Lab package 2 supplement

Friday, November 5, 2021 9:17 AM

Below are the codes after the Silhouettee analysis (end of class on November 4).
It will be most helpful if you enter these codes in your colab notebook before our class on Nov 11, so that we can spend the class hours on explaining the concepts and how the codes work.

```
[29] clusters = KMeans(n_clusters=2, random_state=1).fit(Xs_10DF)
     Xs_10DF['kmeans2'] = clusters.labels_

[30] Xs_10DF.columns

[31] import plotly.express as px
     fig = px.parallel_coordinates(Xs_10DF, color='kmeans2', labels=Xs_10DF.columns, color_continuous_scale=[(0, 'green'), (0.5, 'green'), (0.5, 'orange'), (1,'orange')])
     fig.show()

[32] gdfTract2020['kmeans2'] = Xs_10DF['kmeans2']
     fig, ax = plt.subplots(1, figsize=(14, 8))
     gdfTract2020.plot(ax=ax,
                       column='kmeans2',
                       categorical=True,
                       cmap='Spectral',
                       legend=True,
                       legend_kwds={'loc': 'lower right'})
     ax.set_title('KMeans 2 demographic classes on tracts')
     plt.tight_layout()
     plt.savefig('/content/drive/MyDrive/package2/KMeans2.png', dpi=300)

[33] agg_cluster = AgglomerativeClustering().fit(Xs_10DF)
     agg_cluster.labels_
```

The codes below are copied from https://scikit-learn.org/stable/auto_examples/cluster/plot_agglomerative_dendrogram.html#sphx-glr-auto-examples-cluster-plot-agglomerative-dendrogram-py

```
[34] from scipy.cluster.hierarchy import dendrogram

     def plot_dendrogram(model, **kwargs):
         # Create linkage matrix and then plot the dendrogram

         # create the counts of samples under each node
         counts = np.zeros(model.children_.shape[0])
         n_samples = len(model.labels_)
         for i, merge in enumerate(model.children_):
             current_count = 0
             for child_idx in merge:
                 if child_idx < n_samples:
                     current_count += 1  # leaf node
                 else:
                     current_count += counts[child_idx - n_samples]
             counts[i] = current_count

         linkage_matrix = np.column_stack(
             [model.children_, model.distances_, counts]
         ).astype(float)

         # Plot the corresponding dendrogram
         dendrogram(linkage_matrix, **kwargs)
```

```
[35] agg_cluster = AgglomerativeClustering(distance_threshold=0, n_clusters= None).fit(Xs_10DF)
     plt.title('Hierarchical Cluster Dendrogram')
     plot_dendrogram(agg_cluster, truncate_mode='level', p=10)
     plt.xlabel('Number of points in node (or index of point if no parenthesis')
     plt.show()

[36] agg_7 = AgglomerativeClustering(n_clusters=7).fit(Xs_10DF)
     gdfTract2020['agg_7'] = agg_7.labels_
     gdfTract2020.plot(column='agg_7', categorical=True, cmap='Spectral', legend=True, legend_kwds={'loc': 'lower right'})

[37] silhouette_score(Xs_10DF, agg_7.labels_)

[38] aff = AffinityPropagation(preference=-500, random_state=10).fit(Xs_10DF)
     n_clusters = len(aff.cluster_centers_indices_)
     silhouette = silhouette_score(Xs_10DF, aff.labels_)

     print('estimated number of clusters: %d' % n_clusters)
     print('Silhouettee coefficient: %0.3f' % silhouette)

[39] aff.cluster_centers_indices_

[40] idxlist = aff.cluster_centers_indices_.tolist()
     idxlist
```

```
[40] idxlist = aff.cluster_centers_indices_.tolist()
     idxlist

[41] prototype = gdfTract2020.iloc[idxlist]
     prototype

[42] fig, ax = plt.subplots(figsize=(10,8))

     gdfTract2020['aff_2'] = aff.labels_
     gdfTract2020.plot(column='aff_2', ax=ax)
     prototype.plot(color='red', ax=ax)

[43] gdfTract2020.plot(column='kmeans2')

[44] gdfTract2020.columns

[45] agg_2 = AgglomerativeClustering(n_clusters=2).fit(Xs_10DF)
     gdfTract2020['agg_2'] = agg_2.labels_
     gdfTract2020.plot(column='agg_2')

[46] !pip install pysal
     !pip install shapely
```

```
[47] from pysal.lib import weights

[48] wrook = weights.contiguity.Rook.from_dataframe(gdfTract2020)
     wqueen = weights.contiguity.Queen.from_dataframe(gdfTract2020)

[49] wrook.neighbors

[50] pd.DataFrame(*wrook.full()).astype(int)

[51] wrook.nonzero

[52] agg_spatial = AgglomerativeClustering(n_clusters=5, connectivity=wqueen.sparse).fit(Xs_10DF)
     gdfTract2020['agg_spatial'] = agg_spatial.labels_
     gdfTract2020.plot(column='agg_spatial')

     silhouette = silhouette_score(Xs_10DF, agg_spatial.labels_)
     print('Silhouettee coefficient: %0.3f' % silhouette)

[53] Xs_10DF.drop('kmeans2', axis=1, inplace=True)

[54] !pip install sklearn_som

[55] from sklearn_som.som import SOM
     X = Xs_10DF.to_numpy()
     som_cls = SOM(m=1, n=2, dim=10).fit_predict(X)
```

```
[56] som_cls

[57] silhouette = silhouette_score(Xs_10DF, som_cls)
     print('Silhouettee coefficient: %0.3f' % silhouette)

[58] gdfTract2020['som2'] = som_cls
     gdfTract2020.plot(column='som2')
```