In [273]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import heapq
import io

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from collections import Counter

df = pd.read_csv("Desktop/column_2C.csv")

df.head()
```

Out[273]:

	PI	PT	LLA	SS	PR	GOS	CLASS
0	63.03	22.55	39.61	40.48	98.67	-0.25	AB
1	39.06	10.06	25.02	29.00	114.41	4.56	AB
2	68.83	22.22	50.09	46.61	105.99	-3.53	AB
3	69.30	24.65	44.31	44.64	101.87	11.21	AB
4	49.71	9.65	28.32	40.06	108.17	7.92	AB

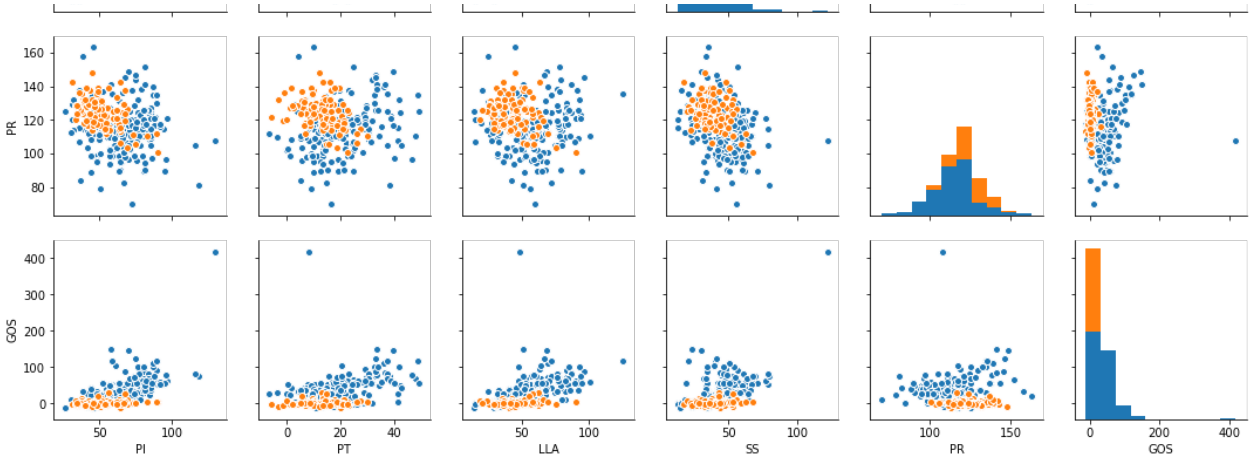
In [156]:

```
df['feature']=df['CLASS'].map({'AB':1,'NO':0})
```

In [284]:

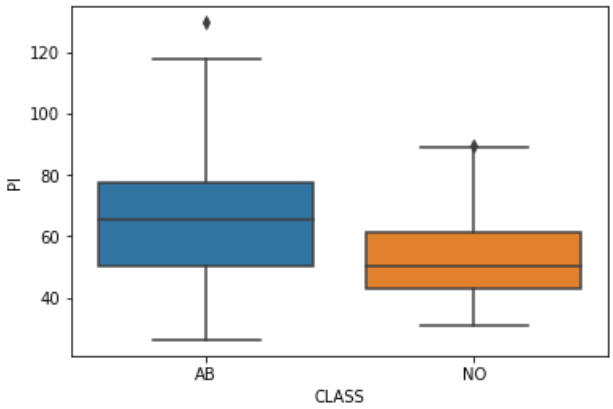
```
#scatterplot and boxplot

sns.pairplot(df,hue = 'CLASS')
plt.show()
```



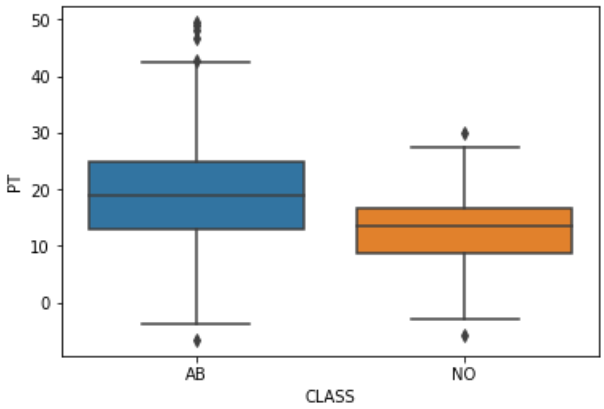
In [278]:

```
sns.boxplot(x='CLASS',y = 'PI',data = df)
plt.show()
```



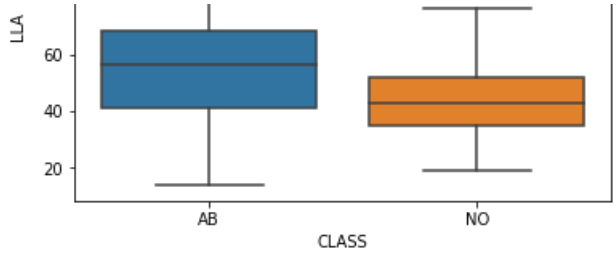
In [279]:

```
sns.boxplot(x='CLASS',y = 'PT',data = df)
plt.show()
```



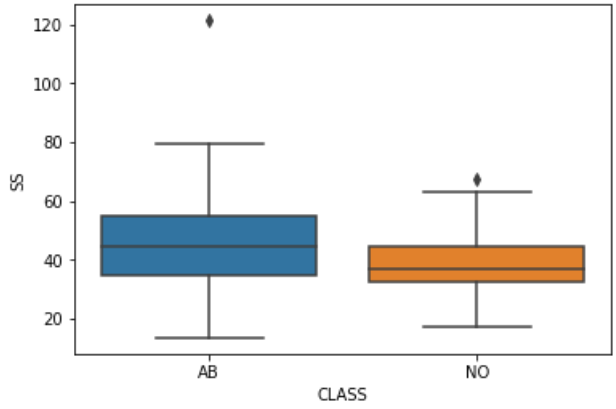
In [280]:

```
sns.boxplot(x='CLASS',y = 'LLA',data = df)
plt.show()
```



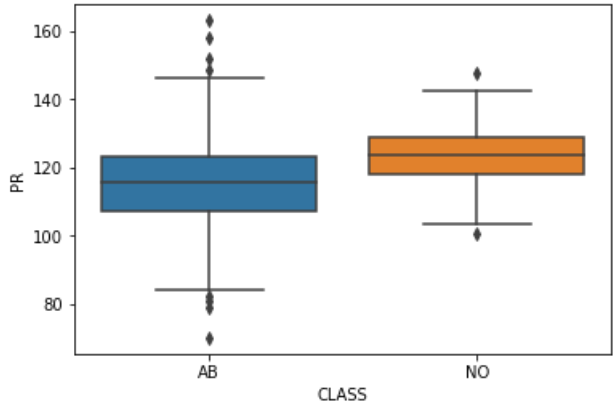
In [281]:

```
sns.boxplot(x='CLASS',y = 'SS',data = df)
plt.show()
```



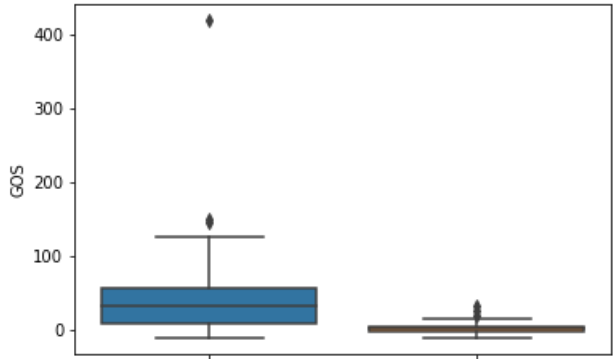
In [282]:

```
sns.boxplot(x='CLASS',y = 'PR',data = df)
plt.show()
```



In [283]:

```
sns.boxplot(x='CLASS',y = 'GOS',data = df)
plt.show()
```



AB NO

CLASS

In [164]:

```
#split data

trainAB=df[df['CLASS']=='AB'].iloc[:140,:]
trainNO=df[df['CLASS']=='NO'].iloc[:70,:]
train=pd.concat([trainAB,trainNO])
train
```

Out[164]:

	PI	PT	LLA	SS	PR	GOS	CLASS	feature
0	63.03	22.55	39.61	40.48	98.67	-0.25	AB	1
1	39.06	10.06	25.02	29.00	114.41	4.56	AB	1
2	68.83	22.22	50.09	46.61	105.99	-3.53	AB	1
3	69.30	24.65	44.31	44.64	101.87	11.21	AB	1
4	49.71	9.65	28.32	40.06	108.17	7.92	AB	1
5	40.25	13.92	25.12	26.33	130.33	2.23	AB	1
6	53.43	15.86	37.17	37.57	120.57	5.99	AB	1
7	45.37	10.76	29.04	34.61	117.27	-10.68	AB	1
8	43.79	13.53	42.69	30.26	125.00	13.29	AB	1
9	36.69	5.01	41.95	31.68	84.24	0.66	AB	1
10	49.71	13.04	31.33	36.67	108.65	-7.83	AB	1
11	31.23	17.72	15.50	13.52	120.06	0.50	AB	1
12	48.92	19.96	40.26	28.95	119.32	8.03	AB	1
13	53.57	20.46	33.10	33.11	110.97	7.04	AB	1
14	57.30	24.19	47.00	33.11	116.81	5.77	AB	1
15	44.32	12.54	36.10	31.78	124.12	5.42	AB	1
16	63.83	20.36	54.55	43.47	112.31	-0.62	AB	1
17	31.28	3.14	32.56	28.13	129.01	3.62	AB	1
18	38.70	13.44	31.00	25.25	123.16	1.43	AB	1
19	41.73	12.25	30.12	29.48	116.59	-1.24	AB	1
20	43.92	14.18	37.83	29.74	134.46	6.45	AB	1
21	54.92	21.06	42.20	33.86	125.21	2.43	AB	1
22	63.07	24.41	54.00	38.66	106.42	15.78	AB	1
23	45.54	13.07	30.30	32.47	117.98	-4.99	AB	1
24	36.13	22.76	29.00	13.37	115.58	-3.24	AB	1
25	54.12	26.65	35.33	27.47	121.45	1.57	AB	1
26	26.15	10.76	14.00	15.39	125.20	-10.09	AB	1
27	43.58	16.51	47.00	27.07	109.27	8.99	AB	1
28	44.55	21.93	26.79	22.62	111.07	2.65	AB	1
29	66.88	24.89	49.28	41.99	113.48	-2.01	AB	1
...	...	...	...	...	...	...	...	...
250	36.16	-0.81	33.63	36.97	135.94	-2.09	NO	0
251	40.75	1.84	50.00	38.91	139.25	0.67	NO	0
252	42.92	-5.85	58.00	48.76	121.61	-3.36	NO	0
253	63.79	21.35	66.00	42.45	119.55	12.38	NO	0

	PI	PT	LLA	SS	PR	GOS	CLASS	feature
254	72.96	19.58	61.01	53.38	111.23	0.81	NO	0
255	67.54	14.66	58.00	52.88	123.63	25.97	NO	0
256	54.75	9.75	48.00	45.00	123.04	8.24	NO	0
257	50.16	-2.97	42.00	53.13	131.80	-8.29	NO	0
258	40.35	10.19	37.97	30.15	128.01	0.46	NO	0
259	63.62	16.93	49.35	46.68	117.09	-0.36	NO	0
260	54.14	11.94	43.00	42.21	122.21	0.15	NO	0
261	74.98	14.92	53.73	60.05	105.65	1.59	NO	0
262	42.52	14.38	25.32	28.14	128.91	0.76	NO	0
263	33.79	3.68	25.50	30.11	128.33	-1.78	NO	0
264	54.50	6.82	47.00	47.68	111.79	-4.41	NO	0
265	48.17	9.59	39.71	38.58	135.62	5.36	NO	0
266	46.37	10.22	42.70	36.16	121.25	-0.54	NO	0
267	52.86	9.41	46.99	43.45	123.09	1.86	NO	0
268	57.15	16.49	42.84	40.66	113.81	5.02	NO	0
269	37.14	16.48	24.00	20.66	125.01	7.37	NO	0
270	51.31	8.88	57.00	42.44	126.47	-2.14	NO	0
271	42.52	16.54	42.00	25.97	120.63	7.88	NO	0
272	39.36	7.01	37.00	32.35	117.82	1.90	NO	0
273	35.88	1.11	43.46	34.77	126.92	-1.63	NO	0
274	43.19	9.98	28.94	33.22	123.47	1.74	NO	0
275	67.29	16.72	51.00	50.57	137.59	4.96	NO	0
276	51.33	13.63	33.26	37.69	131.31	1.79	NO	0
277	65.76	13.21	44.00	52.55	129.39	-1.98	NO	0
278	40.41	-1.33	30.98	41.74	119.34	-6.17	NO	0
279	48.80	18.02	52.00	30.78	139.15	10.44	NO	0

210 rows × 8 columns

In [165]:

```
testAB=df[df['CLASS']=='AB'].iloc[140:,:]
testNO=df[df['CLASS']=='NO'].iloc[70:,:]
test=pd.concat([testAB,testNO])
test
```

Out[165]:

	PI	PT	LLA	SS	PR	GOS	CLASS	feature
140	69.56	15.40	74.44	54.16	105.07	29.70	AB	1
141	89.50	48.90	72.00	40.60	134.63	118.35	AB	1
142	85.29	18.28	100.74	67.01	110.66	58.88	AB	1
143	60.63	20.60	64.54	40.03	117.23	104.86	AB	1
144	60.04	14.31	58.04	45.73	105.13	30.41	AB	1
145	85.64	42.69	78.75	42.95	105.14	42.89	AB	1
146	85.58	30.46	78.23	55.12	114.87	68.38	AB	1
147	55.08	-3.76	56.00	58.84	109.92	31.77	AB	1
148	65.76	9.83	50.82	55.92	104.39	39.31	AB	1
149	79.25	23.94	40.80	55.30	98.62	36.71	AB	1

	PI	PT	LLA	SS	PR	GOS	CLASS	feature
150	81.11	20.69	60.69	60.42	94.02	40.51	AB	1
151	48.03	3.97	58.34	44.06	125.35	35.00	AB	1
152	63.40	14.12	48.14	49.29	111.92	31.78	AB	1
153	57.29	15.15	64.00	42.14	116.74	30.34	AB	1
154	41.19	5.79	42.87	35.39	103.35	27.66	AB	1
155	66.80	14.55	72.08	52.25	82.46	41.69	AB	1
156	79.48	26.73	70.65	52.74	118.59	61.70	AB	1
157	44.22	1.51	46.11	42.71	108.63	42.81	AB	1
158	57.04	0.35	49.20	56.69	103.05	52.17	AB	1
159	64.27	12.51	68.70	51.77	95.25	39.41	AB	1
160	92.03	35.39	77.42	56.63	115.72	58.06	AB	1
161	67.26	7.19	51.70	60.07	97.80	42.14	AB	1
162	118.14	38.45	50.84	79.70	81.02	74.04	AB	1
163	115.92	37.52	76.80	78.41	104.70	81.20	AB	1
164	53.94	9.31	43.10	44.64	124.40	25.08	AB	1
165	83.70	20.27	77.11	63.43	125.48	69.28	AB	1
166	56.99	6.87	57.01	50.12	109.98	36.81	AB	1
167	72.34	16.42	59.87	55.92	70.08	12.07	AB	1
168	95.38	24.82	95.16	70.56	89.31	57.66	AB	1
169	44.25	1.10	38.00	43.15	98.27	23.91	AB	1
...	...	...	...	...	...	...	...	...
280	50.09	13.43	34.46	36.66	119.13	3.09	NO	0
281	64.26	14.50	43.90	49.76	115.39	5.95	NO	0
282	53.68	13.45	41.58	40.24	113.91	2.74	NO	0
283	49.00	13.11	51.87	35.88	126.40	0.54	NO	0
284	59.17	14.56	43.20	44.60	121.04	2.83	NO	0
285	67.80	16.55	43.26	51.25	119.69	4.87	NO	0
286	61.73	17.11	46.90	44.62	120.92	3.09	NO	0
287	33.04	-0.32	19.07	33.37	120.39	9.35	NO	0
288	74.57	15.72	58.62	58.84	105.42	0.60	NO	0
289	44.43	14.17	32.24	30.26	131.72	-3.60	NO	0
290	36.42	13.88	20.24	22.54	126.08	0.18	NO	0
291	51.08	14.21	35.95	36.87	115.80	6.91	NO	0
292	34.76	2.63	29.50	32.12	127.14	-0.46	NO	0
293	48.90	5.59	55.50	43.32	137.11	19.85	NO	0
294	46.24	10.06	37.00	36.17	128.06	-5.10	NO	0
295	46.43	6.62	48.10	39.81	130.35	2.45	NO	0
296	39.66	16.21	36.67	23.45	131.92	-4.97	NO	0
297	45.58	18.76	33.77	26.82	116.80	3.13	NO	0
298	66.51	20.90	31.73	45.61	128.90	1.52	NO	0
299	82.91	29.89	58.25	53.01	110.71	6.08	NO	0
300	50.68	6.46	35.00	44.22	116.59	-0.21	NO	0
301	89.01	26.08	69.02	62.94	111.48	6.06	NO	0
302	54.60	21.49	29.36	33.11	118.34	-1.47	NO	0
303	34.38	2.06	32.39	32.32	128.30	-3.37	NO	0

	PI	PT	LLA	SS	PR	GOS	CLASS	feature
304	45.08	12.31	44.58	32.77	147.89	-8.94	NO	0
305	47.90	13.62	36.00	34.29	117.45	-4.25	NO	0
306	53.94	20.72	29.22	33.22	114.37	-0.42	NO	0
307	61.45	22.69	46.17	38.75	125.67	-2.71	NO	0
308	45.25	8.69	41.58	36.56	118.55	0.21	NO	0
309	33.84	5.07	36.64	28.77	123.95	-0.20	NO	0

100 rows × 8 columns

In [166]:

```
#(c)i

K=[]
Errortest=[]
Errortrain=[]
for k in range(1,len(train),3):
    K.append(k)
    neighbor=KNeighborsClassifier(n_neighbors=k)
    neighbor.fit(train.iloc[:,:6],train.loc[:,'feature'])
    predtesty=neighbor.predict(test.iloc[:,:6])
    predtrainy=neighbor.predict(train.iloc[:,:6])
    cmtest=confusion_matrix(test.loc[:,'feature'],predtesty)
    cmtrain=confusion_matrix(train.loc[:,'feature'],predtrainy)
    errortest=((cmtest[0,1]+cmtest[1,0])/len(test))
    errortrain=((cmtrain[0,1]+cmtrain[1,0])/len(train))

    Errortest.append(errortest)
    Errortrain.append(errortrain)
```

In [173]:

```
#(c)ii

table=pd.DataFrame({'K':K,'test error':Errortest,'train error':Errortrain})
```

In [174]:

```
plt.figure
plt.plot(table['K'],table['test error'])
plt.plot(table['K'],table['train error'])
plt.title('train error v.s. test error')
plt.xlabel('K')
```

Out[174]:

```
<matplotlib.text.Text at 0x136af1860>
```

In [127]:

```
# From the plot above, we could know that the optimal k is 4

neighbor=KNeighborsClassifier(n_neighbors=4)
    neighbor.fit(train.iloc[:,:6],train.loc[:,'feature'])
    predtesty=neighbor.predict(test.iloc[:,:6])
    predtrainy=neighbor.predict(train.iloc[:,:6])
    cmtest=confusion_matrix(test.loc[:,'feature'],predtesty)
    cmtrain=confusion_matrix(train.loc[:,'feature'],predtrainy)


TN=cmtest[0,0]/cmtest[0,0]+cmtest[0,1]
TP=cmtest[1,1]/cmtest[1,0]+cmtest[1,1]
precision=cmtest[1,1]/cmtest[0,1]+cmtest[1,1]
F=(1+0.3**2)*(TP+precision)/(0.3**2*pecision+TP)
```

```
  File "<ipython-input-127-ab6c0e4abc7e>", line 4
    neighbor.fit(train.iloc[:,:6],train.loc[:,'feature'])
    ^
IndentationError: unexpected indent
```

In [175]:

```
# From the plot above, we could know that the optimal k is 4

neighbor=KNeighborsClassifier(n_neighbors=4)
neighbor.fit(train.iloc[:,:6],train.loc[:,'feature'])
predtesty=neighbor.predict(test.iloc[:,:6])
predtrainy=neighbor.predict(train.iloc[:,:6])
cmtest=confusion_matrix(test.loc[:,'feature'],predtesty)
cmtrain=confusion_matrix(train.loc[:,'feature'],predtrainy)

TN=cmtest[0,0]/cmtest[0,0]+cmtest[0,1]
TP=cmtest[1,1]/cmtest[1,0]+cmtest[1,1]
precision=cmtest[1,1]/cmtest[0,1]+cmtest[1,1]
F=(1+0.3**2)*(TP+precision)/(0.3**2*precision+TP)

TN
```

Out[175]:

```
6.0
```

In [176]:

```
TP
```

Out[176]:

```
138.0
```

In [177]:

```
precision
```

Out[177]:

```
82.8
```

In [178]:

```
F
```

Out[178]:

```
1.6546489563567364
```

In [191]:

```
#(c)iii
K=[]
N=[]
Errortest=[]
Errortrain=[]

for n in range(10,211,10):
    trainAB=df[df['CLASS']=='AB'].iloc[:n-round(n/3),:]
    trainNO=df[df['CLASS']=='NO'].iloc[:round(n/3),:]
    train=pd.concat([trainAB,trainNO])
    testAB=df[df['CLASS']=='AB'].iloc[n-round(n/3):,:]
    testNO=df[df['CLASS']=='NO'].iloc[round(n/3):,:]
    test=pd.concat([testAB,testNO])


for k in range(1,n,5):
    neighbor=KNeighborsClassifier(n_neighbors=k)
    neighbor.fit(train.iloc[:,:6],train.loc[:,'feature'])
```

```
    predtesty=neighbor.predict(test.iloc[:,:6])
    predtrainy=neighbor.predict(train.iloc[:,:6])
    cmtest=confusion_matrix(test.loc[:,'feature'],predtesty)
    cmtrain=confusion_matrix(train.loc[:,'feature'],predtrainy)
    errortest=((cmtest[0,1]+cmtest[1,0])/len(test))
    errortrain=((cmtrain[0,1]+cmtrain[1,0])/len(train))
    K.append(k)
    N.append(n)
    Errortest.append(errortest)
    Errortrain.append(errortrain)
```

In [195]:

```
tableadjust=pd.DataFrame({'train size':N,'K':K,'test error':Errortest,'train error':Errortrain})
tableadjust[(tableadjust['test error']==min(tableadjust['test error']))]
```

Out[195]:

	K	test error	train error	train size
1	6	0.08	0.147619	210

In [212]:

```
#(d)

#i Minkowski Distance

#a.Manhattan Distance

#i
K=[]
Errortest=[]
Errortrain=[]

trainAB=df[df['CLASS']=='AB'].iloc[:140,:]
trainNO=df[df['CLASS']=='NO'].iloc[:70,:]
train=pd.concat([trainAB,trainNO])
testAB=df[df['CLASS']=='AB'].iloc[140:,:]
testNO=df[df['CLASS']=='NO'].iloc[70:,:]
test=pd.concat([testAB,testNO])

for k in range(1,197,5):
    neighbor=KNeighborsClassifier(n_neighbors=k,metric='minkowski',p=1)
    neighbor.fit(train.iloc[:,:6],train.loc[:,'feature'])
    predtesty=neighbor.predict(test.iloc[:,:6])
    predtrainy=neighbor.predict(train.iloc[:,:6])
    cmtest=confusion_matrix(test.loc[:,'feature'],predtesty)
    cmtrain=confusion_matrix(train.loc[:,'feature'],predtrainy)
    errortest=((cmtest[0,1]+cmtest[1,0])/len(test))
    errortrain=((cmtrain[0,1]+cmtrain[1,0])/len(train))
    K.append(k)
    Errortest.append(errortest)
    Errortrain.append(errortrain)

table_min=pd.DataFrame({'K':K,'test error':Errortest,'train error':Errortrain})
table_min[(table_min['test error']==min(table_min['test error']))]

#optimal k is 6
```

Out[212]:

	K	test error	train error
1	6	0.11	0.138095
2	11	0.11	0.142857
5	26	0.11	0.166667

In [218]:

```
#ii select p
I=[]
Errortest=[]
Errortrain=[]


for i in range(1,11):
    I.append(i/10)
    neighbor=KNeighborsClassifier(n_neighbors=k,metric='minkowski',p=10**(i/10))
    neighbor.fit(train.iloc[:,:6],train.loc[:,'feature'])
    predtesty=neighbor.predict(test.iloc[:,:6])
    predtrainy=neighbor.predict(train.iloc[:,:6])
    cmtest=confusion_matrix(test.loc[:,'feature'],predtesty)
    cmtrain=confusion_matrix(train.loc[:,'feature'],predtrainy)
    errortest=((cmtest[0,1]+cmtest[1,0])/len(test))
    errortrain=((cmtrain[0,1]+cmtrain[1,0])/len(train))
    K.append(k)
    Errortest.append(errortest)
    Errortrain.append(errortrain)

table_min1=pd.DataFrame({'log10p':I,'test error':Errortest,'train error':Errortrain})
table_min1[(table_min1['test error']==min(table_min1['test error']))]
```

Out[218]:

	log10p	test error	train error
0	0.1	0.3	0.333333
1	0.2	0.3	0.333333
2	0.3	0.3	0.333333
3	0.4	0.3	0.333333
4	0.5	0.3	0.333333
5	0.6	0.3	0.333333
6	0.7	0.3	0.333333
7	0.8	0.3	0.333333
8	0.9	0.3	0.333333
9	1.0	0.3	0.333333

In [276]:

```
#Mahalanobis Distance

K=[]
Errortest=[]
Errortrain=[]

trainAB=df[df['CLASS']=='AB'].iloc[:140,:]
trainNO=df[df['CLASS']=='NO'].iloc[:70,:]
train=pd.concat([trainAB,trainNO])
testAB=df[df['CLASS']=='AB'].iloc[140:,:]
testNO=df[df['CLASS']=='NO'].iloc[70:,:]
test=pd.concat([testAB,testNO])
x=np.array(train.iloc[:,:6])
y=np.array(test.iloc[:,:6])
test1=np.cov(df.iloc[:,:6].T)
train1=np.cov(train.iloc[:,:6].T)
invtest=np.linalg.inv(test1)
invtrain=np.linalg.inv(train1)

for k in range(1,197,5):
    testfeature=[]
    trainfeature=[]
    for j in range(len(test)):
        distances=[]
        for i in range(len(train)):
            distances.append(np.sqrt(np.dot(np.dot(x[i]-y[j],invtest),(x[i]-y[j]).T)))
```

```
        kmin=heapq.nsmallest(k,distances)
        index=[]
        for distance in kmin:
            index.append(distances.index(distance))
        knn=train.iloc[index,7]
        testfeature.append(Counter(knn).most_common()[0][0])
    for j in range(len(train)):
        distance=[]
        for i in range (len(train)):
            distances.append(np.sqrt(np.dot(np.dot(x[i]-y[j],invtrain),(x[i]-y[j]).T)))

        kmin=heapq.nsmallest(k,distances)
        index=[]
        for distance in kmin:
            index.append(distances,index(distance))
        knn=train.iloc[index,7]
        trainfeature.append(Counter(knn).most_common()[0][0])
        cmtest=confusion_matrix(test.loc[:,'feature'],testfeature)
        cmtrain=confusion_matrix(train.loc[:,'feature'],trainfeature)

        errortest=((cmtest[0,1]+cmtest[1,0])/len(test))
        errortrain=((cmtrain[0,1]+cmtrain[1,0])/len(train))
        K.append(k)
        Errortest.append(errortest)
        Errortrain.append(errortrain)

table_maha=pd.DataFrame({'K':K,'test error':Errortest,'train error':Errortrain})
table_min1[(table_maha['test error']==min(table_maha['test error']))]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-276-1335199554f1> in <module>()
     29         for distance in kmin:
     30             index.append(distances.index(distance))
---> 31         knn=train.iloc[index,7]
     32         testfeature.append(Counter(knn).most_common()[0][0])
     33     for j in range(len(train)):

/Users/chengshihan/anaconda/lib/python3.5/site-packages/pandas/core/indexing.py in
__getitem__(self, key)
   1323             except (KeyError, IndexError):
   1324                 pass
-> 1325             return self._getitem_tuple(key)
   1326         else:
   1327             key = com._apply_if_callable(key, self.obj)

/Users/chengshihan/anaconda/lib/python3.5/site-packages/pandas/core/indexing.py in _getitem_tuple(
self, tup)
   1660     def _getitem_tuple(self, tup):
   1661
-> 1662         self._has_valid_tuple(tup)
   1663         try:
   1664             return self._getitem_lowerdim(tup)

/Users/chengshihan/anaconda/lib/python3.5/site-packages/pandas/core/indexing.py in
_has_valid_tuple(self, key)
    187             if i >= self.obj.ndim:
    188                 raise IndexingError('Too many indexers')
--> 189             if not self._has_valid_type(k, i):
    190                 raise ValueError("Location based indexing can only have [%s] "
    191                                  "types" % self._valid_types)

/Users/chengshihan/anaconda/lib/python3.5/site-packages/pandas/core/indexing.py in _has_valid_type
(self, key, axis)
   1595             return True
   1596         elif is_integer(key):
-> 1597             return self._is_valid_integer(key, axis)
   1598         elif is_list_like_indexer(key):
   1599             return self._is_valid_list_like(key, axis)

/Users/chengshihan/anaconda/lib/python3.5/site-packages/pandas/core/indexing.py in
```

```
_is_valid_integer(self, key, axis)
   1636         l = len(ax)
   1637         if key >= l or key < -l:
-> 1638             raise IndexError("single positional indexer is out-of-bounds")
   1639         return True
   1640

IndexError: single positional indexer is out-of-bounds
```

In [ ]:

```
#(f)

#Until what I did now, the lowest training error is 0.138095
```