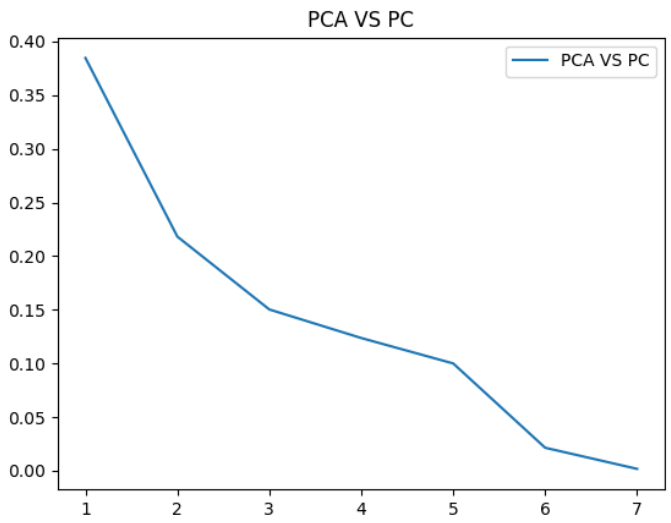
Materials Informatics – Fall 2017
Computer Project 3 – Solutions
Due on: Nov 7 2017 11:59pm

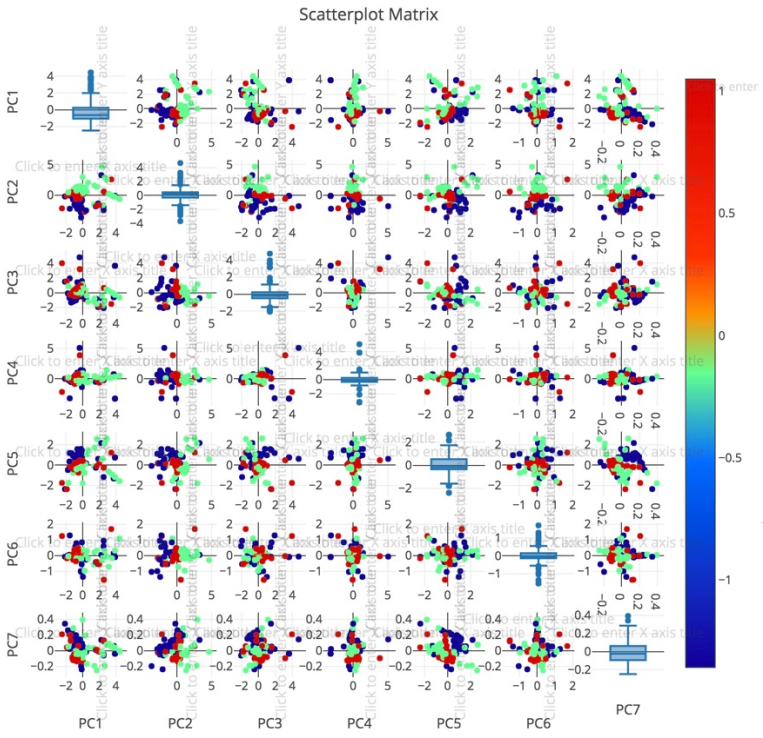
Jianfeng Song
UIN:426009910
Jsong26@tamu.edu

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.

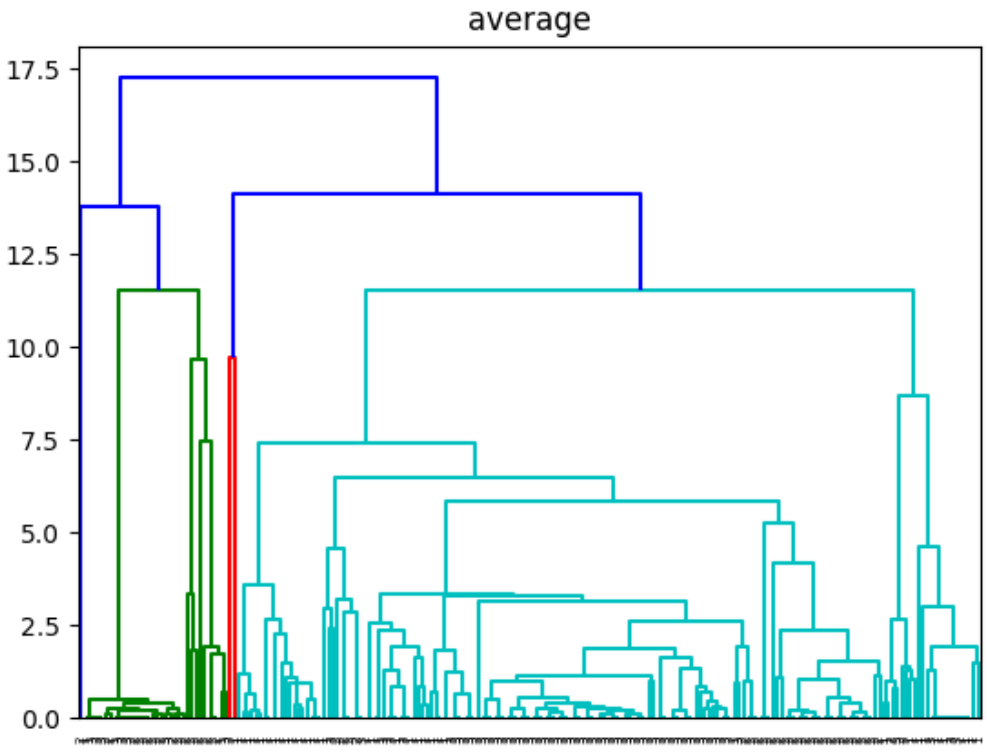
d.

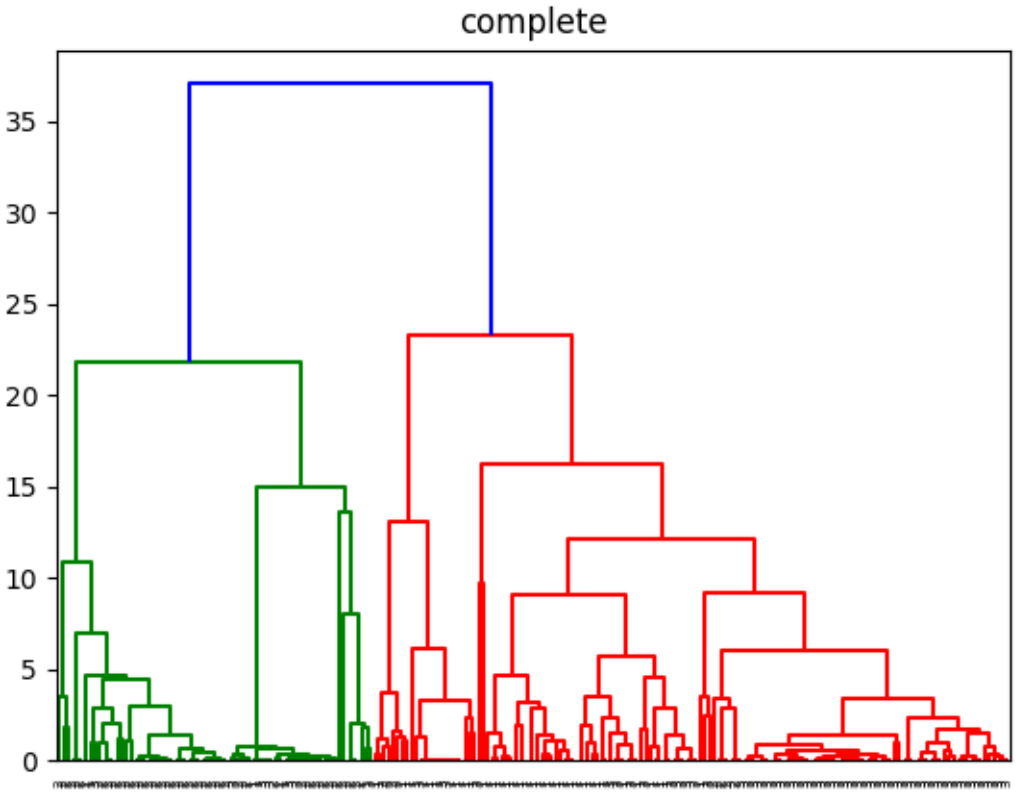
Fe
Ni
C
Si
Cr
N
Mn

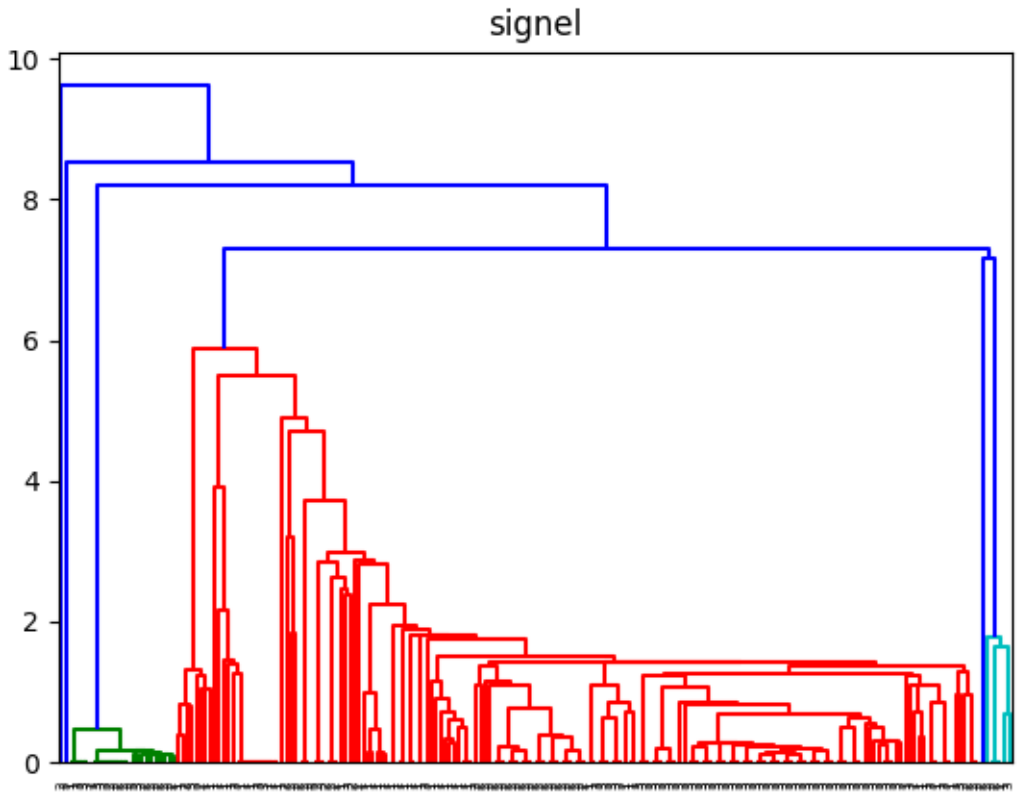
For this part, Fe contribute the most to the discriminating PC, and Ni is the second, which is the same as in project 1 and 2. So we get the correct result by using unsupervised learning.

Assignment 2

a.



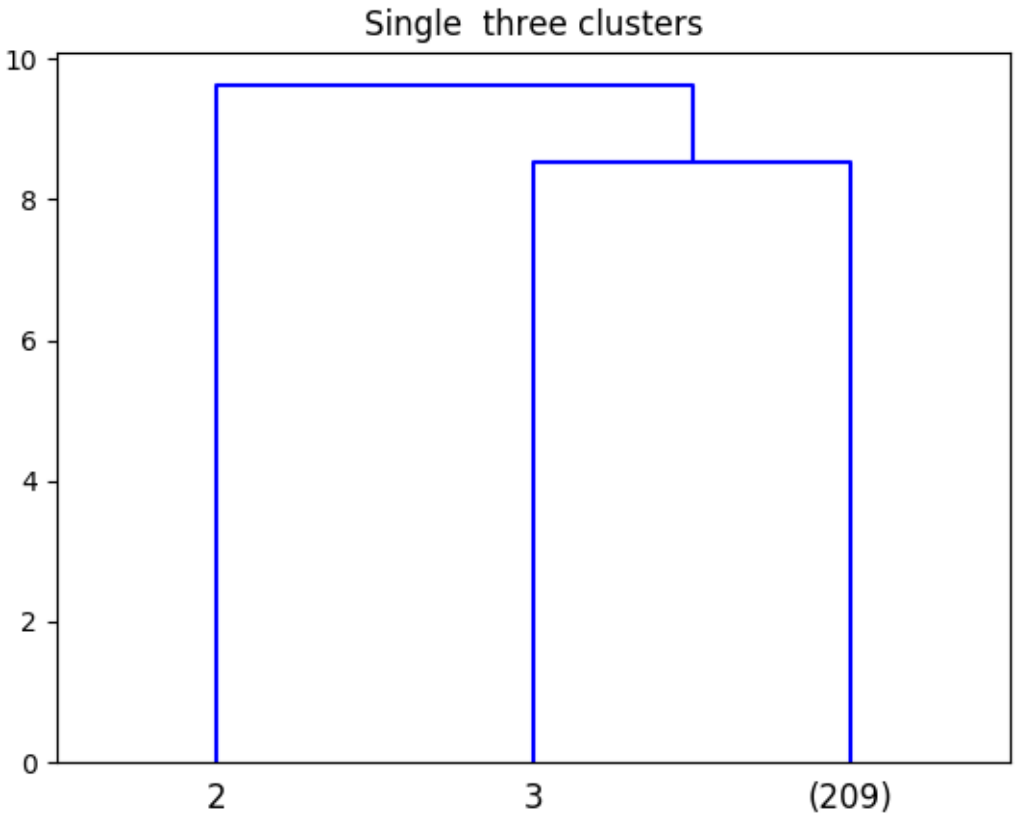

complete
35
30
25
20
15
10
5
0

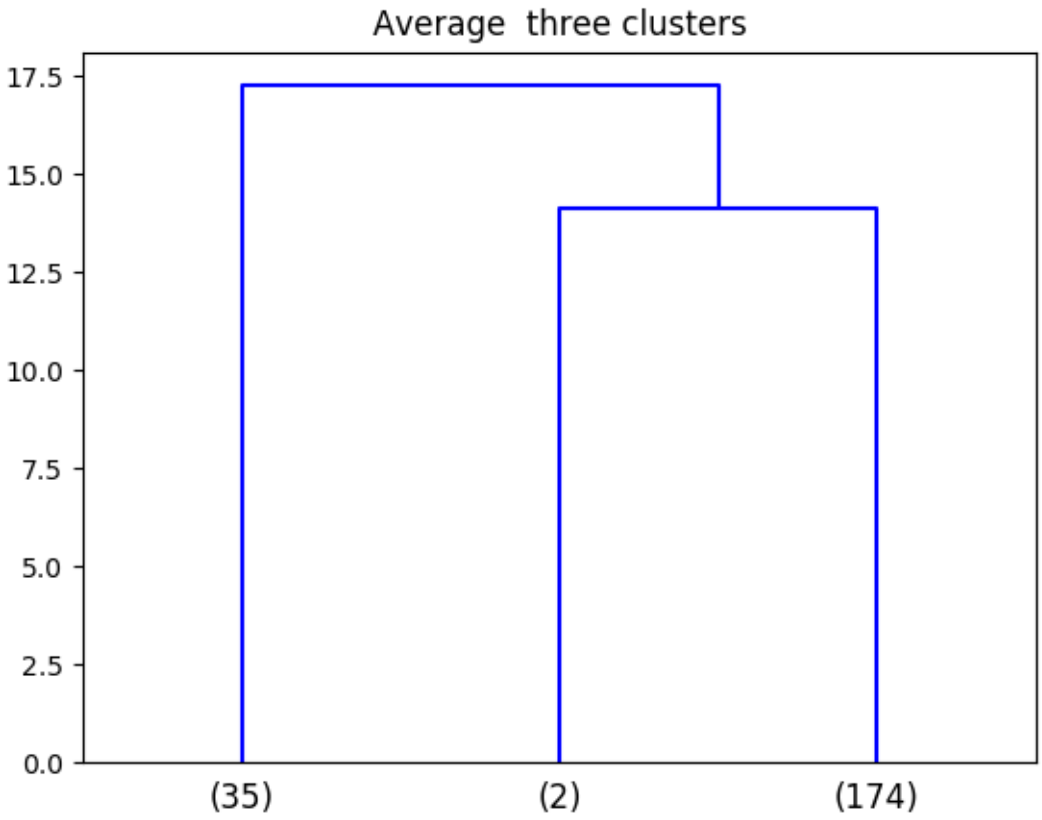


In this part, for single linkage is suffering from chaining effects, so it is sensitive to outliers

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

b.


Single three clusters
10
8
6
4
2
0
2
3
(209)


Average three clusters
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
(35)
(2)
(174)

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)
```