Materials Informatics – Fall 2017
Computer Project 1 – Solutions
Due on: Oct 10 2017 11:59pm

Jianfeng Song
Jsong26@tamu.edu

Assignment 1:

a.

$$\varepsilon = \Phi\left(\frac{-1}{2}\delta\right)$$

$$\delta^2 = (u_1 - u_0)^T \Sigma^{-1} (u_1 - u_0)$$

$$u_1 = (1,1) \qquad u_0 = (0,0)$$

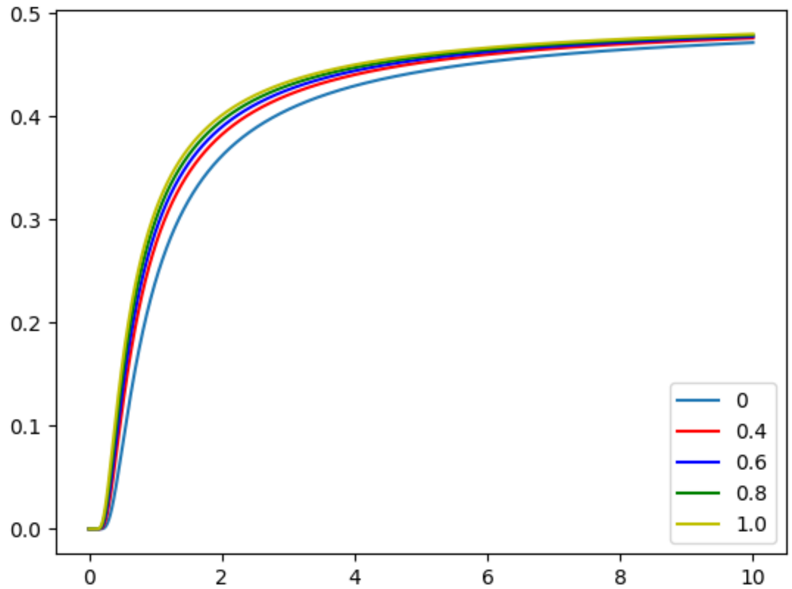
$$\Sigma = \sigma^2 \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \qquad \rho = 0.2$$

$$\Sigma^{-1} = \begin{pmatrix} \sigma^2 & -\rho\sigma^2 \\ -\rho\sigma^2 & \sigma^2 \end{pmatrix} \frac{1}{\sigma^4 - \sigma^4\rho^2}$$

$$\delta^2 = \frac{2(1-\rho)}{\sigma^2(1+\rho)(1-\rho)} = \frac{1 \cdot 2}{\sigma^2(1+\rho)}$$

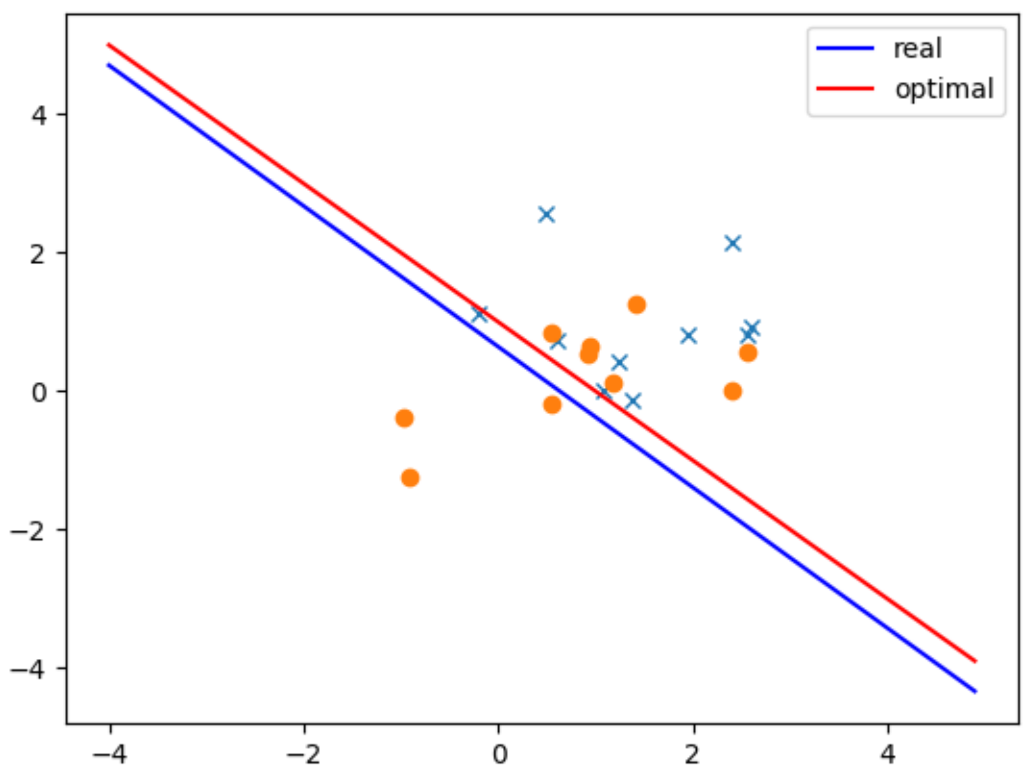
$$\varepsilon = \Phi\left(\frac{-1}{\sqrt{2}\,\sigma\sqrt{1+\rho}}\right)$$

b.




When we fixed p, as we increase variance the optimal error is also increased, when we fixed variance, the optimal error will increase as we increase p.

c.




LDA uses the Gaussian assumption to estimate the optimal classifier, using the sample means and sample covariance matrices.

$$\hat{\mu}_0 = \frac{1}{n_0}\sum_{i=1}^{n} X_i I_{Y_i=0} \qquad \hat{\mu}_1 = \frac{1}{n_1}\sum_{i=1}^{n} X_i I_{Y_i=1}$$

$$\hat{\Sigma} = \frac{1}{n-2}\sum_{i=1}^{n}\left[(X_i-\hat{\mu}_0)(X_i-\hat{\mu}_0)^T I_{Y_i=0} + (X_i-\hat{\mu}_1)(X_i-\hat{\mu}_1)^T I_{Y_i=1}\right]$$

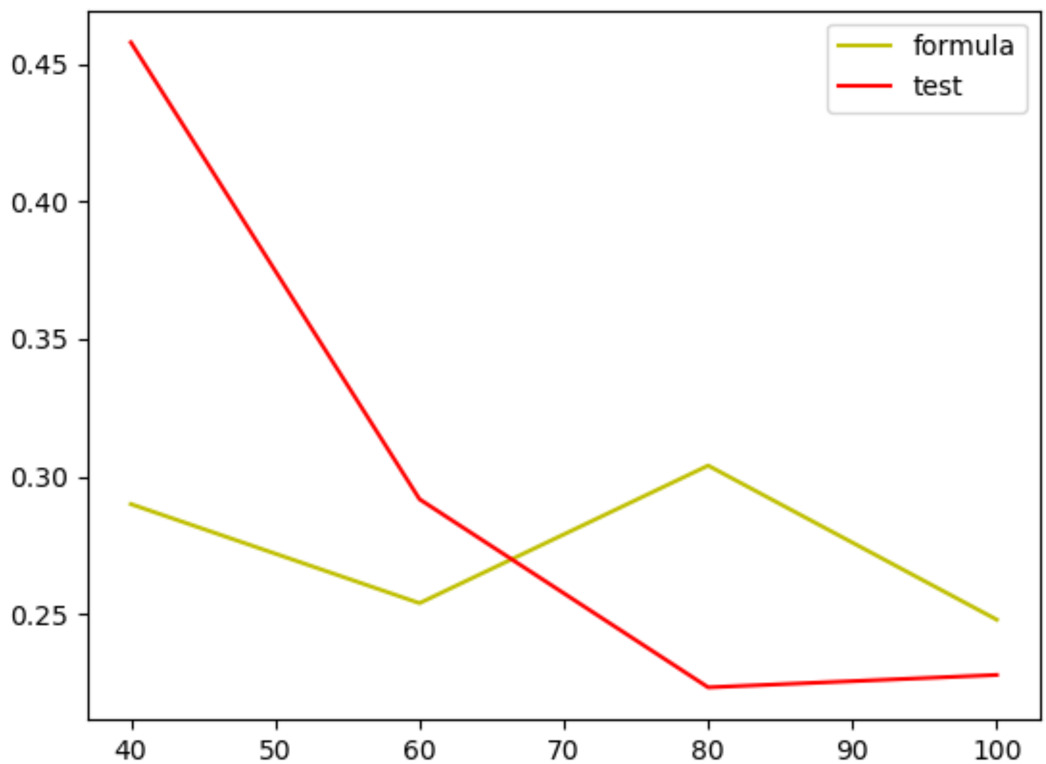
$$a_n = \hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_0)$$

$$b_n = -\frac{1}{2}(\hat{\mu}_1 - \hat{\mu}_0)^T \hat{\Sigma}^{-1}(\hat{\mu}_1 + \hat{\mu}_0)$$

$$g_n(x) = a_n^T x + b_n = 0$$

By develop the optimal classifier, we find that the optimal classifier has better describe about the data. Our designed LDA classifier should be better as we increase the size of our data, so the mean and covariance will be close to optimal classifier.

d.




As we increase size of sample data, my test set error and formula error both became smaller. When the size of sample data is small, the test set error is bigger than formula error. In the figure above, when size of sample data is more than 65,then the test set error is better than formula error.

Assignment 2:

a. This is my training data after pre-processing

```
[[  6.00000000e-02   7.46000000e+00   6.47610000e+01   8.96000000e+00
    1.82000000e+01   3.30000000e+01]
 [  6.00000000e-02   1.33000000e+01   6.72400000e+01   4.00000000e-01
    1.73000000e+01   6.80000000e+01]
 [  1.11800000e+01   7.06500000e+01   1.00000000e-02   1.00000000e-02
    1.81300000e+01   1.93000000e+01]
 ...,
 [  1.40000000e+01   6.96700000e+01   2.00000000e-02   8.00000000e-02
    1.62000000e+01   2.30000000e+01]
 [  2.60000000e-02   1.12000000e+01   6.82300000e+01   1.18000000e+00
    1.89000000e+01   3.47000000e+01]
 [  7.00000000e-02   1.61300000e+01   5.48180000e+01   9.64000000e+00
    1.84800000e+01   6.50000000e+01]]
```

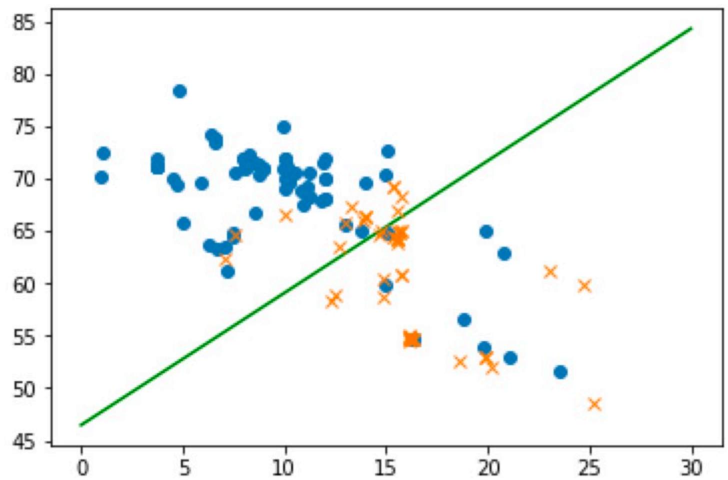
This is my test data after pre-processing

```
[[  7.00000000e-03   2.30000000e+01   6.11270000e+01   4.00000000e-02
    1.58000000e+01   6.44000000e+01]
 [  4.00000000e-03   1.56000000e+01   6.43170000e+01   3.00000000e-02
    1.75000000e+01   5.06000000e+01]
 [  2.00000000e-02   7.56000000e+00   7.05000000e+01   7.82000000e+00
    7.88000000e+00   6.39000000e+00]
 ...,
 [  3.00000000e-02   1.63000000e+01   5.46980000e+01   9.64000000e+00
    1.84800000e+01   7.30000000e+01]
 [  2.60000000e-02   1.12000000e+01   6.82300000e+01   1.18000000e+00
    1.89000000e+01   3.45000000e+01]
 [  3.00000000e-02   5.90000000e+00   6.96500000e+01   7.10000000e+00
    1.62000000e+01   1.75000000e+01]]
```

b. A

{'Ni': Ttest_indResult(statistic=2.1832777249025574, pvalue=0.044256486360899815),
'Fe': Ttest_indResult(statistic=1.4593669470171782, pvalue=0.16371947301978981),
'C': Ttest_indResult(statistic=-1.4370077964097883, pvalue=0.16872848917102379),
'Mn': Ttest_indResult(statistic=-0.555911128003719, pvalue=0.58332167105335986),
'Cr': Ttest_indResult(statistic=-0.36725489505964665, pvalue=0.7164182125651759)}

c. Classification error with top 2: [0.07575757575757576]




d. A
Classification error with top 3:[0.06944444444444445]
Classification error with top 4:[0.06818181818181818]
Classification error with top 5:[0.06781414141414141]
As we increase number of predictors, the classification error will decrease, because we will have a better classifier.

Code:
Assignment 1
b.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Oct  6 21:32:11 2017

@author: jianfengsong
"""
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
import scipy.stats as ns
import math

infn=np.inf

def var(x1,y1):
    h=-np.sqrt(2)*x1*np.sqrt(1+y1)
    return 1/h
#    return -math.pow(h,-1)

#def snrv(x):
```

```
#    return [(2*np.pi)**0.5]*{integrate.quad(np.exp(-(x**2))/2,lambda x :-infn,lambda
x:x}

x=np.arange(0.001,10.0,0.001)
y=np.arange(0.4,1.0,0.2)

plt.figure(1)
#for a in y:
#    for b in x:
plt.plot(x,ns.norm.cdf(var(x,0)),label='0')
plt.plot(x,ns.norm.cdf(var(x,0.4)),'r',label='0.4')
plt.plot(x,ns.norm.cdf(var(x,0.6)),'b',label='0.6')
plt.plot(x,ns.norm.cdf(var(x,0.8)),'g',label='0.8')
plt.plot(x,ns.norm.cdf(var(x,1.0)),'y',label='1.0')
#      print(x,var(x,a))
plt.legend()
plt.show()

#print (snrv(1))
```

c.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Oct  6 21:32:11 2017

@author: jianfengsong
"""
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
import scipy.stats as ns
import math
import sympy as sym
#import scipy as sym

x=sym.Symbol('x')
p=np.array([[1,0.2],[0.2,1]])
pt=p**-1
u1=np.array([[1,1]])
u1t=np.matrix.transpose(u1)
u0=np.array([[0,0]])
u0t=np.matrix.transpose(u0)
x1ur=0
x2ur=0
y1ur=0
```

```
y2ur=0
plt.figure(1)
x1l=list()
x2l=list()
sample_size=10
x1= np.random.multivariate_normal([1,1], p,sample_size)
x2= np.random.multivariate_normal([0,0], p,sample_size)

an=np.dot(pt,(u1t-u0t))
bn=(-1/2)*np.dot(np.dot([np.matrix.transpose(u1t-u0t)],p**-1),(u1t+u0t))
plt.plot(x1[:,0],x1[:,1],'x')
plt.plot(x2[:,0],x2[:,1],'o')

sumx=0
sumy=0
for a in x1:
   sumx=a+sumx
sum1=sumx/sample_size #sum1 is mean of u1(1,1)
for a in x2:
   sumy=a+sumy
sum0=sumy/sample_size#sum0 is mean of u0(0,0)
pn1=(np.dot((x1-sum0).T,(x1-sum0))+np.dot((x2-sum1).T,(x2-
sum1)))/(2*sample_size-2)

an1=np.dot(pn1**-1,(sum1-sum0))
bn1=(-1/2)*np.dot(np.dot((sum1-sum0).T,pn1**-1),(sum1-sum0))
print(an1)
print(bn1)
x1=np.arange(-4,5,0.1)
x2p=-bn1/an1[1]-an1[0]*x1/an1[1]
plt.plot(x1,x2p,'b',label='real')
x12=np.arange(-4,5,0.1)
bn2=(-1/2)*np.dot(np.dot((u1t-u0t).T,p**-1),(u1t-u0t))
x22p=-bn2/an[1]-an[0]*x12/an[1]
plt.plot(x12,x22p.T,'r',label='optimal')
plt.legend()
plt.show()
#optimal
#pn2=(np.dot((x1-u0.T).T,(x1-u0.T))+np.dot((x2-u1.T).T,(x2-
u1.T)))/(2*sample_size-2)
#an2=np.dot(pn2**-1,(u1-u0))
#bn2=(-1/2)*np.dot(np.dot((u1-u0).T,pn2**-1),(u1-u0))
##print(an1)
##print(bn1)
#x12=np.arange(-4,5,0.1)
#x22p=-bn2/an2[1]-an2[0]*x1/an2[1]
```

```
#plt.plot(x12,x22p,'r')
```
d.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Oct  8 15:56:58 2017

@author: jianfengsong
"""
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as ns
import math
from scipy.linalg import det
p=np.array([[1,0.2],[0.2,1]])
u0=np.array([[0,0]])
u1=np.array([[1,1]])
sample_size=np.array([20,30,40,50])
LDA_error_set=list()
error_set=list()

for a in sample_size:
    x1= np.random.multivariate_normal([1,1], p,a)
    x2= np.random.multivariate_normal([0,0], p,a)
    x3= np.random.multivariate_normal([1,1],p,250)
    x4= np.random.multivariate_normal([0,0],p,250)
    sumx1=0
    sumx2=0
    for b in x1:
        sumx1=b+sumx1
    mean_x1=sumx1/a
    for c in x2:
        sumx2=c+sumx2
    mean_x0=sumx2/a

    cov=(np.dot((x1-mean_x0).T,(x1-mean_x0))+np.dot((x2-mean_x1).T,(x2-
mean_x1)))/(2*a-2)
    an=np.dot(cov**-1,(mean_x1-mean_x0))
    bn=(-1/2)*np.dot(np.dot((mean_x1-mean_x0).T,cov**(-1)),(mean_x1-mean_x0))
    var_x0=(np.dot(an,mean_x0.T)+bn)/math.sqrt(np.dot(np.dot(an,p),an.T))
    var_x1=(np.dot(an,mean_x1.T)+bn)/math.sqrt(np.dot(np.dot(an,p),an.T))
    LDA_error=1/2*(ns.norm.cdf(var_x0)+ns.norm.cdf(-var_x1))
    LDA_error_set.append(LDA_error)
    clas_x3_y=-bn/an[1]-an[0]*x3[:,0]/an[1]
    clas_x4_y=-bn/an[1]-an[0]*x4[:,0]/an[1]
    error_time=0
```

```
    for t in range(250):
        if x3[t,1] < clas_x3_y[t]:
            error_time=error_time+1
        if x4[t,1] > clas_x4_y[t]:
            error_time=error_time+1
    error_set.append(error_time/(500))

plt.plot(sample_size*2,error_set,'y',label='formula')
plt.plot(sample_size*2,LDA_error_set,'r',label='test')
plt.legend()
plt.show()
```

Assignment 2

a. A

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Oct  8 20:04:49 2017

@author: jianfengsong
"""
import xlrd as xl
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
import scipy.stats as ns
import math
import random
import xlwt as xlw
rows_value=list()
cols_value=list()
excel=xl.open_workbook('SFE_Dataset.xlsx')
data_table=excel.sheet_by_index(0)
rows=data_table.nrows
cols=data_table.ncols
for a in range(1,rows,1):
    if data_table.row_values(a,cols-1)[0]>=45 or data_table.row_values(a,cols-
1)[0]<=35 :
        rows_value.append(data_table.row_values(a))


#for a in range(1,cols,1):
#    if data_table.col_values(cols-1):
#        cols_value.append(data_table.col_values(a))
#for a in range(rows):
ele_0=list()
```

```
num_0=0
for a in range(0,cols,1):
    for b in range(0,len(rows_value),1):
        if rows_value[b][a]<=0.00000001:
            num_0=num_0+1
    if num_0/len(rows_value)>=0.4:
#        print(len(rows_value))
        num_0=0
        ele_0.append(a)
for row in rows_value:
    num_del=0
    for col in ele_0:
        col=col-num_del
        row.remove(row[col])
        num_del=num_del+1
num_del_row=list()
num_0_row=0
#delete row value that are 0
for row in rows_value:
#    print (row)
    for a in row:
#        print(a)
        if a==0:
            num_0_row=num_0_row+1
    if num_0_row>0:
        num_0_row=0
        num_del_row.append(row)
for a in range(0,len(num_del_row),1):
    num=0
    a=a-num
    rows_value.remove(num_del_row[a])
    num=num+1

#random choose value for test set

numtrain=list()
numtest=list()
test_set=list()
train_set=list()
status=True
go=0
#num=0
while (status):
    num=0
    go=go+1
    print(go)
```

```
    random_set=random.sample(range(len(rows_value)),len(rows_value))
    for a in range(int(len(rows_value)*0.2)):
        numtrain.append(random_set[a])
    for a in range(len(numtrain)):
        train_set.append(rows_value[numtrain[a]])
    for a in range(len(rows_value)-int(len(rows_value)*0.2)):
        numtest.append(random_set[a+int(len(rows_value)*0.2)])
    for a in range(len(numtest)):
        test_set.append(rows_value[numtest[a]])
    for row in train_set:
        if row[len(row)-1]<=35:
            num=num+1
    if num/len(train_set)>=0.55 or num/len(train_set)<=0.45:
        numtrain=list()
        numtest=list()
        test_set=list()
        train_set=list()
        print("not yet")
#    if num/len(train_set)<0.55 and num/len(train_set)<0.45:
    else:
        status=False
        print("you are good to go")

data_final=xlw.Workbook()
sheet1=data_final.add_sheet('sheet1',cell_overwrite_ok=True)
for i in range(len(train_set)):
    for j in range(len(train_set[i])):
        sheet1.write(i,j,train_set[i][j])
data_final.save('data_final.xls')
test_set1=np.asarray(test_set)
train_set0=np.asarray(train_set)
print(test_set1)
print(train_set0)
```

b. A

```
import xlrd as xl
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
import scipy.stats as ns
import math
import random
rows_value=list()
cols_value=list()
excel=xl.open_workbook('SFE_Dataset.xlsx')
data_table=excel.sheet_by_index(0)
```

```
rows=data_table.nrows
cols=data_table.ncols
for a in range(1,rows,1):
    if data_table.row_values(a,cols-1)[0]>=45 or data_table.row_values(a,cols-
1)[0]<=35 :
        rows_value.append(data_table.row_values(a))
ele_0=list()
num_0=0
for a in range(0,cols,1):
    for b in range(0,len(rows_value),1):
        if rows_value[b][a]<=0.00000001:
            num_0=num_0+1
    if num_0/len(rows_value)>=0.4:
        num_0=0
        ele_0.append(a)
for row in rows_value:
    num_del=0
    for col in ele_0:
        col=col-num_del
        row.remove(row[col])
        num_del=num_del+1
num_del_row=list()
num_0_row=0
for row in rows_value:
    for a in row:
        if a==0:
            num_0_row=num_0_row+1
    if num_0_row>0:
        num_0_row=0
        num_del_row.append(row)
for a in range(0,len(num_del_row),1):
    num=0
    a=a-num
    rows_value.remove(num_del_row[a])
    num=num+1

#random choose value for test set

numtrain=list()
numtest=list()
test_set=list()
train_set=list()
status=True
go=0
while (status):
    num=0
```

```
    go=go+1
#    print(go)
    random_set=random.sample(range(len(rows_value)),len(rows_value))
    for a in range(int(len(rows_value)*0.2)):
        numtrain.append(random_set[a])
    for a in range(len(numtrain)):
        train_set.append(rows_value[numtrain[a]])
    for a in range(len(rows_value)-int(len(rows_value)*0.2)):
        numtest.append(random_set[a+int(len(rows_value)*0.2)])
    for a in range(len(numtest)):
        test_set.append(rows_value[numtest[a]])
    for row in train_set:
        if row[len(row)-1]<=35:
            num=num+1
    if num/len(train_set)>=0.55 or num/len(train_set)<=0.45:
        numtrain=list()
        numtest=list()
        test_set=list()
        train_set=list()
#        print("not yet")
#    if num/len(train_set)<0.55 and num/len(train_set)<0.45:
    else:
        status=False
#        print("you are good to go")
#save excel doc
data_final=xlw.Workbook()
sheet1=data_final.add_sheet('sheet1',cell_overwrite_ok=True)
for i in range(len(train_set)):
    for j in range(len(train_set[i])):
        sheet1.write(i,j,train_set[i][j])
data_final.save('data_final.xls')

#Assignment2(b)
train_set35=list()
train_set45=list()
length_train=len(train_set)
for a in range(0,length_train,1):
    if train_set[a][len(train_set[a])-1]<=35:
        train_set35.append(a)
    else:
        train_set45.append(a)
train35=list()
train45=list()
for a in train_set35:
    train35.append(train_set[a])
for a in train_set45:
```

```
    train45.append(train_set[a])
#train_35=list()
#train_45=list()
#for a in range(len(train35[1])):
#    for b in range(len(train35)):
#         train_35.append(train35[b][a])
#for a in range(len(train45[1])):
#    for b in range(len(train45)):
#         train_45.append(train45[b][a])
train_35=[[] for i in range(len(train35[1]))]
train_45=[[] for i in range(len(train45[1]))]
Tset=list()
for a in range(len(train35[1])):
    for b in range(len(train35)):
         train_35[a].append(train35[b][a])
for a in range(len(train45[1])):
    for b in range(len(train45)):
         train_45[a].append(train45[b][a])
for a in range(len(train_35)-1):
    h=ns.ttest_ind(train_35[a],train_45[a],equal_var=False)
    Tset.append(h)
Tset_sta=list()

for a in range(len(Tset)):
    Tset_sta.append(abs(Tset[a][0]))
#print  (Tset_sta)
Tset_sta_name={'C':Tset_sta[0],'Ni':Tset_sta[1],'Fe':Tset_sta[2],'Mn':Tset_sta[3],'Cr':
Tset_sta[4]}
Tset_name={'C':Tset[0],'Ni':Tset[1],'Fe':Tset[2],'Mn':Tset[3],'Cr':Tset[4]}
Tset_name0=np.asarray(Tset_name)
print(Tset_name0)
import operator
sorted_tset = sorted(Tset_sta_name.items(), key=operator.itemgetter(1),reverse=True)
#print(sorted_tset)
```

c. A

```
import xlrd as xl
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
import scipy.stats as ns
import math
import random
import operator
rows_value=list()
```

```
cols_value=list()
excel=xl.open_workbook('SFE_Dataset.xlsx')
data_table=excel.sheet_by_index(0)
rows=data_table.nrows
cols=data_table.ncols
for a in range(1,rows,1):
    if data_table.row_values(a,cols-1)[0]>=45 or data_table.row_values(a,cols-
1)[0]<=35 :
        rows_value.append(data_table.row_values(a))
ele_0=list()
num_0=0
for a in range(0,cols,1):
    for b in range(0,len(rows_value),1):
        if rows_value[b][a]<=0.00000001:
            num_0=num_0+1
    if num_0/len(rows_value)>=0.4:
        num_0=0
        ele_0.append(a)
for row in rows_value:
    num_del=0
    for col in ele_0:
        col=col-num_del
        row.remove(row[col])
        num_del=num_del+1
num_del_row=list()
num_0_row=0
for row in rows_value:
    for a in row:
        if a==0:
            num_0_row=num_0_row+1
    if num_0_row>0:
        num_0_row=0
        num_del_row.append(row)
for a in range(0,len(num_del_row),1):
    num=0
    a=a-num
    rows_value.remove(num_del_row[a])
    num=num+1

#random choose value for test set

numtrain=list()
numtest=list()
test_set=list()
train_set=list()
status=True
```

```
go=0
while (status):
    num=0
    go=go+1
#    print(go)
    random_set=random.sample(range(len(rows_value)),len(rows_value))
    for a in range(int(len(rows_value)*0.2)):
        numtrain.append(random_set[a])
    for a in range(len(numtrain)):
        train_set.append(rows_value[numtrain[a]])
    for a in range(len(rows_value)-int(len(rows_value)*0.2)):
        numtest.append(random_set[a+int(len(rows_value)*0.2)])
    for a in range(len(numtest)):
        test_set.append(rows_value[numtest[a]])
    for row in train_set:
        if row[len(row)-1]<=35:
            num=num+1
    if num/len(train_set)>=0.55 or num/len(train_set)<=0.45:
        numtrain=list()
        numtest=list()
        test_set=list()
        train_set=list()
    else:
        status=False
#save excel doc
#data_final=xlw.Workbook()
#sheet1=data_final.add_sheet('sheet1',cell_overwrite_ok=True)
#for i in range(len(train_set)):
#    for j in range(len(train_set[i])):
#        sheet1.write(i,j,train_set[i][j])
#data_final.save('data_final.xls')

#Assignment2(b)
train_set35=list()
train_set45=list()
length_train=len(train_set)
for a in range(0,length_train,1):
    if train_set[a][len(train_set[a])-1]<=35:
        train_set35.append(a)
    else:
        train_set45.append(a)
train35=list()
train45=list()
for a in train_set35:
    train35.append(train_set[a])
for a in train_set45:
```

```
    train45.append(train_set[a])
train_35=[[] for i in range(len(train35[1]))]
train_45=[[] for i in range(len(train45[1]))]
Tset=list()
for a in range(len(train35[1])):
    for b in range(len(train35)):
        train_35[a].append(train35[b][a])
for a in range(len(train45[1])):
    for b in range(len(train45)):
        train_45[a].append(train45[b][a])
for a in range(len(train_35)-1):
    h=ns.ttest_ind(train_35[a],train_45[a],equal_var=False)
    Tset.append(h)
Tset_sta=list()
for a in range(len(Tset)):
    Tset_sta.append(abs(Tset[a][0]))
#print  (Tset_sta)
Tset_sta_name={'C':Tset_sta[0],'Ni':Tset_sta[1],'Fe':Tset_sta[2],'Mn':Tset_sta[3],'Cr':
Tset_sta[4]}
sorted_tset = sorted(Tset_sta_name.items(), key=operator.itemgetter(1),reverse=True)
print(sorted_tset)

#Assignment2(c)
top=list()
#for a in range(len(sorted_tset)):
#    top.append(sorted_tset[a][1])
#x1=[[] for i in range(len(train35))]
#x2=[[] for i in range(len(train45))]
x11=list()
x21=list()
#for a in range(len(x1)):
#for a in range(len(train35)):
first=Tset_sta.index(sorted_tset[0][1])
second=Tset_sta.index(sorted_tset[1][1])
for b in range(len(train35)):
    x11.append([train35[b][1],train35[b][2]])
for b in range(len(train45)):
    x21.append([train45[b][1],train45[b][2]])
###different
#for b in range(len(train35)):
#    x11.append([train35[b][first],train35[b][second]])
#for b in range(len(train45)):
#    x21.append([train45[b][first],train45[b][second]])
sumx1=0
sumx2=[0,0]
x1=np.asarray(x11)
```

```
x2=np.asarray(x21)
for b in x1:
   sumx1=b+sumx1
mean_x1=sumx1/len(x1)
for c in x2:
   sumx2=c+sumx2
mean_x0=sumx2/len(x2)
cov=(np.dot((x1-mean_x0).T,(x1-mean_x0))+np.dot((x2-mean_x1).T,(x2-
mean_x1)))/(min(len(train35),len(train45)-2))
an=np.dot(cov**-1,(mean_x1-mean_x0))
bn=(-1/2)*np.dot(np.dot((mean_x1-mean_x0).T,cov**(-1)),(mean_x1-mean_x0))
plt.figure(1)
plt.plot(x1[:,0],x1[:,1],'x')
plt.plot(x2[:,0],x2[:,1],'o')
x1=np.arange(-10,100,1)
x2p=-bn/an[1]-an[0]*x1/an[1]
plt.plot(x1,x2p.T,'r')

#test set
test_set35=list()
test_set45=list()
length_test=len(test_set)
for a in range(0,length_test,1):
   if test_set[a][len(test_set[a])-1]<=35:
      test_set35.append(a)
   else:
      test_set45.append(a)
test35=list()
test45=list()
for a in test_set35:
   test35.append(test_set[a])
for a in test_set45:
   test45.append(test_set[a])
test_35=[[] for i in range(len(test35[1]))]
test_45=[[] for i in range(len(test45[1]))]
#Tset=list()
x112=list()
x212=list()
for a in range(len(test35[1])):
   for b in range(len(test35)):
       test_35[a].append(test35[b][a])
for a in range(len(test45[1])):
   for b in range(len(test45)):
       test_45[a].append(test45[b][a])
for b in range(len(test35)):
   x112.append([test35[b][1],test35[b][2]])
```

```
for b in range(len(test45)):
   x212.append([test45[b][1],test45[b][2]])
sumx12=0
sumx22=[0,0]
x12=np.asarray(x112)
x22=np.asarray(x212)
for b in x12:
   sumx12=b+sumx12
mean_x12=sumx12/len(x12)
for c in x22:
   sumx22=c+sumx22
mean_x02=sumx22/len(x22)
cov2=(np.dot((x12-mean_x02).T,(x12-mean_x02))+np.dot((x22-mean_x12).T,(x22-
mean_x12)))/(min(len(test35),len(test45)-2))
an2=np.dot(cov2**-1,(mean_x12-mean_x02))
bn2=(-1/2)*np.dot(np.dot((mean_x12-mean_x02).T,cov2**(-1)),(mean_x12-
mean_x02))
var_x0=(np.dot(an2,mean_x02.T)+bn2)/math.sqrt(np.dot(np.dot(an2,cov2),an2.T))
var_x1=(np.dot(an2,mean_x12.T)+bn2)/math.sqrt(np.dot(np.dot(an2,cov2),an2.T))
LDA_error=1/2*(ns.norm.cdf(var_x0)+ns.norm.cdf(-var_x1))
clas_x12_y=-bn2/an2[1]-an2[0]*x12[:,0]/an[1]
clas_x22_y=-bn/an[1]-an[0]*x22[:,0]/an[1]
error_time=0
error_set=list()
for t in range(min(len(test45),len(test35))):
   if x22[t,1] > clas_x22_y[t]:
      error_time=error_time+1
   if x12[t,1] < clas_x12_y[t]:
      error_time=error_time+1
error_set.append(error_time/(len(test_set)*len(test_set[1])))
print(error_set)
#print(LDA_error)
#LDA_error_set.append(LDA_error)
```

d. For part d we just need to change two variable to calculate the result that we want.
   With 3 predictors:

```
    for b in range(len(test35)):
       x112.append([test35[b][1],test35[b][2],test35[b][3]])
    for b in range(len(test45)):
       x212.append([test45[b][1],test45[b][2],test45[b][3]])
```

   With 4 predictors:

```
   for b in range(len(test35)):
      x112.append([test35[b][1],test35[b][2],test35[b][3],test35[b][0]])
   for b in range(len(test45)):
      x212.append([test45[b][1],test45[b][2],test45[b][3],test45[b][0]])
```

With 5 predictors:

```
for b in range(len(test35)):
    x112.append([test35[b][1],test35[b][2],test35[b][3],test35[b][0],test35[b][4]])
for b in range(len(test45)):
    x212.append([test45[b][1],test45[b][2],test45[b][3],test45[b][0],test35[b][4]])
```