CorreaSydney

September 18, 2018

##
CSCI E-82
##
HW 1 Dimensionality Reduction
###
Due: Sept 17, 2018 11:59pm EST

Note that this is an individual homework to be completed without collaborations except through Piazza.

We encourage you to make progress this weekend since the second homework will likely come out in a week before this one is due.

0.0.1 Your name: Sydney Correa

```
In [163]: import numpy as np
          import sklearn as sk
```

0.0.2 Problem 1 (5 points)

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix}$$

Compute XYT. The answer can be computed by hand and written in Markdown like the above matrices, or computed in python. Either way is acceptable.

```
In [164]: X = np.array([[1,2,3],[4,5,6],[7,8,9]])
          Y = np.array([[1,2,1],[2,1,2]])
          Z = np.dot(X,Y.T)
          print (Z)
```

```
[[ 8 10]
 [20 25]
 [32 40]]
```

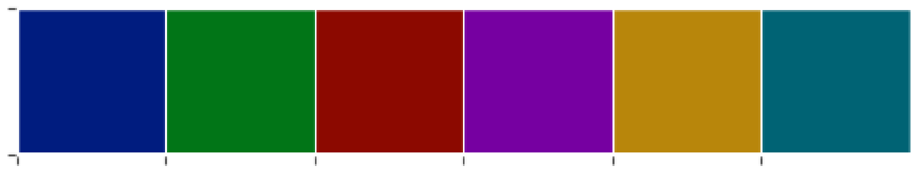
0.0.3 Problem 2

This problem goes through a combination of python data manipulations as well as the full math projection using PCA. We have divided the problem into multiple parts.

0.0.4 Problem 2a (5 points)

Download and load in the data set from the UCI archive https://archive.ics.uci.edu/ml/machine-learning-databases/ecoli/. Print the dimensions and the first few rows to demonstrate a successful load.

```
In [165]: import numpy as np
          import pandas as pd

          import matplotlib.pyplot as plt
          # Remember to use inline to get your plots in the notebook
          %matplotlib inline

          # special matplotlib command for global plot configuration
          from matplotlib import rcParams
          import matplotlib.cm as cm
          import matplotlib as mpl
          import matplotlib.style

          from matplotlib.colors import ListedColormap

          dark2_colors = ['#1b9e77','#d95f02','#7570b3','#e7298a','#66a61e','#e6ab02','#a6761d
          dark2_cmap = ListedColormap(dark2_colors)

          def set_mpl_params():
              rcParams['figure.figsize'] = (10, 6)
              rcParams['figure.dpi'] = 100
              rcParams['axes.prop_cycle'].by_key()['color'][1]
              rcParams['lines.linewidth'] = 2
              rcParams['axes.facecolor'] = 'white'
              rcParams['font.size'] = 12
              rcParams['patch.edgecolor'] = 'white'
              rcParams['patch.facecolor'] = dark2_colors[0]
              rcParams['font.family'] = 'StixGeneral'

          set_mpl_params()
          mpl.style.use('seaborn-dark-palette')

          import seaborn as sns #If you don't have seaborn, use - conda install seaborn
          current_palette = sns.color_palette()
          sns.palplot(current_palette)
```



```
In [183]: #import dataset
          url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/ecoli/ecoli.data'
          #name columns
          eColiCols = ['N_ecoli','mcg','gvh','lip','chg','aac','alm1','alm2','type']
          #read table into dataset
          eColiData = pd.read_table(url, sep='\s+', header= None, names = eColiCols)
          #print few rows
          #print (eColiData.head())
          eColiData.describe()

          #print dimensions of array
          #eColiData.shape

          #split data into data and values for Ecoli Dataset
          eColiData_num = eColiData.iloc[:,1:8]
          eColiData_class = eColiData.iloc[:,8]

          #print data
          print ('EColi Data: \n%s', eColiData_num.head())
          #print class
          print ('EColi Class Data: \n%s', eColiData_class.head())

          #print dataset for two dimensions alm2, alm1
          ax = plt.gca()
          ax.axis([0, 1, 0, 1])
          #ax.scatter(eColiData.iloc[:,6], eColiData.iloc[:,5], s=10)
          ax.scatter(eColiData.alm2, eColiData.alm1, s=10)
          ax.set_title('Raw Data')
          ax.grid(True)
          #cible_border()
          plt.show()
```

```
        ---------------------------------------------------------------------------

        TimeoutError                              Traceback (most recent call last)

        ~\Anaconda3\lib\urllib\request.py in do_open(self, http_class, req, **http_conn_args)
       1317                 h.request(req.get_method(), req.selector, req.data, headers,
    -> 1318                             encode_chunked=req.has_header('Transfer-encoding'))
       1319             except OSError as err: # timeout error


        ~\Anaconda3\lib\http\client.py in request(self, method, url, body, headers, encode_chun
       1238         """Send a complete request to the server."""
    -> 1239         self._send_request(method, url, body, headers, encode_chunked)
       1240


        ~\Anaconda3\lib\http\client.py in _send_request(self, method, url, body, headers, encod
       1284             body = _encode(body, 'body')
    -> 1285         self.endheaders(body, encode_chunked=encode_chunked)
       1286


        ~\Anaconda3\lib\http\client.py in endheaders(self, message_body, encode_chunked)
       1233             raise CannotSendHeader()
    -> 1234         self._send_output(message_body, encode_chunked=encode_chunked)
       1235


        ~\Anaconda3\lib\http\client.py in _send_output(self, message_body, encode_chunked)
       1025         del self._buffer[:]
    -> 1026         self.send(msg)
       1027


        ~\Anaconda3\lib\http\client.py in send(self, data)
        963             if self.auto_open:
    --> 964                 self.connect()
        965             else:


        ~\Anaconda3\lib\http\client.py in connect(self)
       1399             self.sock = self._context.wrap_socket(self.sock,
    -> 1400                                                   server_hostname=server_hostname)
       1401             if not self._context.check_hostname and self._check_hostname:


        ~\Anaconda3\lib\ssl.py in wrap_socket(self, sock, server_side, do_handshake_on_connect
        406                          server_hostname=server_hostname,
    --> 407                          _context=self, _session=session)
        408


        ~\Anaconda3\lib\ssl.py in __init__(self, sock, keyfile, certfile, server_side, cert_rec
        813                         raise ValueError("do_handshake_on_connect should not be spe
    --> 814                     self.do_handshake()
        815


        ~\Anaconda3\lib\ssl.py in do_handshake(self, block)
       1067                 self.settimeout(None)
    -> 1068             self._sslobj.do_handshake()
       1069         finally:


        ~\Anaconda3\lib\ssl.py in do_handshake(self)
        688         """Start the SSL/TLS handshake."""
    --> 689         self._sslobj.do_handshake()
        690         if self.context.check_hostname:


        TimeoutError: [WinError 10060] A connection attempt failed because the connected party


    During handling of the above exception, another exception occurred:


        URLError                                  Traceback (most recent call last)

        <ipython-input-183-ca107e279a21> in <module>()
          4 eColiCols = ['N_ecoli','mcg','gvh','lip','chg','aac','alm1','alm2','type']
          5 #read table into dataset
    ----> 6 eColiData = pd.read_table(url, sep='\s+', header= None, names = eColiCols)
          7 #print few rows
          8 #print (eColiData.head())


        ~\Anaconda3\lib\site-packages\pandas\io\parsers.py in parser_f(filepath_or_buffer, sep
        676                     skip_blank_lines=skip_blank_lines)
        677
    --> 678         return _read(filepath_or_buffer, kwds)
        679
        680     parser_f.__name__ = name


        ~\Anaconda3\lib\site-packages\pandas\io\parsers.py in _read(filepath_or_buffer, kwds)
        422     compression = _infer_compression(filepath_or_buffer, compression)
        423     filepath_or_buffer, _, compression, should_close = get_filepath_or_buffer(
    --> 424         filepath_or_buffer, encoding, compression)
        425     kwds['compression'] = compression
        426
        ~\Anaconda3\lib\site-packages\pandas\io\common.py in get_filepath_or_buffer(filepath_o
        193
        194     if _is_url(filepath_or_buffer):
    --> 195         req = _urlopen(filepath_or_buffer)
        196         content_encoding = req.headers.get('Content-Encoding', None)
        197         if content_encoding == 'gzip':


        ~\Anaconda3\lib\urllib\request.py in urlopen(url, data, timeout, cafile, capath, cadefa
        221     else:
        222         opener = _opener
    --> 223     return opener.open(url, data, timeout)
        224
        225 def install_opener(opener):


        ~\Anaconda3\lib\urllib\request.py in open(self, fullurl, data, timeout)
        524             req = meth(req)
        525
    --> 526         response = self._open(req, data)
        527
        528         # post-process response


        ~\Anaconda3\lib\urllib\request.py in _open(self, req, data)
        542         protocol = req.type
        543         result = self._call_chain(self.handle_open, protocol, protocol +
    --> 544                                   '_open', req)
        545         if result:
        546             return result


        ~\Anaconda3\lib\urllib\request.py in _call_chain(self, chain, kind, meth_name, *args)
        502         for handler in handlers:
        503             func = getattr(handler, meth_name)
    --> 504             result = func(*args)
        505             if result is not None:
        506                 return result


        ~\Anaconda3\lib\urllib\request.py in https_open(self, req)
       1359         def https_open(self, req):
       1360             return self.do_open(http.client.HTTPSConnection, req,
    -> 1361                 context=self._context, check_hostname=self._check_hostname)
       1362
       1363         https_request = AbstractHTTPHandler.do_request_
```

```
    ~\Anaconda3\lib\urllib\request.py in do_open(self, http_class, req, **http_conn_args)
   1318                             encode_chunked=req.has_header('Transfer-encoding'))
   1319                 except OSError as err: # timeout error
-> 1320                     raise URLError(err)
   1321                 r = h.getresponse()
   1322             except:


    URLError: <urlopen error [WinError 10060] A connection attempt failed because the conne
```

0.0.5 Problem 2b (10 points)

Compute and print the covariance matrix for all columns excluding the first and last. Rather than use the built-in function, compute this using python code for practice. The following equation will suffice for this.
Cov(X, Y) = (Xi - X) (Yi - Y) / N

```
In [168]: from sklearn.preprocessing import StandardScaler

          #standardize the data
          X_std = StandardScaler().fit_transform(eColiData_num)

          eColiData_num = eColiData.iloc[:,1:8].values

          #find means for dimensions
          eColiData_mean = np.mean(eColiData_num, axis=0)

          cov_mat = (X_std - eColiData_mean).T.dot((X_std - eColiData_mean)) / (X_std.shape[0]
          print('Covariance matrix \n%s' %cov_mat)

          #validate cov matrix
          print('Covariance matrix validate\n%s' %np.cov(X_std.T))
```

```
Covariance matrix
[[1.25379105 0.70693901 0.39578701 0.32199783 0.47214889 0.64902932
  0.41822671]
 [0.70693901 1.25373134 0.29241285 0.27001367 0.3207939  0.42484554
  0.13005389]
 [0.39578701 0.29241285 1.24921456 0.56209934 0.31889194 0.34463989
  0.19060217]
 [0.32199783 0.27001367 0.56209934 1.2552261  0.20664813 0.24390583
  0.19901358]
 [0.47214889 0.3207939  0.31889194 0.20664813 1.25376119 0.53116752
  0.50405568]
 [0.64902932 0.42484554 0.34463989 0.24390583 0.53116752 1.25391048
  1.06244095]
 [0.41822671 0.13005389 0.19060217 0.19901358 0.50405568 1.06244095
```

```
  1.25346276]]
Covariance matrix validate
[[ 1.00298507  0.45616289  0.14727982  0.07047535  0.22135784  0.39816364
   0.16758494]
 [ 0.45616289  1.00298507  0.04393523  0.01852114  0.07003271  0.17400972
  -0.12055805]
 [ 0.14727982  0.04393523  1.00298507  0.31288221  0.07039954  0.09607353
  -0.05774233]
 [ 0.07047535  0.01852114  0.31288221  1.00298507 -0.04485938 -0.00767652
  -0.05234423]
 [ 0.22135784  0.07003271  0.07039954 -0.04485938  1.00298507  0.28031677
   0.25342882]
 [ 0.39816364  0.17400972  0.09607353 -0.00767652  0.28031677  1.00298507
   0.81173951]
 [ 0.16758494 -0.12055805 -0.05774233 -0.05234423  0.25342882  0.81173951
   1.00298507]]
```

0.0.6 Problem 2c (10 points)

Compute the decomposition of the covariance matrix using singular value decomposition. Using a python function is definitely the way to go here.

```
In [169]: #get cov matrix
          eColiData_cov = np.cov(X_std.T)
          #print Cov dataset
          #print ('Cov dataset')
          #print(eColiData_cov)

          #check value of determinant >=0
          print ('value of Det(Cov)')
          print(np.linalg.det(eColiData_cov))

          #get eigenvalues, eigenvectors
          eigenvalues, eigenvectors = np.linalg.eigh(eColiData_cov)
          print('Unsorted eign values: \n%s', eigenvalues)
          print('Unsorted eign values: \n%s', eigenvectors)
```

```
value of Det(Cov)
0.13729373206325216
Unsorted eign values:
%s [0.12680082 0.48008494 0.671441   0.85971779 1.20508931 1.46555238
 2.21220928]
Unsorted eign values:
%s [[-0.08154845  0.75965086 -0.00490985 -0.06707282 -0.29219921  0.37265574
  -0.43301833]
 [-0.18599716 -0.60363588  0.05019625 -0.14655068 -0.55288668  0.47029644
  -0.22398568]
 [-0.10478843 -0.10724976 -0.67128037  0.20125402  0.49223261  0.48177748
  -0.10951653]
 [ 0.01462969 -0.03639389  0.67163159 -0.20553058  0.5451792   0.45560856
  -0.01816912]
 [ 0.01385432 -0.09557701  0.29459176  0.88426143 -0.00818155 -0.04745278
  -0.34590722]
 [ 0.70393508 -0.16949665 -0.09365082 -0.24341844  0.11045841 -0.16428058
  -0.6070701 ]
 [-0.67219322 -0.08867463  0.01396843 -0.22384663  0.23907035 -0.4135461
  -0.51167924]]
```

```
In [170]: #sort eigenvalues to get higher values
          idx = np.argsort(-eigenvalues)
          #apply index to sort eigenvalues
          eigenvalues = eigenvalues[idx]
          print('Sorted eigen values: ', eigenvalues)

          #sort eigenvectors
          eigenvectors = eigenvectors[idx]
          #print ('Eigen vector shape: ', eigenvectors.shape)
          #print ('Sorted Eigenvectors: ', eigenvectors)

          #calculate Singular value decomp
          S = eigenvalues*np.eye(7)
          U = eigenvectors
          V = U
          D = np.dot(U, np.dot(S,V.T))
          #print ('D value: ')
          #print(D)

          #validate SVD results using np svd function
          U1, S1, V1 = np.linalg.svd(X_std.T)
          print('Validate U1 value: ')
          print(U1)
```

```
Sorted eigen values:  [2.21220928 1.46555238 1.20508931 0.85971779 0.671441   0.48008494
 0.12680082]
Validate U1 value:
[[-0.43301833  0.37265574 -0.29219921  0.06707282  0.00490985  0.75965086
   0.08154845]
 [-0.22398568  0.47029644 -0.55288668  0.14655068 -0.05019625 -0.60363588
   0.18599716]
 [-0.10951653  0.48177748  0.49223261 -0.20125402  0.67128037 -0.10724976
   0.10478843]
 [-0.01816912  0.45560856  0.5451792   0.20553058 -0.67163159 -0.03639389
  -0.01462969]
 [-0.34590722 -0.04745278 -0.00818155 -0.88426143 -0.29459176 -0.09557701
```

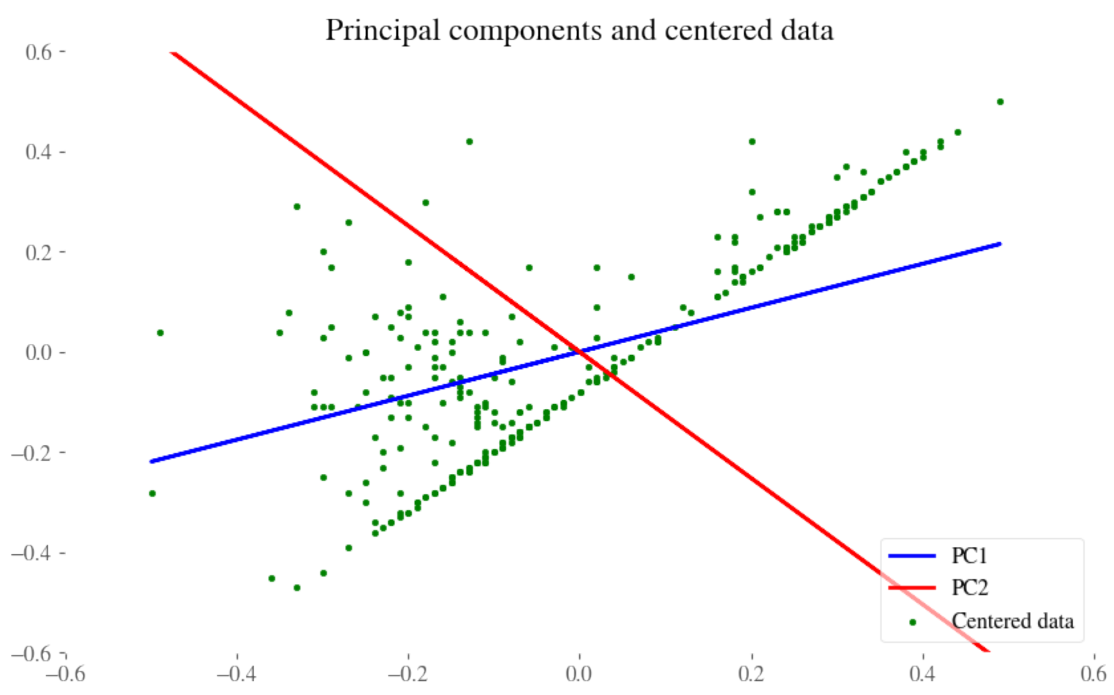
```
  -0.01385432]
 [-0.6070701  -0.16428058  0.11045841  0.24341844  0.09365082 -0.16949665
  -0.70393508]
 [-0.51167924 -0.4135461   0.23907035  0.22384663 -0.01396843 -0.08867463
   0.67219322]]
```

0.0.7 Problem 2d (10 points)

```
Compute the projection of the raw data onto the appropriate two eigenvectors.  Consider which
```

Answer: Considering the last two feature vectors alm2, alm1 to be projected

```
In [171]: #finding orientation for axes
          h1 = (eigenvectors[6][0]/eigenvectors[5][0])*eColiData_0.alm2
          h2 = (eigenvectors[6][1]/eigenvectors[5][1])*eColiData_0.alm2

In [172]: #ploting the PCA
          ax = plt.gca()
          ax.plot(eColiData_0.alm2, h1, color='blue', label='PC1')
          ax.plot(eColiData_0.alm2, h2, color='red', label='PC2')
          ax.axis([-0.6, 0.6, -0.6, 0.6])
          ax.scatter(eColiData_0.alm2, eColiData_0.alm1, s=10, c='green', label='Centered data
          ax.set_title('Principal components and centered data')
          ax.grid(True)
          #cible_border()
          legend = ax.legend(loc=4,frameon=True,framealpha=0.6, scatterpoints=1)
          rect = legend.get_frame()
          #rect.set_facecolor(light_grey)
          rect.set_linewidth(0.6)
          plt.show()
```



```
In [ ]:
```

0.0.8 Problem 2e (10 points)

Plot the projected points such that the 8 different classes can be visually identified. Be sure to label the classes and axes. Commont on the quality of the separation of the different classes using PCA.

```
In [173]: from sklearn.decomposition import PCA as sklearnPCA

          sklearn_pca = sklearnPCA(n_components=2)
          pca_fit = sklearn_pca.fit_transform(X_std)

In [ ]: x1 = pca_fit[:,0]
        y1 = pca_fit[:,1]
        n = eColiData.iloc[:,8:9]

        ax = plt.gca()
        ax.axis([-3, 3, -3, 3])
        ax.scatter(x1, y1, s=10, c='green', label='Centered data')

        for i, txt in enumerate(n.values):
            ax.annotate(txt[0], (x1[i],y1[i]), zorder=1, size=14, color=shade_black)

        ax.set_title('Principal components and centered data')
        ax.grid(True)
```

```
#cible_border()
legend = ax.legend(loc=4,frameon=True,framealpha=0.6, scatterpoints=1)
rect = legend.get_frame()
#rect.set_facecolor(light_grey)
rect.set_linewidth(0.6)
plt.show()
```

0.0.9 Problem 2f (10 points)

The PCA that you have just completed takes each data point and projects it using a weighted sum of features. One could also do the opposite to map the features as a weighted sum of the data entries. How could this be done? What is a potential issue? Describe these in a few sentences (do not code it).

Answer: To map features as weighted sum of data entries we would need to transpose each of the EColi variation into a feature. For this we would need to transpose each row of the data to a feature vector and then apply the same steps of PCA. However by doing it this way we end up with very high number of features and data would be more localized. This may not help with analysis of the overall protien localization for entire dataset.

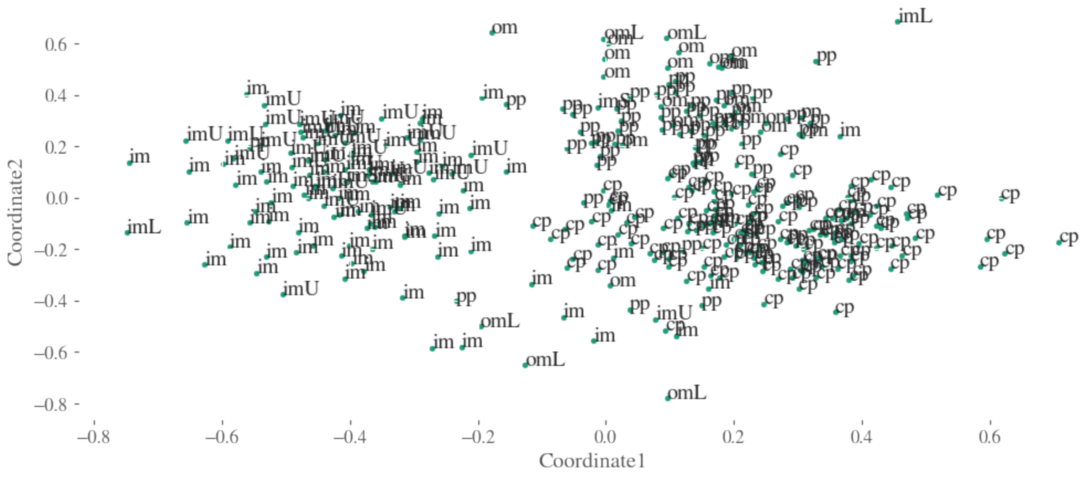
0.0.10 Problem 3 MDS (10 points)

For the same data set, repeat 2e using sklearn's Multidimensional scaling algorithm.

```
In [174]: from sklearn.metrics.pairwise import cosine_similarity, euclidean_distances
          from sklearn.manifold import MDS

          #get eculidean dist
          Euclid_dist = euclidean_distances(eColiData_num)

          #Default metric = True, for NonMetric set metric=False
          mds = MDS(n_components=2, dissimilarity="precomputed", random_state=99)

          mds_fit = mds.fit_transform(Euclid_dist)  # shape (n_samples, n_components)
```

```
In [175]: #plot the MDS fit

          x1 = mds_fit[:,0]
          y1 = mds_fit[:,1]
          n = eColiData.iloc[:,8:9]

          fig, ax = plt.subplots(figsize=(12,5))

          dark2_colors = ['#1b9e77','#d95f02','#7570b3','#e7298a','#66a61e','#e6ab02','#a6761d
          shade_black = '#262626'
          ax.scatter(x1, y1, s=10, zorder=0, c=dark2_colors[0])

          for i, txt in enumerate(n.values):
              ax.annotate(txt[0], (x1[i],y1[i]), zorder=1, size=14, color=shade_black)
```

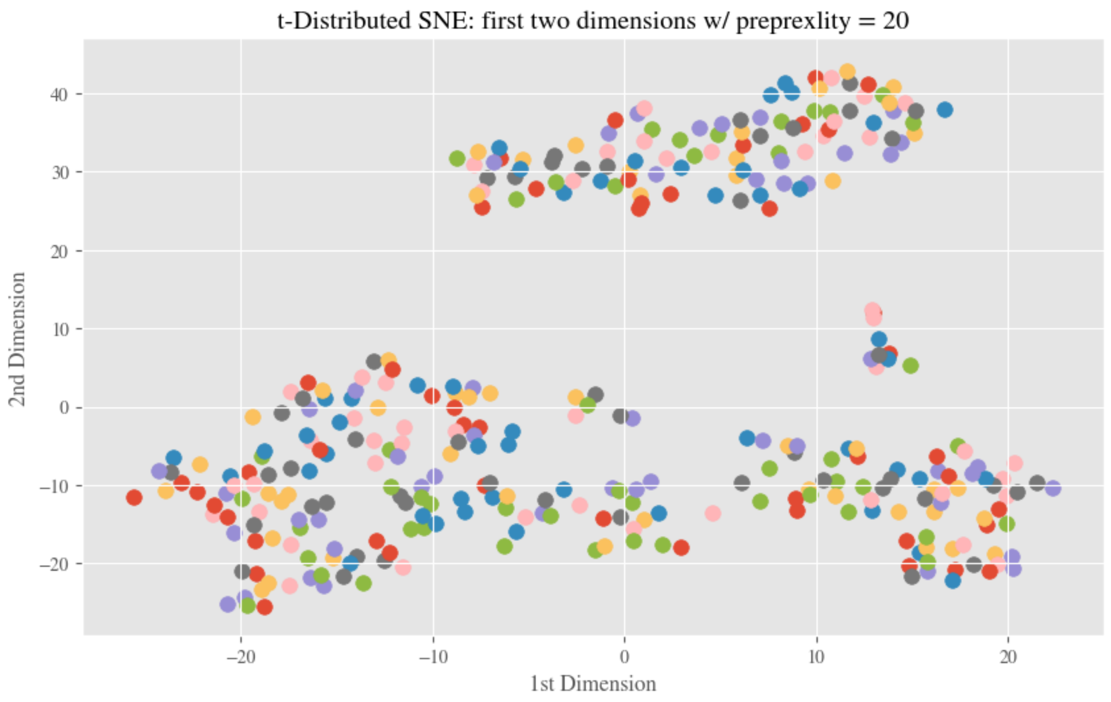
```
        ax.set_xlabel('Coordinate1')
        ax.set_ylabel('Coordinate2');
```



0.0.11 Problem 4a t-SNE (5 points)

Repeat 2e using a t-SNE plot with the default settings.
    Answer: Running t-SNE with components=2 and preplexity=20

```
In [176]: from sklearn.manifold import TSNE

          #run TSNE with 2 components and preplexity 20
          X_embedded = TSNE(n_components=2, perplexity=20, verbose=0).fit_transform(eColiData_(

In [177]: mpl.style.use('ggplot')

          plt.figure(figsize=(10,6))
          for i in range(X_embedded.shape[0]):
              plt.scatter(X_embedded[i,0], X_embedded[i,1], s = 70, zorder=0)
          plt.ylabel('2nd Dimension')
          plt.xlabel('1st Dimension')
          plt.title('t-Distributed SNE: first two dimensions w/ preprexlity = 20')
          plt.show()
```



0.0.12 Problem 4b t-SNE perplexity (5 points)

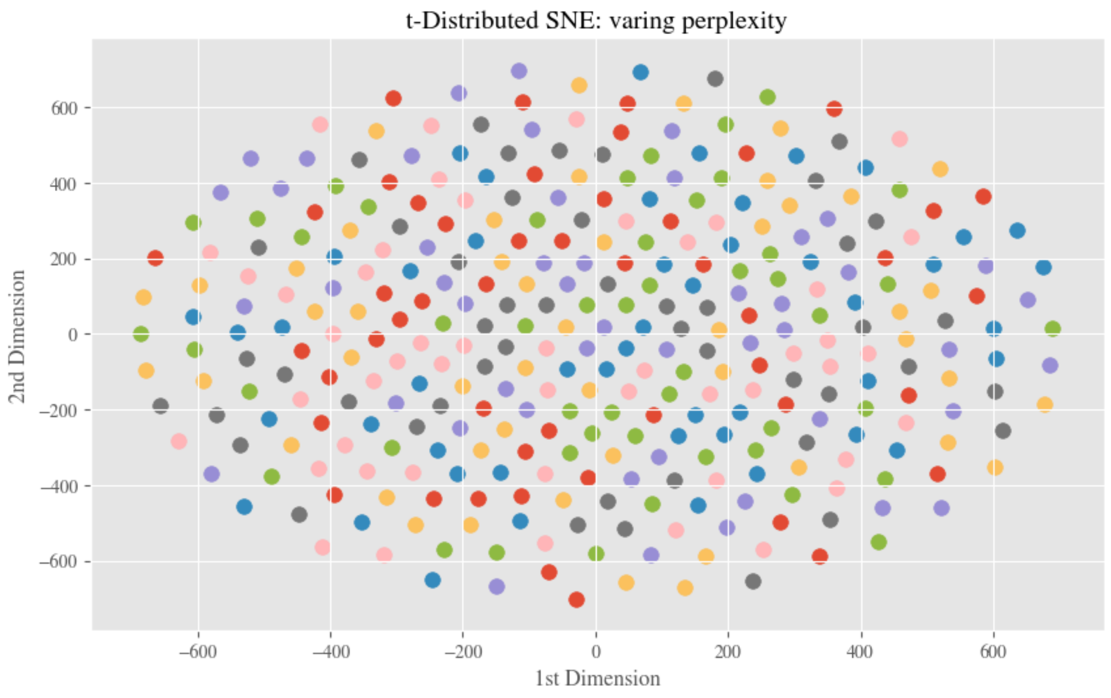
Try out a few t-SNE plots by varying the perplexity. State the best perplexity for separating the 8 different classes and describe your rationale in a sentence or two. Report the average calculation time for the t-SNE projection over a number of iterations.

Answer: Looking at the below points, preplexity <20 can be used for more localized visualizations and analysis. Having preplexity >20 helps with more distinct clusters and helps with more higher level (global) analysis for the 8 different classes.

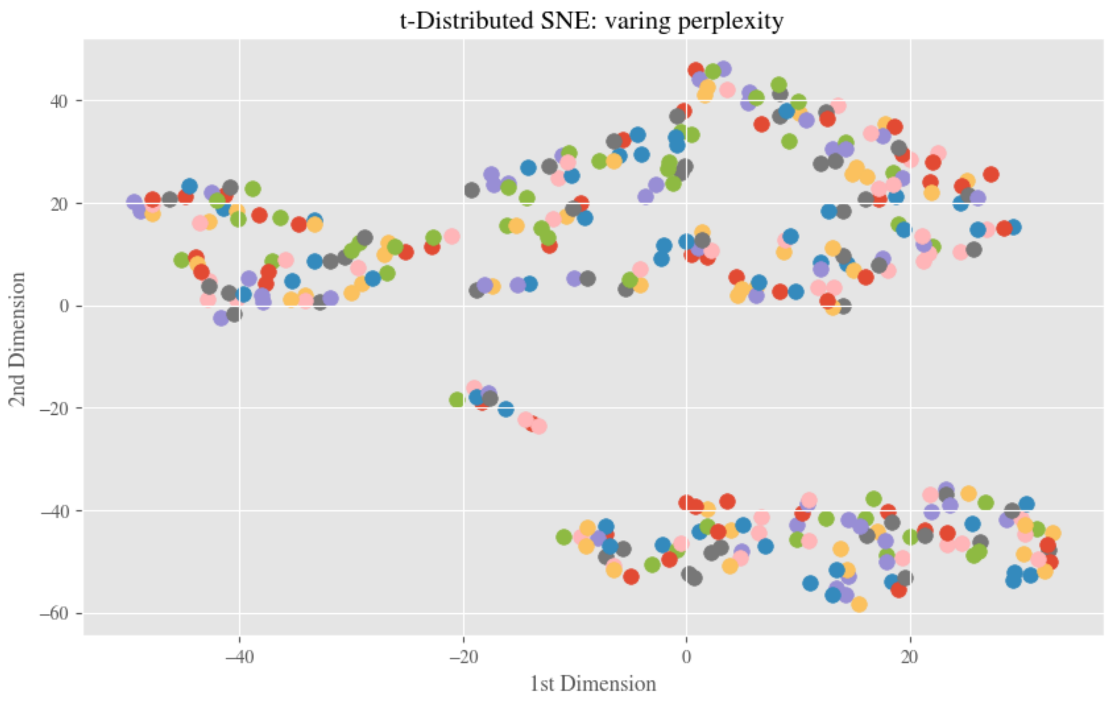
Also looking at times taken, it is a less than 10 seconds for preplexity = 0 and 10 and then staying constant after prelexity >=20

```
In [179]: import timeit

          for iCounter in range(0,60,10):
              start = timeit.default_timer()
              for nloop in range(11):
                  X_embedded = TSNE(n_components=2, perplexity=iCounter, verbose=0).fit_transf
              stop = timeit.default_timer()
              print('Time taken: ', round((stop - start)/nloop,2), 'in seconds, varing preplex

              #plotting the t-SNE
              mpl.style.use('ggplot')
              plt.figure(figsize=(10,6))
              for i in range(X_embedded.shape[0]):
                  plt.scatter(X_embedded[i,0], X_embedded[i,1], s = 70, zorder=0)
              plt.ylabel('2nd Dimension')
              plt.xlabel('1st Dimension')
              plt.title('t-Distributed SNE: varing perplexity')
              plt.show()
```

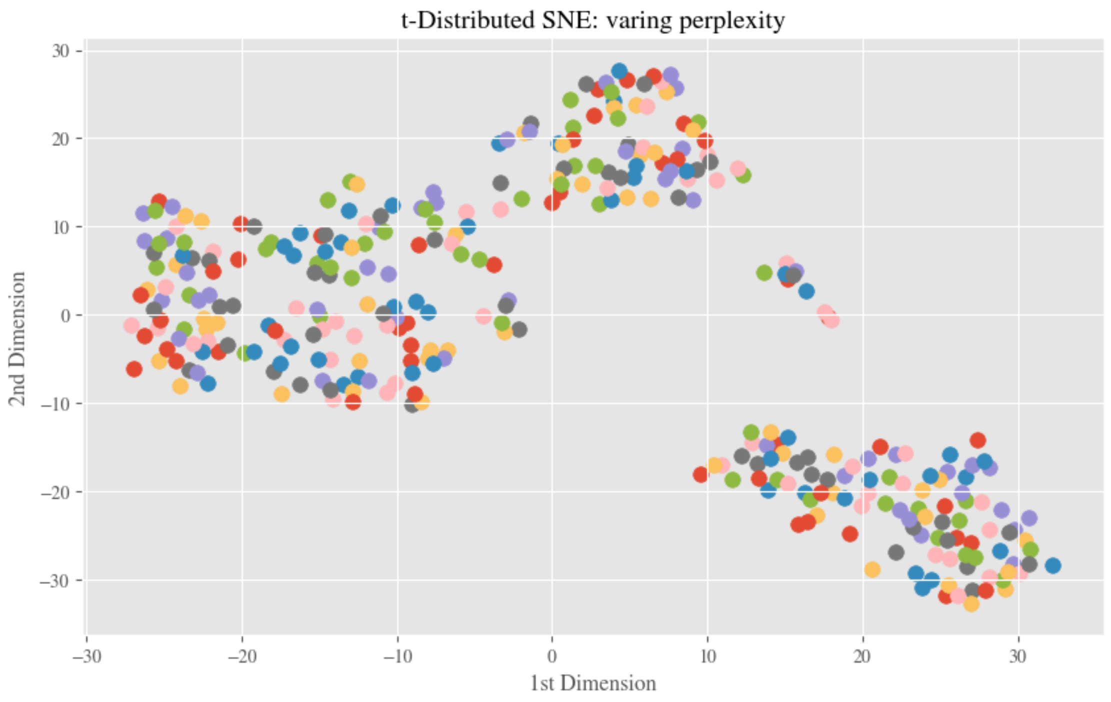
```
Time taken:  2.5 in seconds, varing preplexity = 0  and  10  iterations
```



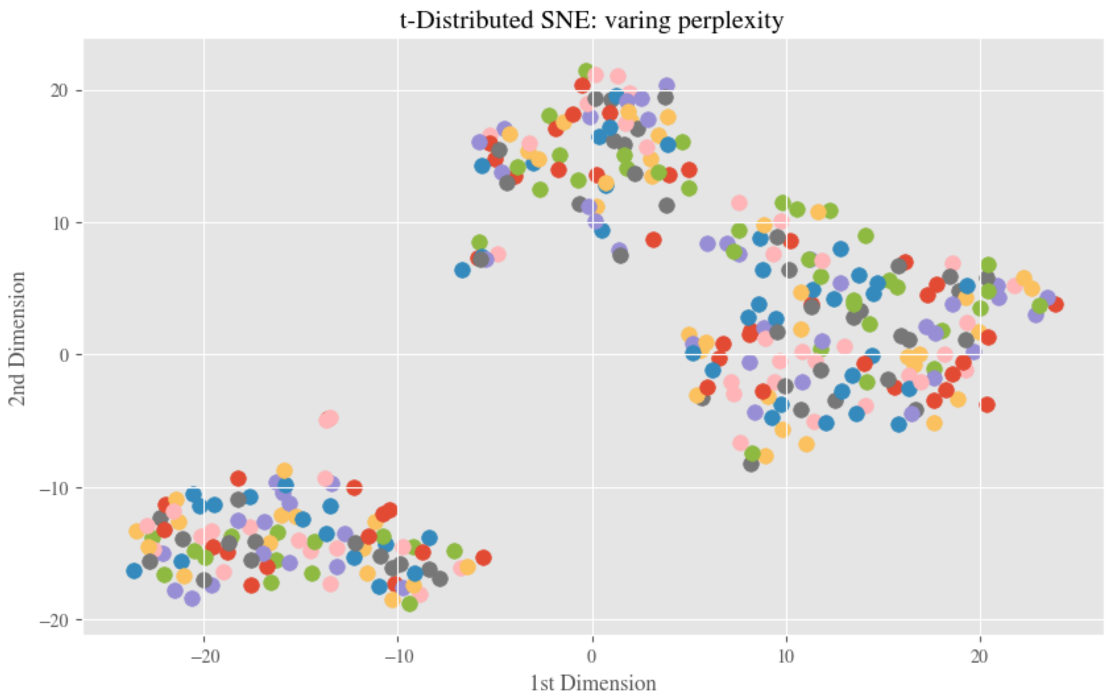
```
Time taken:  5.0 in seconds, varing preplexity = 10  and  10  iterations
```



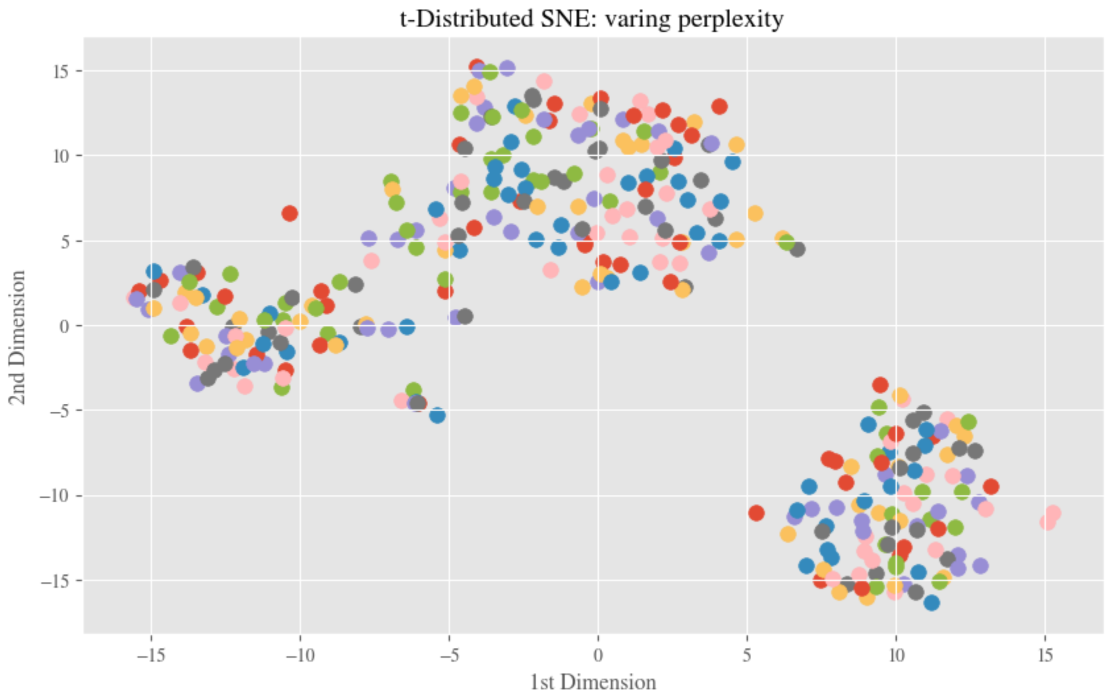
```
Time taken:  6.65 in seconds, varing preplexity = 20  and  10  iterations
```



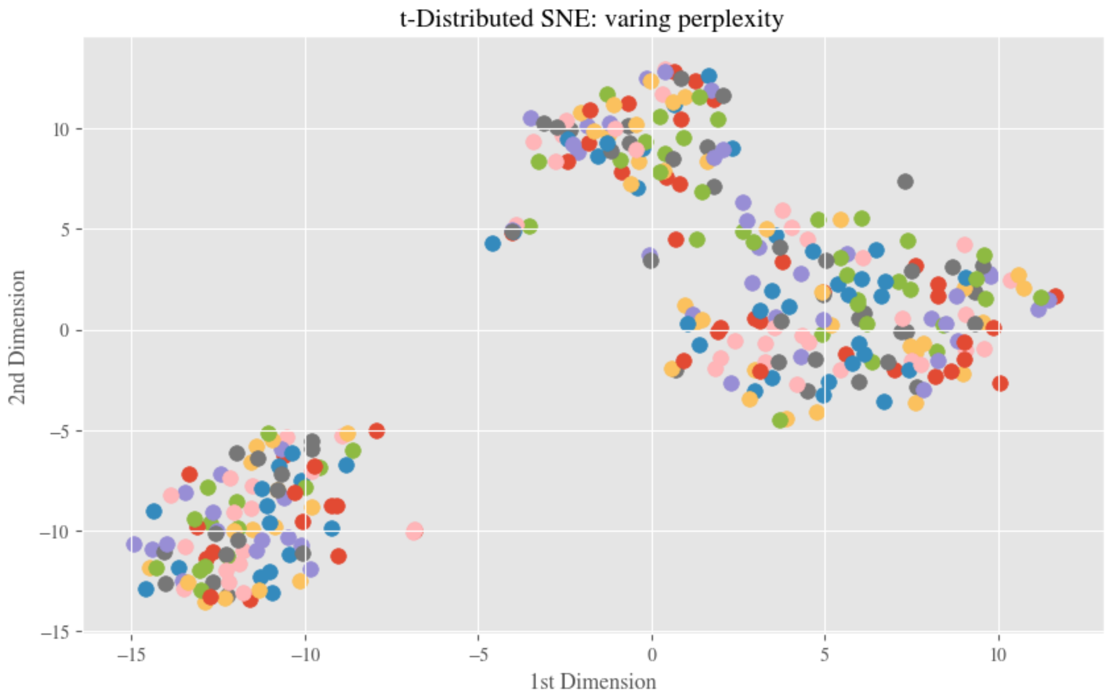
```
Time taken:  8.6 in seconds, varing preplexity = 30  and  10  iterations
```



```
Time taken:  11.04 in seconds, varing preplexity = 40  and  10  iterations
```



```
Time taken:  11.14 in seconds, varing preplexity = 50  and  10  iterations
```



0.0.13 Problem 4c t-SNE randomization (10 points)

The S of t-SNE means stochastic or random, usually as a function of time. Explore whether you can reproduce the result in 4b through a second projection and plot.

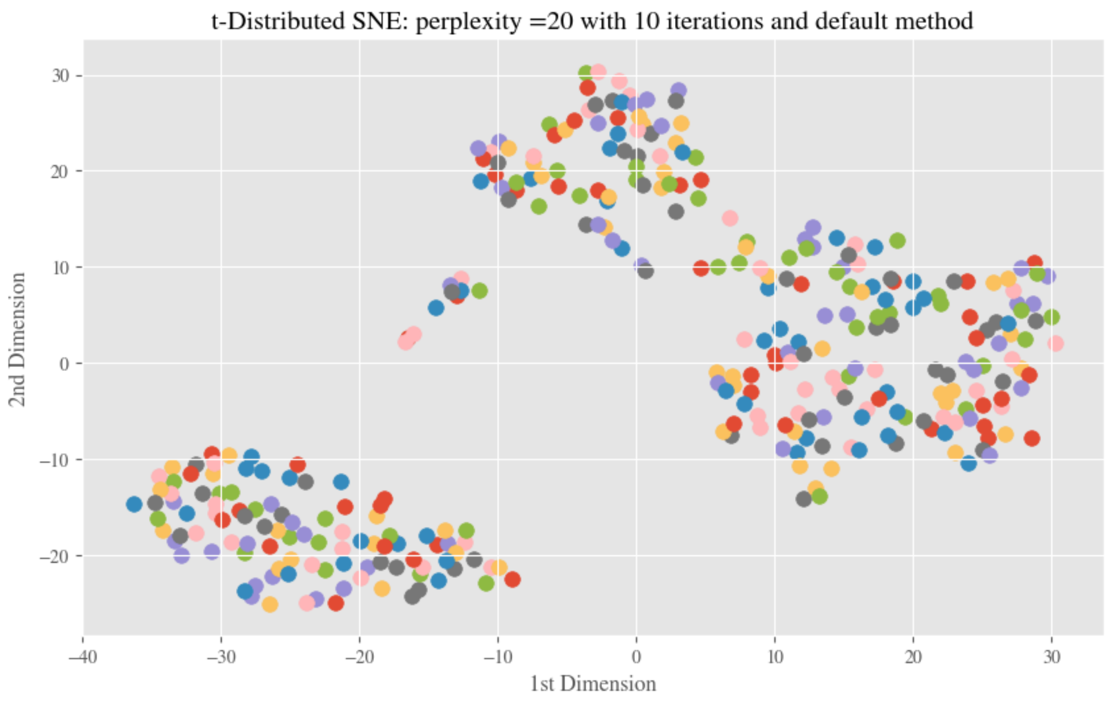
0.1 Problem 4d t-SNE Barnes-Hut (5 points)

The default t-SNE method of 4b uses the Barnes-Hut approximation. Keeping the other parameters the same as 4b, plot the t-SNE result using the exact method. Which method do you prefer? Compare the average calculation time for the exact method over a number of iterations.

Answer: Looking at the iteration times for 10 iterations, we see using method exact increased run time by 3 times. This can be a potential problem for very large datasets.

```
In [181]: import timeit

          start = timeit.default_timer()

          for nloop in range(11):
              X_embedded_20 = TSNE(n_components=2, perplexity=20, verbose=0).fit_transform(eCol
          stop = timeit.default_timer()
          print('Time: ', stop - start, 'seconds')

          mpl.style.use('ggplot')

          plt.figure(figsize=(10,6))
          for i in range(X_embedded_20.shape[0]):
              plt.scatter(X_embedded_20[i,0], X_embedded_20[i,1], s = 70, zorder=0)
          plt.ylabel('2nd Dimension')
          plt.xlabel('1st Dimension')
          plt.title('t-Distributed SNE: perplexity =20 with 10 iterations and default method')
          plt.show()
```

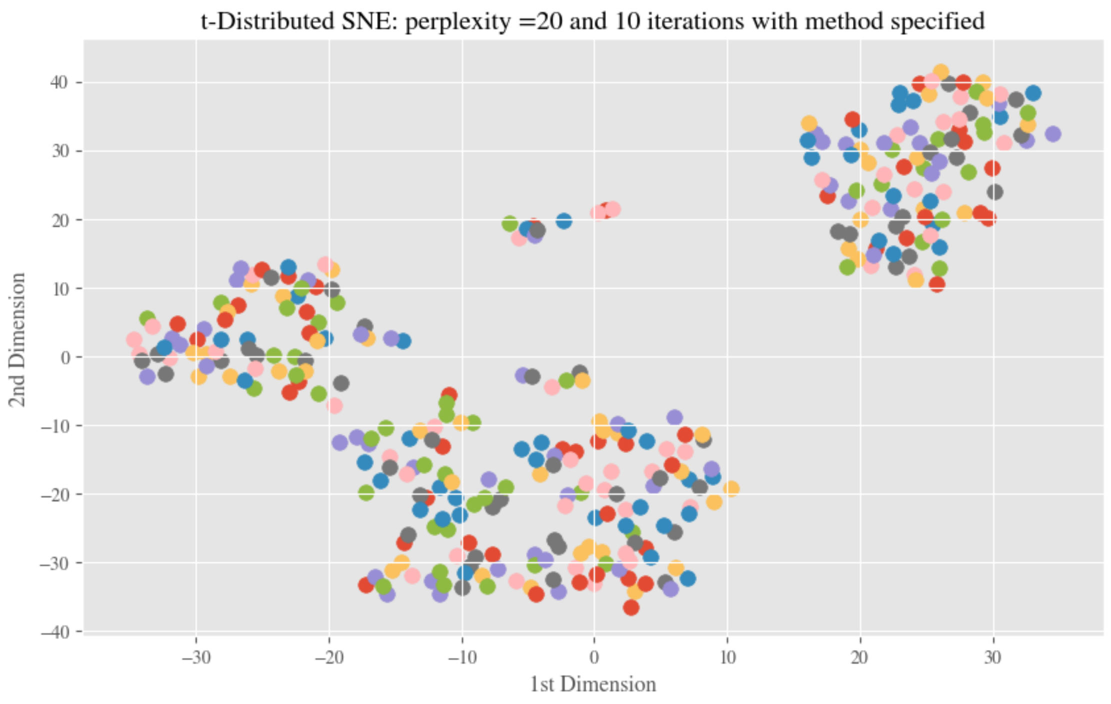
```
Time:  68.4031629940655 seconds
```



```
In [182]: start = timeit.default_timer()

          for nloop in range(11):
              X_embedded_20_s = TSNE(n_components=2, perplexity=20, verbose=0, method='exact')

          stop = timeit.default_timer()
          print('Total Time: ', stop - start, 'seconds')
```

```
#plot tSNE
mpl.style.use('ggplot')

plt.figure(figsize=(10,6))
for i in range(X_embedded_20_s.shape[0]):
    plt.scatter(X_embedded_20_s[i,0], X_embedded_20_s[i,1], s = 70, zorder=0)
plt.ylabel('2nd Dimension')
plt.xlabel('1st Dimension')
plt.title('t-Distributed SNE: perplexity =20 and 10 iterations with method specified
plt.show()
```

```
Total Time:  87.99440425306966 seconds
```



0.1.1 How many hours did this homework take?

This will not affect your grade. We will be monitoring time spent on homework to be sure that we are not over-burdening students.

```
In [ ]: 6 Hrs
```

0.1.2 Last step (5 points)

Save this notebook as LastnameFirstnameHW1.ipynb such as MuskElonHW1.ipynb. Create a pdf of this notebook named similarly. Submit both the python notebook and the pdf version to the Canvas dropbox. We require both versions.