Daniel Douglass

423276

HW2

1.In addition our algorithm must be run multiple times in order to determine the best fit for the data because our initialization will affect what our Gaussian distribution will result in.

b. After running my algorithm multiple times from different initializations and sklearn gmm I got two normal distributions with the following parameters.

Mu1 = 15.514506938, std1 =5.13739 p1 = 0.56946150

Mu2 = 43.352883980, std2 =8.21745 p2 = 0.43053849

Python Script:

**import** **numpy** **as** **np**

**from** **sklearn** **import** mixture

**from** **math** **import** log

In [4]:

X = [[35], [41], [21], [20], [17], [55], [12], [33], [15], [18], [4], [51], [17], [46]]

gmm = mixture.GMM(2,n\_init=100).fit(X)

gmm.means\_

Out[4]:

15.50703948

43.30578194

In [118]:

gmm.covars\_

Out[118]:

26.37059923

68.50908112

In [83]:

**def** computeProbability(x,Std,Mu):

**return**(np.exp(-1\*(x-Mu)\*\*2/(2\*Std\*\*2))/(np.sqrt(2\*np.pi\*Std\*\*2)))

In [101]:

**def** computeGaussianMixtureModel(X1,mu,std,p):

N = len(X1)

N1 = np.zeros(N)

N2 = np.zeros(N)

prevLogLikelihood= 10000

NewlogLikelihood = 0

**while** abs(prevLogLikelihood -NewlogLikelihood)>.001 :

prevLogLikelihood = NewlogLikelihood

**for** j **in** range(0,N):

x = X1[j]

N1[j] = computeProbability(x,std[0],mu[0])\*p[0]/(computeProbability(x,std[0],mu[0])\*p[0]+computeProbability(x,std[1],mu[1])\*p[1])

N2[j] = 1 - N1[j]

p = [sum(N1)/N,sum(N2)/N]

mu = [np.dot(N1,X)/sum(N1),np.dot(N2,X)/sum(N2)]

NewlogLikelihood = sum([log(y,10) **for** y **in** N1]) + sum([log(y,10) **for** y **in** N2])

xsqtemp1 = np.square([x - mu[0] **for** x **in** X1])

xsqtemp2 = np.square([x - mu[1] **for** x **in** X1])

std = [np.sqrt(np.dot(N1,xsqtemp1)/sum(N1)), np.sqrt(np.dot(N2,xsqtemp2)/sum(N2))]

**return** mu,std,p

In [102]:

X1 = [35, 41, 21, 20, 17, 55, 12, 33, 15, 18, 4, 51, 17, 46]

p = [.7,.3]

Mu = [32,40]

std = [7,5]

computeGaussianMixtureModel(X1,Mu,std,p)

Out[102]:

([array([ 15.51450694]), array([ 43.35288398])],

[array([ 5.13739488]), array([ 8.21745879])],

[0.56946150117923078, 0.43053849882076917])

In [109]:

p = [.4,.6]

Mu = [39,45]

std = [4,5]

computeGaussianMixtureModel(X1,Mu,std,p)

Out[109]:

([array([ 15.51449538]), array([ 43.35282842])],

[array([ 5.13738752]), array([ 8.21752672])],

[0.56946040538464371, 0.43053959461535635])

xtemp = [[x - Mu[0] for x in X1], [x - Mu[1] for x in X1]] xtemp

In [110]:

p = [.1,.9]

Mu = [10,45]

std = [4,5]

computeGaussianMixtureModel(X1,Mu,std,p)

Out[110]:

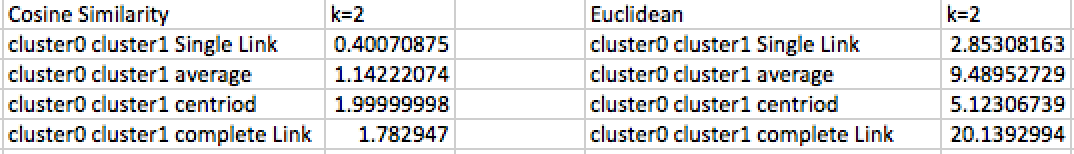
([array([ 15.51450355]), array([ 43.3528695])],

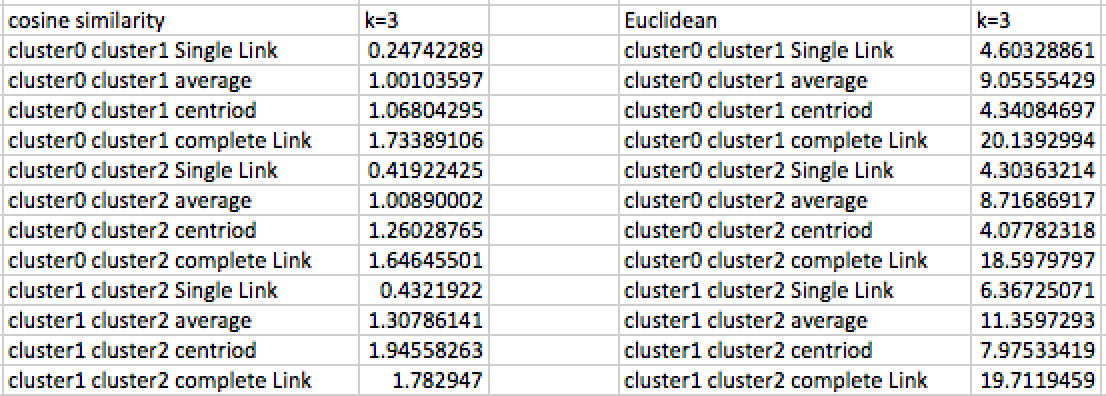
[array([ 5.1373923]), array([ 8.21747615])],

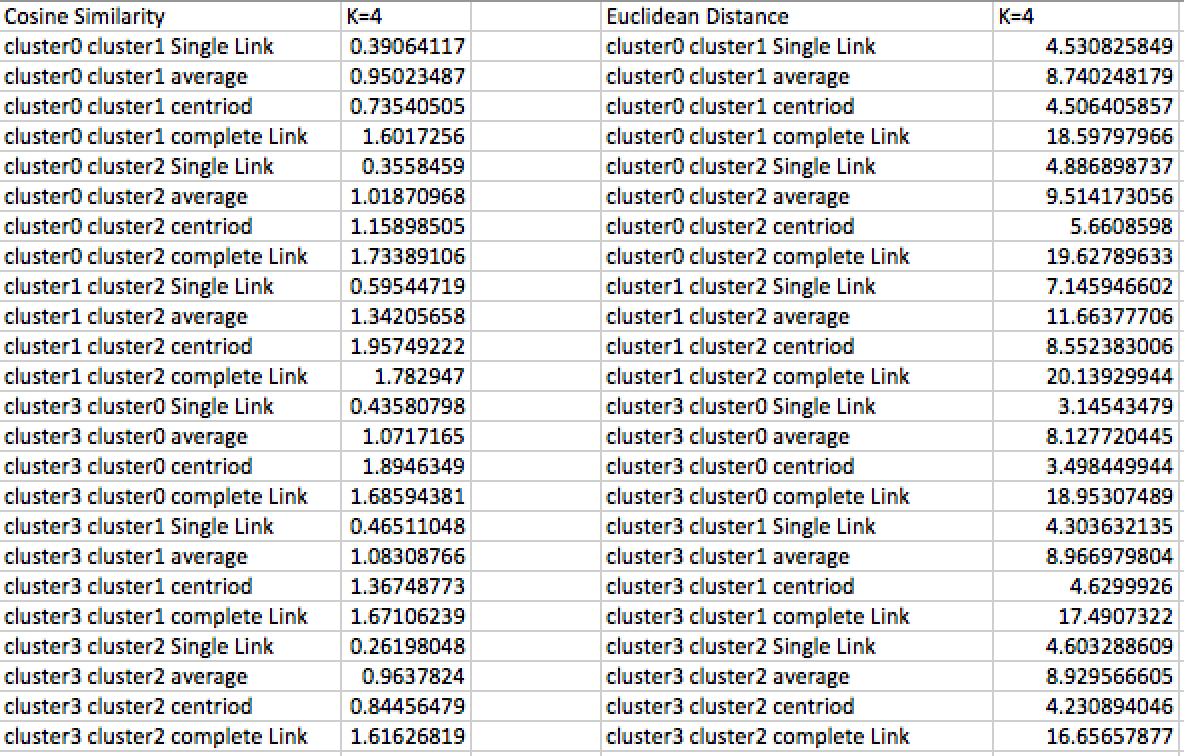
[0.56946120786151355, 0.4305387921384865])

2.

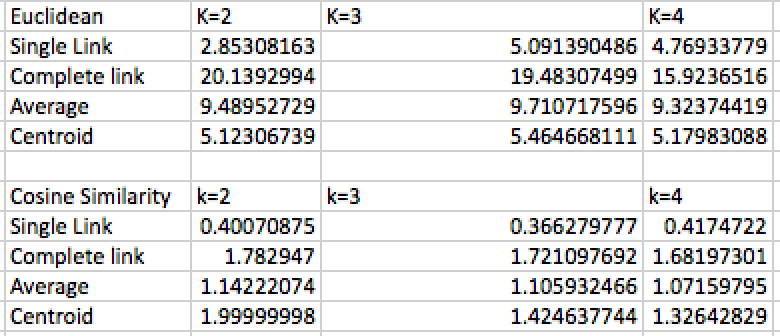
**Distances between clusters based on distance measure and number of clusters**







**Averaged distances for different distance measures across clusters**



Above is a table of the clusters taken by using a distance measure of Euclidean distance and cosine similarity. The cosine similarity is 1-normalized dot product thus has a range between 0 and 2. For cosine similarities closer to 1 represent two spaces that are more dissimilar to each other due to them being orthogonal.

Looking at the Euclidean distance between clusters we can see as K increases from 2 to 3, we have a significant increase in the single link distance. This is important as k=3 provides a greater separation than k=2, creating a better distinction between clusters. K=4 has a slightly smaller Euclidean distance compared to k=3 for single link, signifying k=3 provides the greatest minimal distance between clusters. The average distance and centroid distances when calculated with Euclidean distance show that k=3 provide the largest distance for both metrics. This shows that k=3 separate the data into the most distinction groups as it provides the largest difference of the averages between the groups. When choosing which k to implement, k=3 provides the best clustering based on that it provides the largest single link, Average and Centroid distances.

When comparing the distances based on Euclidean distances we see that K=4 provides for all different distance measures (single link, complete link etc.) the closest value to 1. This tells us that k=4 produce the most dissimilar cluster groups in relation to each other. This tells us that k=4 do a good job of separating the data when evaluated with cosine similarity.

To deal with missing data I filled in the missing data with the mean for the respective gene. As the benefit is that it doesn’t change the sample mean for the feature.

3.a. below are the top 200 features from SVM using a linear Kernel.

|  |
| --- |
| GI\_38327038-I |
| GI\_37543009-S |
| GI\_10835229-S |
| GI\_45446741-S |
| GI\_23238193-A |
| GI\_21359861-S |
| GI\_22035587-A |
| GI\_18641372-S |
| GI\_45598382-S |
| GI\_34222316-S |
| GI\_38016132-S |
| GI\_19913395-I |
| GI\_4507112-S |
| GI\_31343498-A |
| GI\_4502630-S |
| GI\_33946283-I |
| GI\_4557726-S |
| GI\_4503874-S |
| GI\_6912533-S |
| GI\_31077210-A |
| GI\_19923414-S |
| GI\_5453596-S |
| GI\_32313568-S |
| GI\_45238848-S |
| GI\_42661601-S |
| GI\_5901937-S |
| GI\_40806213-S |
| GI\_13899296-S |
| GI\_19557644-S |
| GI\_27498358-S |
| GI\_37548529-S |
| GI\_4757909-S |
| GI\_4557348-S |
| GI\_19923110-S |
| GI\_13899226-S |
| GI\_8051578-S |
| GI\_8922994-S |
| GI\_13430873-S |
| GI\_31341734-S |
| GI\_18641378-S |
| GI\_23957681-S |
| GI\_38327635-S |
| GI\_4757849-S |
| GI\_13540508-S |
| GI\_22035654-A |
| GI\_27485046-S |
| GI\_41393558-I |
| GI\_27894372-A |
| GI\_40254998-S |
| GI\_8051576-A |
| GI\_16753224-S |
| GI\_45545436-I |
| GI\_32171240-S |
| GI\_34147581-S |
| GI\_40217818-S |
| GI\_42655923-S |
| GI\_4504482-S |
| GI\_21389570-S |
| GI\_39777591-S |
| GI\_4502660-S |
| GI\_41872473-S |
| GI\_20357531-S |
| GI\_21389432-S |
| GI\_38261964-A |
| GI\_33946335-S |
| GI\_41327755-S |
| GI\_21361946-S |
| GI\_22538441-S |
| GI\_27437022-A |
| GI\_21040361-A |
| GI\_31343498-I |
| GI\_42558249-I |
| GI\_22095346-S |
| GI\_5453687-S |
| GI\_37594443-S |
| GI\_34916047-S |
| GI\_34452689-S |
| GI\_23943885-S |
| GI\_29029631-S |
| GI\_8924224-S |
| GI\_21361261-S |
| GI\_34147673-S |
| GI\_27894352-A |
| GI\_21735557-A |
| GI\_33598917-S |
| GI\_17921992-I |
| GI\_34222335-S |
| GI\_42661835-S |
| GI\_32483396-S |
| GI\_27475984-S |
| GI\_22749082-S |
| GI\_37550155-S |
| GI\_31543909-S |
| GI\_18765747-A |
| GI\_23957699-S |
| GI\_39930526-S |
| GI\_39841072-S |
| GI\_31317298-A |
| GI\_42656398-S |
| GI\_4504794-S |
| GI\_4503872-I |
| GI\_22055338-S |
| GI\_4503680-S |
| GI\_41327153-S |
| GI\_21389392-S |
| GI\_17981703-S |
| GI\_34916027-S |
| GI\_13129143-S |
| GI\_31542734-S |
| GI\_42661719-S |
| GI\_8923465-S |
| GI\_24497521-I |
| GI\_33636718-S |
| GI\_17978499-A |
| GI\_21312135-S |
| GI\_4503562-S |
| GI\_7262372-S |
| GI\_11184225-S |
| GI\_40254432-S |
| GI\_4758697-S |
| GI\_17975764-A |
| GI\_7661869-S |
| GI\_31982866-S |
| GI\_37558299-S |
| GI\_42476063-S |
| GI\_21614543-S |
| GI\_31377723-S |
| GI\_12232388-S |
| GI\_27894383-S |
| GI\_31542848-S |
| GI\_44771157-S |
| GI\_40538795-S |
| GI\_40255242-S |
| GI\_33386702-S |
| GI\_41281554-S |
| GI\_40018630-A |
| GI\_6912379-S |
| GI\_42661634-S |
| GI\_33504488-S |
| GI\_13430859-S |
| GI\_21361081-S |
| GI\_7706468-S |
| GI\_37551895-S |
| GI\_14165257-S |
| GI\_4755136-S |
| GI\_21614536-I |
| GI\_6031156-S |
| GI\_4503320-S |
| GI\_29893558-S |
| GI\_4505564-S |
| GI\_37552150-S |
| GI\_33354256-S |
| GI\_31621298-S |
| GI\_7706116-S |
| GI\_4506756-S |
| GI\_45592953-S |
| GI\_32171246-I |
| GI\_13375845-S |
| GI\_14042952-S |
| GI\_14670363-I |
| GI\_15718672-S |
| GI\_39995079-S |
| GI\_20357536-I |
| GI\_30150520-S |
| GI\_4557504-S |
| GI\_38158004-A |
| GI\_31542586-S |
| GI\_33300650-S |
| GI\_8922595-S |
| GI\_4502854-S |
| GI\_37541634-S |
| GI\_31747573-S |
| GI\_42661344-S |
| GI\_31712021-S |
| GI\_34147526-S |
| GI\_25092724-S |
| GI\_4580421-A |
| GI\_28195383-S |
| GI\_24432076-S |
| GI\_38348371-S |
| GI\_28557782-I |
| GI\_7657503-S |
| GI\_41222114-S |
| GI\_4505050-S |
| GI\_38570131-S |
| GI\_40549416-S |
| GI\_34485729-S |
| GI\_10346134-S |
| GI\_4503174-S |
| GI\_42403584-A |
| GI\_37547347-S |
| GI\_25952086-S |
| GI\_13375757-S |
| GI\_22748812-S |
| GI\_34222221-S |
| GI\_14110425-S |
| GI\_42476060-S |
| GI\_16507197-S |
| GI\_22208956-S |
| GI\_40804743-S |

b. Below are the top 200 features from SVM using a quadratic kernel. We can see that there are some important interaction features signifying that there was significance correlation between variables in predicting Alzheimer’s classes.

|  |
| --- |
| GI\_24432049-S^2 |
| GI\_20127596-S^2 |
| GI\_34328935-S |
| GI\_32171246-I |
| GI\_14149701-S^2 |
| GI\_31324542-S^2 |
| GI\_4557678-S |
| GI\_4585642-S |
| GI\_27475984-S |
| GI\_7706713-S |
| GI\_27475984-S^2 |
| GI\_22035587-A |
| GI\_42659113-S GI\_14149701-S |
| GI\_17017983-S |
| GI\_37574605-A |
| GI\_24432049-S GI\_7706713-S |
| GI\_4557678-S GI\_38201693-S |
| GI\_4557394-S GI\_24308064-S |
| GI\_21314649-S |
| GI\_21314760-S |
| GI\_38201693-S^2 |
| GI\_13375790-S GI\_24432049-S |
| GI\_45505129-S^2 |
| GI\_16950634-A^2 |
| GI\_23312375-A GI\_42659113-S |
| GI\_13376877-S GI\_31324542-S |
| GI\_42659113-S GI\_42658560-S |
| GI\_20127596-S GI\_31542532-S |
| GI\_4508018-S |
| GI\_34452172-S |
| GI\_14149701-S GI\_34222109-I |
| GI\_22749180-S GI\_13430873-S |
| GI\_31341099-S^2 |
| GI\_20127596-S GI\_16950634-A |
| GI\_4503314-S GI\_27475984-S |
| GI\_22035587-A GI\_24432049-S |
| GI\_31377655-S |
| GI\_21361878-S GI\_24432049-S |
| GI\_13787213-I GI\_5730010-S |
| GI\_14917110-S |
| GI\_20302037-S |
| GI\_14249161-S GI\_42659113-S |
| GI\_40068510-S |
| GI\_4557678-S^2 |
| GI\_11641417-S GI\_16604251-S |
| GI\_16604251-S^2 |
| GI\_40255181-S GI\_5730010-S |
| GI\_42658538-S |
| GI\_22748680-S GI\_16604251-S |
| GI\_42659113-S GI\_13787213-I |
| GI\_34915989-S^2 |
| GI\_45505129-S |
| GI\_19923616-S^2 |
| GI\_41584204-S^2 |
| GI\_28195383-S GI\_34147355-S |
| GI\_38505264-S GI\_31324542-S |
| GI\_24432049-S GI\_24308064-S |
| GI\_38679945-S |
| GI\_16950634-A GI\_31341099-S |
| GI\_24432049-S GI\_20357531-S |
| GI\_20127596-S GI\_21729887-A |
| GI\_22749180-S GI\_40068510-S |
| GI\_14149701-S GI\_27475984-S |
| GI\_40255181-S GI\_31324542-S |
| GI\_28195383-S GI\_13787213-I |
| GI\_22035587-A GI\_20357531-S |
| GI\_16604251-S GI\_13430873-S |
| GI\_34915989-S GI\_31324542-S |
| GI\_38201693-S |
| GI\_11641417-S GI\_34452172-S |
| GI\_20127571-S^2 |
| GI\_37543806-S GI\_28195383-S |
| GI\_41584204-S GI\_16950634-A |
| GI\_34147559-S |
| GI\_28195383-S GI\_11641417-S |
| GI\_27475984-S GI\_31324542-S |
| GI\_34915989-S GI\_38201693-S |
| GI\_32171246-I GI\_13430873-S |
| GI\_24430180-I |
| GI\_20127596-S |
| GI\_28195383-S GI\_21314760-S |
| GI\_42659113-S GI\_38176289-A |
| GI\_4557394-S GI\_7706713-S |
| GI\_20127571-S GI\_19923616-S |
| GI\_32171246-I GI\_38176289-A |
| GI\_34147355-S |
| GI\_13430873-S^2 |
| GI\_4758147-S GI\_31324542-S |
| GI\_21729887-A |
| GI\_21729887-A GI\_42659113-S |
| GI\_42659113-S GI\_31542532-S |
| GI\_23312375-A GI\_22749180-S |
| GI\_37574713-S |
| GI\_29171691-A |
| GI\_34222109-I^2 |
| GI\_32171195-S^2 |
| GI\_4557394-S GI\_11641417-S |
| GI\_24432049-S GI\_32171195-S |
| GI\_4503314-S |
| GI\_37543806-S GI\_21361878-S |
| GI\_4557394-S GI\_16604251-S |
| GI\_24432049-S GI\_31317255-A |
| GI\_21361878-S GI\_31442419-S |
| GI\_13376877-S GI\_38201693-S |
| GI\_4502408-S |
| GI\_21314649-S GI\_31324542-S |
| GI\_21729887-A GI\_21314649-S |
| GI\_22035587-A GI\_14149701-S |
| GI\_22749180-S GI\_42659113-S |
| GI\_37543806-S GI\_34452172-S |
| GI\_4885062-S^2 |
| GI\_21314649-S GI\_14149701-S |
| GI\_21729887-A GI\_23312375-A |
| GI\_13376877-S GI\_4557678-S |
| GI\_31341099-S GI\_42658560-S |
| GI\_34915989-S GI\_24308064-S |
| GI\_27475984-S GI\_4557678-S |
| GI\_45505129-S GI\_20336758-S |
| GI\_21314649-S^2 |
| GI\_27475984-S GI\_38201693-S |
| GI\_20127596-S GI\_41584204-S |
| GI\_40804754-S GI\_31324542-S |
| GI\_5730010-S^2 |
| GI\_42659113-S GI\_21450730-S |
| GI\_20127596-S GI\_22538496-S |
| GI\_21361878-S GI\_22035587-A |
| GI\_42659113-S GI\_34106709-A |
| GI\_22749180-S GI\_4503314-S |
| GI\_21729887-A GI\_34452172-S |
| GI\_31377719-S |
| GI\_40804754-S GI\_11641417-S |
| GI\_40804754-S GI\_27475984-S |
| GI\_28195383-S GI\_4885062-S |
| GI\_21729887-A GI\_14149701-S |
| GI\_21361878-S^2 |
| GI\_22538496-S GI\_42659113-S |
| GI\_24432049-S GI\_21450730-S |
| GI\_4885062-S GI\_38201693-S |
| GI\_20336758-S^2 |
| GI\_14149701-S GI\_16604251-S |
| GI\_22035587-A GI\_31324542-S |
| GI\_28195383-S GI\_21314649-S |
| GI\_23312375-A GI\_21314760-S |
| GI\_21729887-A GI\_38201693-S |
| GI\_38505264-S GI\_13376877-S |
| GI\_7657541-S GI\_16604251-S |
| GI\_31377655-S GI\_20357531-S |
| GI\_24432049-S GI\_16604251-S |
| GI\_42659113-S GI\_13376690-S |
| GI\_16950634-A |
| GI\_40255181-S GI\_27475984-S |
| GI\_38505264-S GI\_24432049-S |
| GI\_32261315-S GI\_13376877-S |
| GI\_20127596-S GI\_37543806-S |
| GI\_13376877-S GI\_45505129-S |
| GI\_32171246-I GI\_38201693-S |
| GI\_16604251-S GI\_21314760-S |
| GI\_27475984-S GI\_41584204-S |
| GI\_20357531-S GI\_31341099-S |
| GI\_42658538-S GI\_4557394-S |
| GI\_22749180-S GI\_31324542-S |
| GI\_21314649-S GI\_34915989-S |
| GI\_4502408-S GI\_16604251-S |
| GI\_20127596-S GI\_4557678-S |
| GI\_24432049-S GI\_32307180-S |
| GI\_42659113-S GI\_31324542-S |
| GI\_22749180-S GI\_24432049-S |
| GI\_40548379-S GI\_24432049-S |
| GI\_4508018-S GI\_42659113-S |
| GI\_24308064-S |
| GI\_38176289-A GI\_13430873-S |
| GI\_14149858-S GI\_8923658-S |
| GI\_23312375-A GI\_34452172-S |
| GI\_20127596-S GI\_5730010-S |
| GI\_32171246-I GI\_4557394-S |
| GI\_20127596-S GI\_4557394-S |
| GI\_32171246-I GI\_37543806-S |
| GI\_31542532-S |
| GI\_13376877-S GI\_24432049-S |
| GI\_22538496-S GI\_4885062-S |
| GI\_34452172-S GI\_19923616-S |
| GI\_21729887-A^2 |
| GI\_14149701-S GI\_45505129-S |
| GI\_14249161-S GI\_7706713-S |
| GI\_42658560-S GI\_13430873-S |
| GI\_7657541-S GI\_14149858-S |
| GI\_7706713-S GI\_19923616-S |
| GI\_20127596-S GI\_23312375-A |
| GI\_31377719-S GI\_14249161-S |
| GI\_40255181-S GI\_40068510-S |
| GI\_42659113-S GI\_34147559-S |
| GI\_34328935-S GI\_22035587-A |
| GI\_34106709-A GI\_38201693-S |
| GI\_23312375-A GI\_31542532-S |
| GI\_21729887-A GI\_40255181-S |
| GI\_16604251-S GI\_38176289-A |
| GI\_42659113-S GI\_34222109-I |
| GI\_20357531-S GI\_16604251-S |
| GI\_21314649-S GI\_45006902-S |

Appendix Python Code

Problem 2:

**import** **sklearn**

**from** **sklearn.ensemble** **import** RandomForestClassifier

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**from** **sklearn.cluster** **import** KMeans

**import** **random**

**from** **scipy.spatial.distance** **import** cdist

**from** **sklearn.preprocessing** **import** Imputer

**from** **collections** **import** defaultdict

**import** **itertools**

**from** **sklearn.metrics.pairwise** **import** cosine\_similarity

In [2]:

df = pd.read\_csv('/Users/Daniel/Documents/Data Mining 514/HW1/problem5/allPatients.csv',low\_memory=**False**)

In [3]:

X = df.drop('Classes',axis=1)

features = X.columns

X.replace(to\_replace='?', value = 'NaN', inplace=**True**)

imp = Imputer(missing\_values='NaN', strategy='mean', axis=0)

X=imp.fit\_transform(X)

Y = df['Classes']

X = pd.DataFrame(X,columns=features)

In [4]:

clf = RandomForestClassifier(n\_estimators=300, criterion = 'entropy', max\_features = 100)

clf.fit(X,Y)

freqFeatures = np.zeros(shape=(8560), dtype=int)

**for** tree **in** clf:

**for** i **in** tree.tree\_.feature:

**if** i !=-2:

freqFeatures[i] = freqFeatures[i]+1

top200Features = abs(freqFeatures).argsort()[::-1][:200]

pd.DataFrame(top200Features[:100]).to\_csv('/Users/Daniel/Documents/Data Mining 514/HW2/top100Features.csv')

featureSelectedX = X.ix[:,top200Features]

In [16]:

**def** getClusterGroups(labels,data,k):

d = dict();

**for** i **in** range(0,k):

d["cluster"+str(i)] = data.ix[np.nonzero(labels==i)]

**return** d

**def** getDistances(clusters,distanceMeasure):

distDict = dict()

**for** cluster1 **in** itertools.combinations(clusters, 2):

dist = cdist(clusters[cluster1[0]],clusters[cluster1[1]], distanceMeasure)

n,m = dist.shape

distDict[cluster1[0]+" " +cluster1[1] + " Single Link"] = np.nanmin(dist)

distDict[cluster1[0]+" " +cluster1[1] + " complete Link"] = np.nanmax(dist)

dist = np.nan\_to\_num(dist)

distDict[cluster1[0]+" " + cluster1[1] + " average"] = sum(sum(dist))/(n\*m)

distDict[cluster1[0] + " "+ cluster1[1] + " centriod"] = np.amax(cdist(pd.DataFrame(clusters[cluster1[0]].mean()).transpose(),pd.DataFrame(clusters[cluster1[1]].mean()).transpose(),distanceMeasure))

**return** distDict

In [17]:

estimatorsEuclidean = {'k\_means\_euclidean\_2': KMeans(n\_clusters=2).fit(featureSelectedX),

'k\_means\_euclidean\_3': KMeans(n\_clusters=3).fit(featureSelectedX),

'k\_means\_euclidean\_4': KMeans(n\_clusters=4).fit(featureSelectedX)}

**def** new\_euclidean\_distances(X, Y=**None**, Y\_norm\_squared=**None**, squared=**False**):

**return** cosine\_similarity(X,Y)

*# monkey patch (ensure cosine dist function is used)*

KMeans.euclidean\_distances = new\_euclidean\_distances

estimatorsDot = {'k\_means\_dot\_2': KMeans(n\_clusters=2).fit(featureSelectedX),

'k\_means\_dot\_3': KMeans(n\_clusters=3).fit(featureSelectedX),

'k\_means\_dot\_4': KMeans(n\_clusters=4).fit(featureSelectedX)}

In [18]:

**for** cluster **in** estimatorsEuclidean:

k = len(np.unique(estimatorsEuclidean[cluster].labels\_))

groups = getClusterGroups(estimatorsEuclidean[cluster].labels\_,featureSelectedX,len(np.unique(estimatorsEuclidean[cluster].labels\_)))

dist = getDistances(groups,'euclidean')

pd.DataFrame([dist]).transpose().to\_csv('/Users/Daniel/Documents/Data Mining 514/HW2/Problem 2/results/ResultsEuclideanK(**{0}**).csv'.format(k))

**for** cluster **in** estimatorsDot:

k = len(np.unique(estimatorsDot[cluster].labels\_))

groups = getClusterGroups(estimatorsDot[cluster].labels\_,featureSelectedX,k)

dist = getDistances(groups,'cosine')

pd.DataFrame([dist]).transpose().to\_csv('/Users/Daniel/Documents/Data Mining 514/HW2/Problem 2/results/ResultsDotK(**{0}**).csv'.format(k))

Problem 3:

**import** **sklearn**

**from** **sklearn.ensemble** **import** RandomForestClassifier

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**from** **sklearn.cluster** **import** KMeans

**import** **random**

**from** **sklearn.preprocessing** **import** Imputer

**from** **sklearn.preprocessing** **import** PolynomialFeatures

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**from** **sklearn.svm** **import** SVC

**from** **sklearn.feature\_selection** **import** SelectFromModel

In [3]:

df = pd.read\_csv('/Users/Daniel/Documents/Data Mining 514/HW1/problem5/allPatients.csv',low\_memory=**False**)

X = df.drop('Classes',axis=1)

features = X.columns

X.replace(to\_replace='?', value = 'NaN', inplace=**True**)

imp = Imputer(missing\_values='NaN', strategy='mean', axis=0)

X=imp.fit\_transform(X)

Y = df['Classes']

X = pd.DataFrame(X,columns=features)

In [17]:

*#fit linear svm determine*

clf = SVC(kernel ='linear')

clf.fit(X,Y)

clf.score(X,Y)

bestFeatures = abs(clf.coef\_[0]).argsort()[::-1][:200]

topFeatures = pd.DataFrame(X.columns).ix[bestFeatures]

topFeatures.to\_csv('/Users/Daniel/Documents/Data Mining 514/HW2/top200Featureslinear.csv')

*# get top 100 feature*

In [7]:

**def** get\_feature\_names(self, input\_features=**None**):

*"""*

*Return feature names for output features*

*Parameters*

*----------*

*input\_features : list of string, length n\_features, optional*

*String names for input features if available. By default,*

*"x0", "x1", ... "xn\_features" is used.*

*Returns*

*-------*

*output\_feature\_names : list of string, length n\_output\_features*

*"""*

powers = self.powers\_

**if** input\_features **is** **None**:

input\_features = ['x**%d**' % i **for** i **in** range(powers.shape[1])]

feature\_names = []

**for** row **in** powers:

inds = np.where(row)[0]

**if** len(inds):

name = " ".join("**%s**^**%d**" % (input\_features[ind], exp)

**if** exp != 1 **else** input\_features[ind]

**for** ind, exp **in** zip(inds, row[inds]))

**else**:

name = "1"

feature\_names.append(name)

**return** feature\_names

top100 = pd.read\_csv('/Users/Daniel/Documents/Data Mining 514/HW2/top100Features.csv')

featureSelectedX = X.ix[:,top100.ix[:,1]]

poly = PolynomialFeatures(2)

Xpoly = poly.fit\_transform(featureSelectedX)

names = get\_feature\_names(poly,featureSelectedX.columns)

clf = SVC(kernel ='linear')

clf.fit(Xpoly,Y)

clf.coef\_[0]

bestFeatures = abs(clf.coef\_[0]).argsort()[::-1][:200]

topFeatures = pd.DataFrame(names).ix[bestFeatures]

In [15]:

topFeatures.to\_csv('/Users/Daniel/Documents/Data Mining 514/HW2/top200FeaturesPolynomial.csv')