Market Customer Segmentation

Face Feature Extraction

Name : Meghna Maan

Course : AI & ML (Batch 4)

Prerequisites

1. Software:

   Python 3

2. Tools:

   - Matplotlib
   - Numpy
   - Pandas
   - Visuals

Implementation

1. Loading Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import cm as cm
from IPython.display import display
import visuals as vs
%matplotlib inline

try:
    data = pd.read_csv("customers.csv")
    data.drop(['Region', 'Channel'], axis = 1, inplace=True)
    print("Wholesale customers dataset has {} samples with {} features each.".format(*data.shape))
except:
    print "Dataset could not be loaded. Is the dataset missing?"
```

2. Selecting Samples

```
indices = [7,70,296]

samples = pd.DataFrame(data.loc[indices], columns = data.keys()).reset_index(drop=True)
print("Chosen samples of wholesale customers dataset:")
display(samples)
```

3. Data Processing

```
log_data = np.log(data)

log_samples = np.log(samples)

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

# Numeric visualization
display(data.corr())
```

4. Outlier Detection

```
for feature in log_data.keys():


    Q1, Q3 = np.percentile(log_data[feature], 25), np.percentile(log_data[feature], 75)


    step = 1.5 * (Q3 - Q1)


    print("Data points considered outliers for the feature '{}':".format(feature))
    display(log_data[~((log_data[feature] >= Q1 - step) & (log_data[feature] <= Q3 + step))]

outliers  = [65,66,75,128,154]

good_data = log_data.drop(log_data.index[outliers]).reset_index(drop=True)
```

5. Feature Transformation

```
from sklearn.decomposition import PCA

# Apply PCA by fitting the good data with the same number of dimensions as features
pca = PCA()
pca.fit(good_data)

pca_samples = pca.transform(log_samples)

# Generate PCA results plot
pca_results = vs.pca_results(good_data, pca)
```

6. Dimensionality Reduction

```
# Apply PCA by fitting the good data with only two dimensions
pca = PCA(n_components=2)
pca.fit(good_data)

# Transform the good data using the PCA fit above
reduced_data = pca.transform(good_data)

# Transform the sample log-data using the PCA fit above
pca_samples = pca.transform(log_samples)

# Create a DataFrame for the reduced data
reduced_data = pd.DataFrame(reduced_data, columns=['Dimension 1', 'Dimension 2'])
```

7. Creating Cluster

```
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score

# Apply clustering algorithm to the reduced data
clusterer = GaussianMixture(n_components=2, n_init=10, random_state=42)
clusterer.fit(reduced_data)

# Predict the cluster for each data point
preds = clusterer.predict(reduced_data)

# Find the cluster centers
centers = clusterer.means_

# Predict the cluster for each transformed sample data point
sample_preds = clusterer.predict(pca_samples)

# Calculate the mean silhouette coefficient for the number of clusters chosen
score = silhouette_score(reduced_data,preds)

print 'Silhouette score using GMM: {:.3f}'.format(score)
```

8. Cluster Visualization

```
probs = pd.DataFrame(clusterer.predict_proba(reduced_data))
border_indexes = np.array(probs[(probs[0] >= 0.4) & (probs[0] <= 0.6)].index)
border_data = reduced_data.values[border_indexes]
# Display the results of the clustering from implementation
vs.cluster_results(reduced_data, preds, centers, pca_samples, border_data)
```

9. Visualizing Underlying Distribution and final result

```
# Display the clustering results based on 'Channel' data
vs.channel_results(reduced_data, outliers, pca_samples)
```