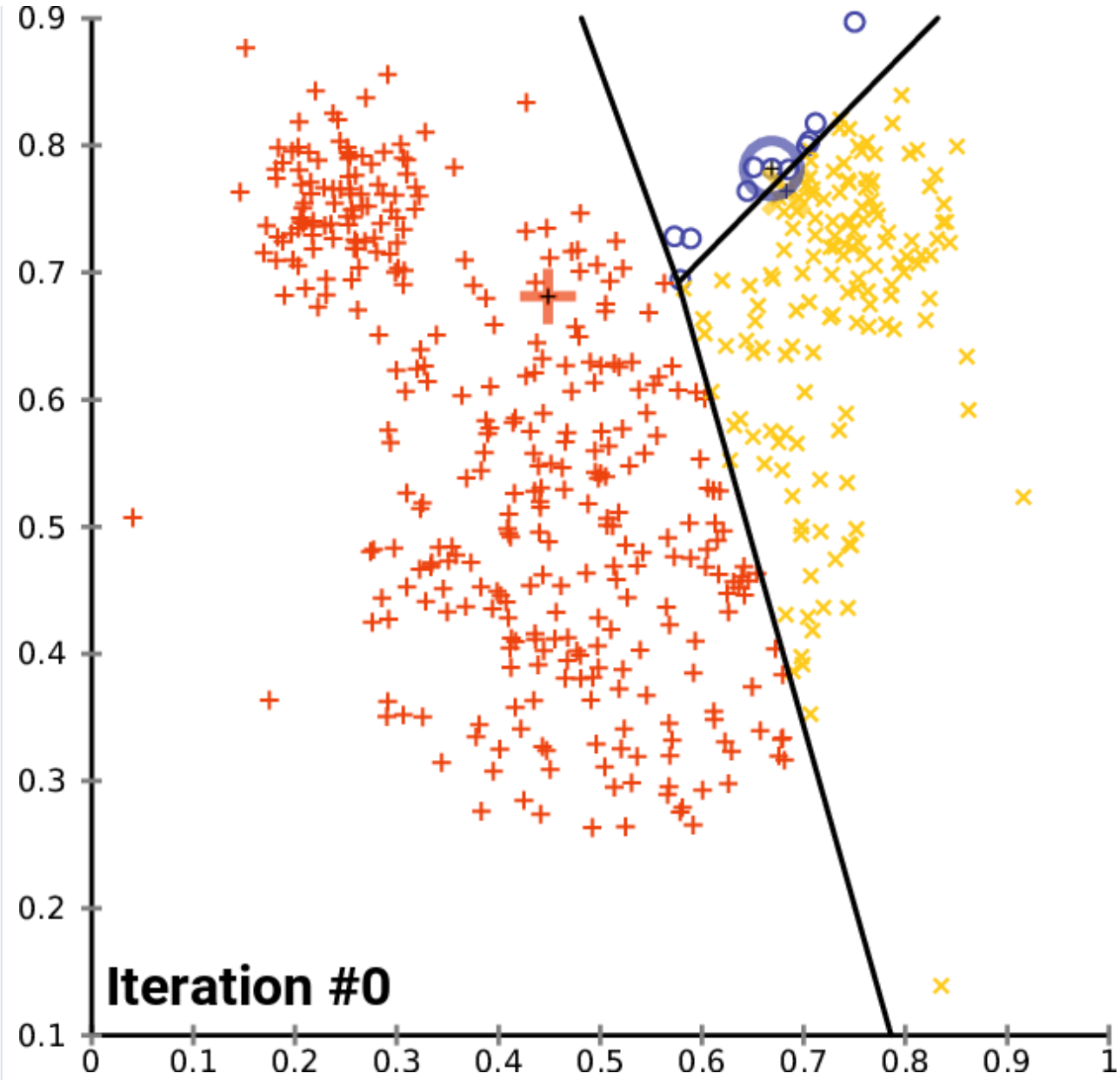


6

Also, the following gif illustrates how GMM works:



And this one, how K-Means works:

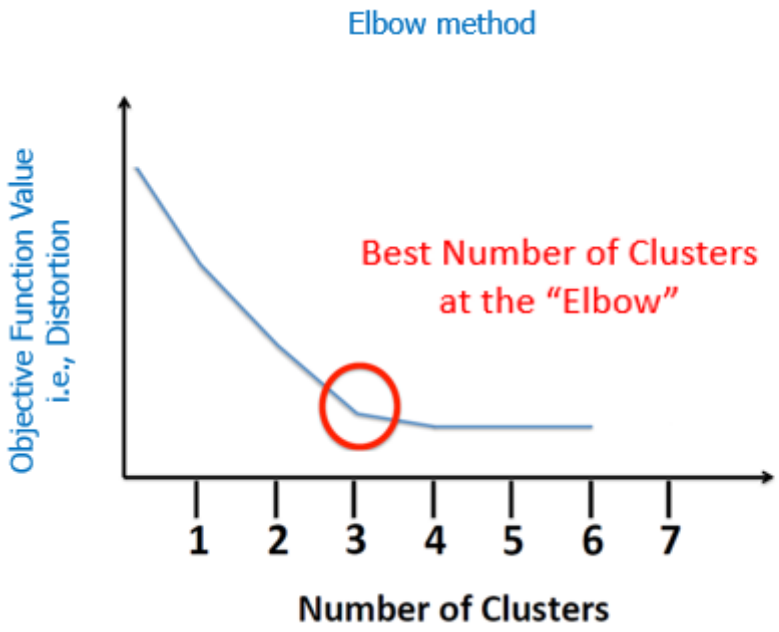


We can clearly see that GMM is a generalized case of k-means: GMM performs a soft assignment while k-means perform a hard assignment.

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.

Comment

That's correct! The Elbow method is also very popular for choosing the best number of clusters. here's an illustration on how it works:



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.

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.

Comment

Correct! We have two possibilities:

- We could either train a supervised learner using the predicted labels as target variable;
- Or use the unsupervised trained model to make the prediction itself.

The later one would work in a similar way to a k-Nearest Neighbours model :)

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