L11_PCA_Kmeans_filled

January 25, 2019

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# you will likely need to add some imports here
%matplotlib inline
```

0.0.1 K-means clustering

In [3]:
```
harvard = pd.read_csv('https://raw.githubusercontent.com/UWDIRECT/UWDIRECT.github.io/ma
```

Perform k-means clustering on the harvard data. Vary the `k` from 5 to 50 in increments of 5. Run the `k-means` using a loop.

Be sure to use standard scalar normalization! `StandardScaler().fit_transform(dataArray)`

In [ ]:
```
```

Use silhouette analysis to pick a good k.

In [ ]:
```
```

0.0.2 PCA and plotting

The functions below may be helpful but they aren't tested in this notebook.

In [1]:
```
def make_pca(array, components):
    pca = PCA(n_components=components, svd_solver='full')
    pca.fit(array)
    return pca

def plot_pca(pca, array, outplt, x_axis, y_axis, colorList=None):
    markers = pca.transform(array)
    plt.scatter(markers[:, x_axis], markers[:, y_axis], color='c')
    if colorList is not None:
        x_markers = [markers[i, x_axis] for i in colorList]
        y_markers = [markers[i, y_axis] for i in colorList]
        plt.scatter(x_markers, y_markers, color='m')

    plt.xlabel("Component 1 ({}%)".format(pca.explained_variance_ratio_[x_axis]*
```

```
100))
    plt.ylabel("Component 2 ({}%)".format(pca.explained_variance_ratio_[y_axis]*
100))
    plt.tight_layout()
    plt.savefig(outplt)

def plot_clusters(pca, array, outplt, x_axis, y_axis, colorList=None):
    xkcd = [x.rstrip("\n") for x in open("xkcd_colors.txt")]

    markers = pca.transform(array)
    if colorList is not None:
        colors = [xkcd[i] for i in colorList]
    else:
        colors = ['c' for i in range(len(markers))]
    plt.scatter(markers[:, x_axis], markers[:, y_axis], color=colors)


    plt.xlabel("Component {0} ({1:.2f}%)".format((x_axis+1), pca.explained_varia
nce_ratio_[x_axis]*100))
    plt.ylabel("Component {0} ({1:.2f}%)".format((y_axis+1), pca.explained_varia
nce_ratio_[y_axis]*100))
    plt.tight_layout()
    plt.savefig(outplt)

data = pd_to_np(df)
pca = make_pca(data, args.n_components)
print(pca.explained_variance_ratio_)


  File "<ipython-input-1-3cc7d82d4f37>", line 34
nce_ratio_[x_axis]*100))
          ^
SyntaxError: invalid syntax
```

In [ ]: