**Materials Informatics – Fall 2017**

**Computer Project 3 – Solutions**

**Due on: Nov 7 2017 11:59pm**

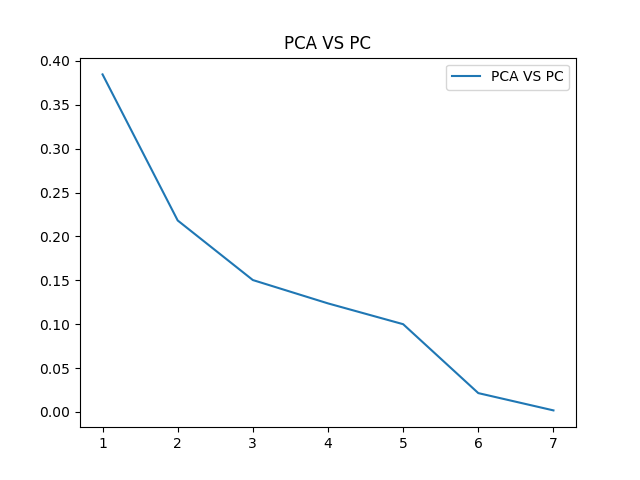
Jianfeng Song

UIN:426009910

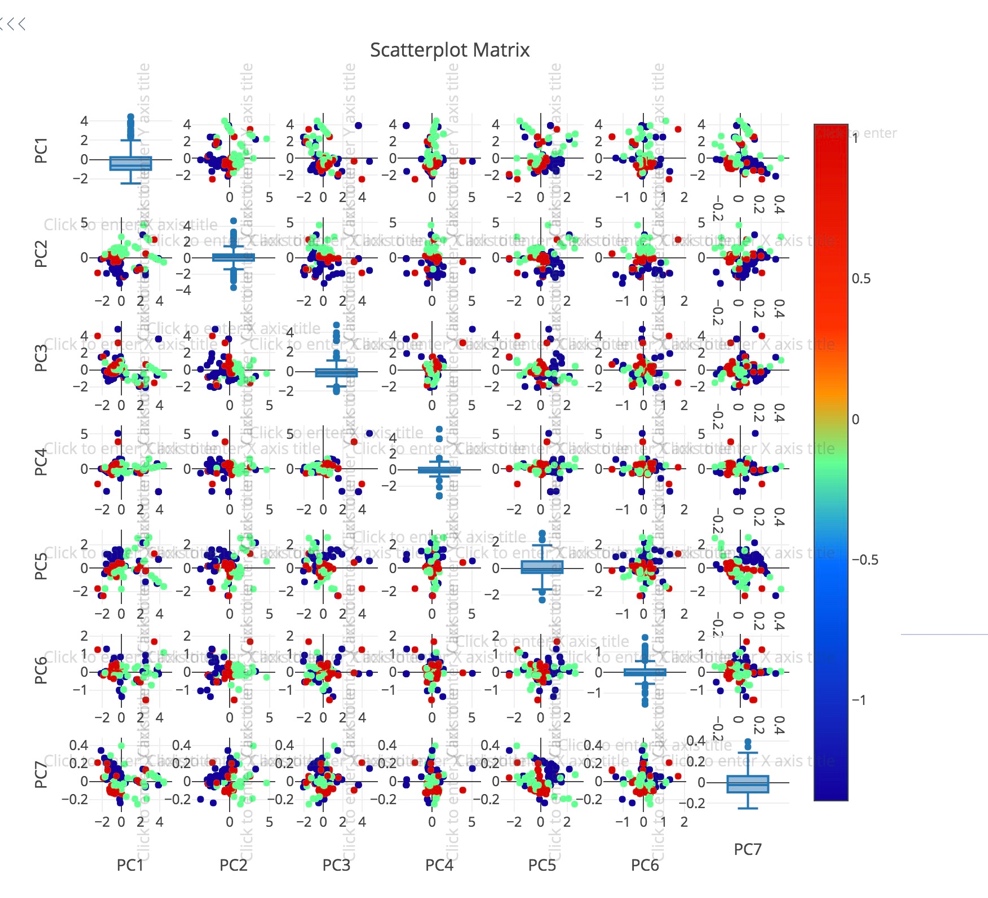
[Jsong26@tamu.edu](mailto:Jsong26@tamu.edu)

Assignment 1

b.



c.



From the plot, we can find that by selecting PC1 and PC2 we can observe the best classification, where PC1 and PC2 are the first two largest PC. When we use PC6 and PC 7 to form our classifier, the group of data are very messy. If I have to project the data down to one PC, I will choose PC2.

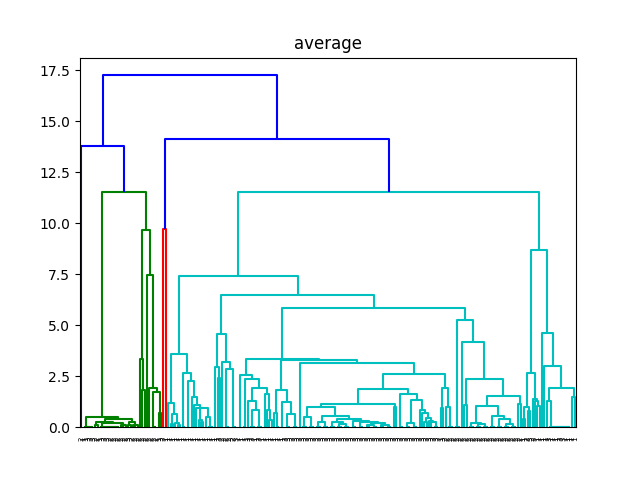
d.

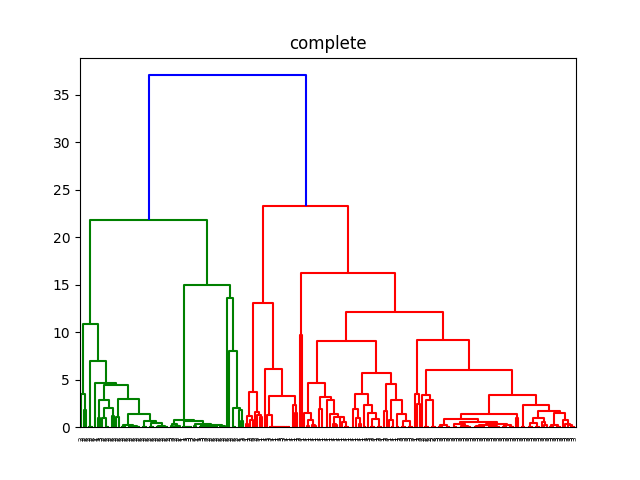
|  |
| --- |
| Fe |
| Ni |
| C |
| Si |
| Cr |
| N |
| Mn |

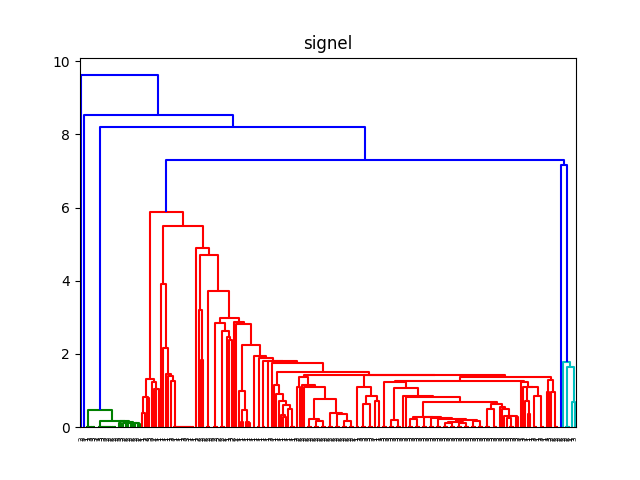
For this part, Fe contribute the most to the discriminating PC, and Ni is the second, which is the same as in project 1 and 2. So we get the correct result by using unsupervised learning.

Assignment 2





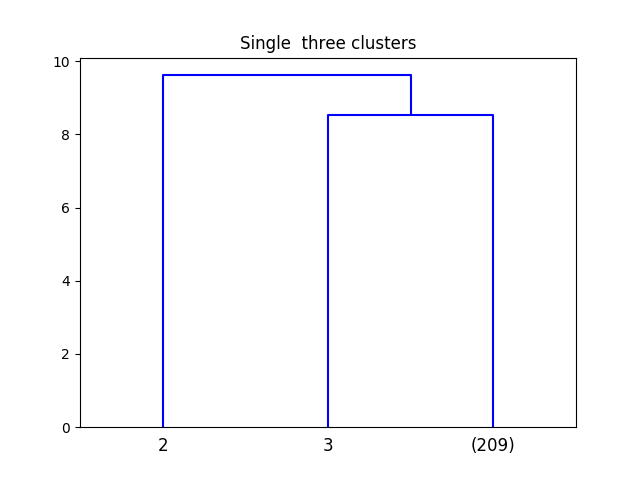


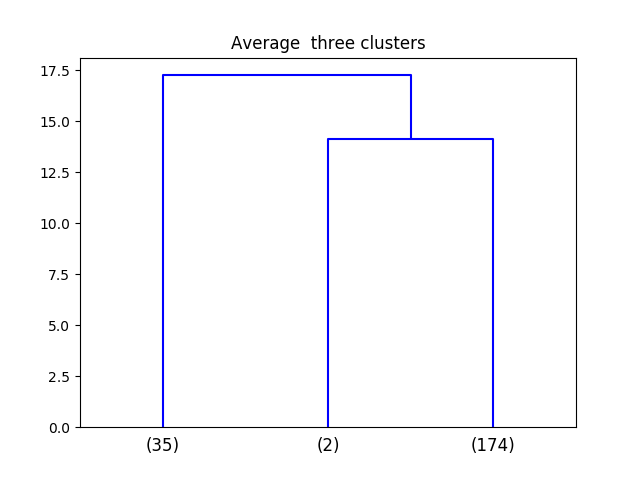


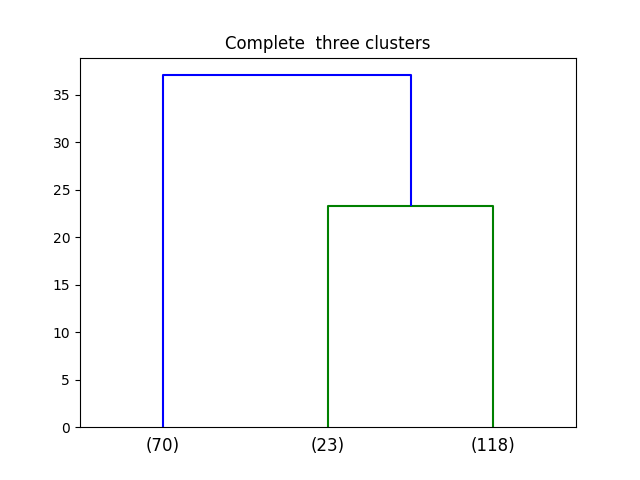
In this part, for single linkage is suffering from chaining effects, so it is sensitive to outliers

For complete and average linkage, they are not sensitive to outliers, and so they tend to break large cluster, they are better than single linkage









By three plots, I observed the single linkage and average linkage are suffering from overfitting, so even they have three group, but at least one group are having small amount of data, so they are not good classifiers. The complete linkage also has three group, and the classification error is relatively small, so complete linkage is the best of three.

Code：

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Thu Nov 2 20:27:26 2017

@author: jianfengsong

"""

import xlrd as xl

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA

from sklearn.decomposition import PCA

from itertools import combinations

import sklearn as sklearn

import matplotlib.pyplot as plt

import plotly.plotly as py

import plotly.figure\_factory as ff

import pandas as pd

import plotly

import operator

from scipy.cluster.hierarchy import cut\_tree

from scipy.cluster.hierarchy import single

from scipy.cluster.hierarchy import average

from scipy.cluster.hierarchy import complete

from scipy.cluster.hierarchy import dendrogram

class fun():

def excel\_data(n):

train\_rows\_value=list()

train\_cols\_value=list()

excel=xl.open\_workbook(n)

data\_table=excel.sheet\_by\_index(0)

rows=data\_table.nrows

cols=data\_table.ncols

for a in range(rows):

train\_rows\_value.append(data\_table.row\_values(a))

train\_row=np.asarray(train\_rows\_value)

data\_set=[[] for x in range(len(train\_row)-1)]

for a in range (1,len(train\_row),1):

for b in range(0,len(train\_row[a])):

data\_set[a-1].append(float(train\_row[a][b]))

return np.asarray(data\_set)

#########################################################################

def get\_data():

data\_set=fun.excel\_data('SFE\_Dataset.xlsx')

data=data\_set.T

data\_1,data\_2=list(),list()

for a in range(len(data)):

num\_0=0

for b in range(len(data[a])):

if data[a][b]==0:

num\_0+=1

else:

num\_0=num\_0

pre\_0=num\_0/len(data[a])

if pre\_0<=0.4:

data\_1.append(data[a])

data\_1=np.asarray(data\_1).T

for a in range(len(data\_1)):

num\_0=0

for b in range(len(data\_1[a])):

if data\_1[a][b]==0:

num\_0+=1

if num\_0==0:

data\_2.append(data\_1[a])

data\_good=np.asarray(data\_2)

return data\_good

##############################################################################

def data\_clas():

data\_set=fun.get\_data()

data\_no\_sfe=list()

data\_label=list()

for a in range(len(data\_set)):

b=len(data\_set[a])-1

if data\_set[a][b]<35:

data\_label.append(1)

elif data\_set[a][b]>45:

data\_label.append(2)

else:

data\_label.append(3)

data\_set1=data\_set.T

for a in range(0,len(data\_set1)-1):

data\_no\_sfe.append(data\_set1[a])

data\_nosfe=np.asarray(data\_no\_sfe)

return data\_nosfe,data\_label

#y=fun.get\_data()

#x,z=fun.data\_clas()

#h=x.T

#g=sklearn.preprocessing.scale(h, axis=0, with\_mean=True, with\_std=True, copy=True)

#pca = PCA(n\_components=7)

#pca.fit(g)

###############################################################################

######### PCA #################

sample\_set,sample\_label=fun.data\_clas()

sample\_pca=sample\_set.T

pca\_data=sklearn.preprocessing.scale(sample\_pca, axis=0, with\_mean=True, with\_std=True, copy=True)

pca\_label=sklearn.preprocessing.scale(sample\_label, axis=0, with\_mean=True, with\_std=True, copy=True)

pca = PCA(n\_components=7)

pca\_fit\_data=pca.fit(pca\_data)

#plt.figure(1)

#plt.title('PCA VS PC')

#plt.plot(range(1,8),pca.explained\_variance\_ratio\_,label='PCA VS PC')

#plt.legend()

#plt.show

########## C ##############################################################

#plotly.tools.set\_credentials\_file(username='jsong26', api\_key='0PJZaMHBnugUbyATBYXl')

#trans\_pca=pca.fit\_transform(pca\_data)

#dataframe = pd.DataFrame(trans\_pca,columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7'])

#

#dataframe['PCA'] = pd.Series(pca\_label)

#

#

#fig = ff.create\_scatterplotmatrix(dataframe, diag='box', index='PCA',

# height=800, width=800)

#py.iplot(fig, filename='Box plots along Diagonal Subplots')

################### D ######################################################

w=pca\_fit\_data.components\_

index\_set=list()

for i in range(len(w)):

index, value = max(enumerate(w[i]), key=operator.itemgetter(1))

index\_set.append(index)

####################### Assignment 2 a #####################################

single\_m=single(sample\_pca)

plt.figure(2)

plt.title('signel')

dendrogram(single\_m,labels=sample\_label)

average\_m=average(sample\_pca)

plt.figure(3)

plt.title('average')

dendrogram(average\_m,labels=sample\_label)

complete\_m=complete(sample\_pca)

plt.figure(4)

plt.title('complete')

dendrogram(complete\_m,labels=sample\_label)

########################################## b ###############################

single\_c=cut\_tree(single\_m,n\_clusters=3)

average\_c=cut\_tree(average\_m,n\_clusters=3)

complete\_c=cut\_tree(complete\_m,n\_clusters=3)

plt.figure(5)

plt.title('Single three clusters')

Comp\_den = dendrogram(single\_m, p = 3, truncate\_mode = 'lastp', labels = sample\_label)

plt.figure(6)

plt.title('Average three clusters')

Comp\_den = dendrogram(average\_m, p = 3, truncate\_mode = 'lastp', labels = sample\_label)

plt.figure(7)

plt.title('Complete three clusters')

Comp\_den = dendrogram(complete\_m, p = 3, truncate\_mode = 'lastp', labels = sample\_label)