In [2]:

```
import pandas as pd
import numpy as np
from itertools import combinations
from sklearn import preprocessing
from sklearn.svm import LinearSVC,SVC
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score,train_test_split
from sklearn.neural_network import MLPClassifier
```

In [3]:

```
data = pd.read_csv('Assign1_Training_Data.txt',delimiter="\t")
data.head()
d = np.asarray(data)
data.head()
```

Out[3]:

	Sample_Number	AL080059	Contig63649_RC	Contig46218_RC	LOC51203	AA555029_RC	ALDH4	Contig38288_RC	FGF18	Contig2
0	138	-0.227	-0.107	-0.086	-0.057	0.073	0.021	-0.002	0.135	
1	184	0.044	-0.031	0.381	0.226	-0.038	-0.167	0.103	-0.330	
2	127	0.151	-0.210	0.034	0.037	-0.065	-0.048	-0.026	-0.425	
3	166	0.335	-0.031	0.177	0.165	-0.372	0.340	0.112	-0.506	
4	318	-0.098	-0.492	-0.307	-0.097	-0.024	0.197	-0.001	0.369	

5 rows × 72 columns

In [4]:

```
data_testing = pd.read_csv('Assign1_Testing_Data.txt',delimiter="\t")
d_test = np.asarray(data_testing)
Y_test =   d_test[:,-1]
X_test =   d_test[:,1:71]

x0_test = []
y0_test = []
x1_test = []
y1_test = []
for i in range(len(Y_test)):
    if Y_test[i]==0:
        x0_test.append(X_test[i])
        y0_test.append(Y_test[i])
    else:
        x1_test.append(X_test[i])
        y1_test.append(Y_test[i])

print('Number of  0 class in Test set is =',len(x0_test))
print('Number of  1 class in Test set is',len(x1_test))
print('Number of  Total classes in Test set is=',len(X_test))
```

```
Number of  0 class in Test set is = 39
Number of  1 class in Test set is 176
Number of  Total classes in Test set is= 215
```

In [5]:

```
Y =   d[:,-1]
X =   d[:,1:71]
print('Some Features --> ',X[0:5,0:3])
p = np.vstack((X[:,0], X[:,1],X[:,2])).T
print(X[:,2].shape)
```

```
#p = np.vstack((p, )).T
#p.shape
#X.shape
#p
x0 = []
y0 = []
x1 = []
y1 = []
for i in range(len(Y)):
    if Y[i]==0:
        x0.append(X[i])
        y0.append(Y[i])
    else:
        x1.append(X[i])
        y1.append(Y[i])

print('Number of  0 class in Train set is',len(x0))
print('Number of  1 class in Train set is',len(x1))
print('Number of  Total classes in Train set is',len(X))
```

```
Some Features -->  [[-0.227 -0.107 -0.086]
 [ 0.044 -0.031  0.381]
 [ 0.151 -0.21   0.034]
 [ 0.335 -0.031  0.177]
 [-0.098 -0.492 -0.307]]
(80,)
Number of  0 class in Train set is 40
Number of  1 class in Train set is 40
Number of  Total classes in Train set is 80
```

In [6]:

```
x0 = np.array(x0)
x1 = np.array(x1)
x0_test = np.array(x0_test)
x1_test = np.array(x1_test)
```

In [7]:

```
def fs_es(X,Y,clf,x0,x0_test,y0,y0_test,x1,x1_test,y1,y1_test,X_test,Y_test):
    c = X.shape[1]
    temp_e = np.inf
    e = 0
    for i in range(c):
        for j in range(i+1,c):
            x = np.vstack((X[:,i], X[:,j])).T
            y = Y
            Xx_train, Xx_test, yy_train, yy_test = train_test_split(x,y, test_size=0.0,
random_state=0)
            e = 1 - clf.fit(Xx_train, yy_train).score(Xx_train, yy_train)
            #print(e)
            if e < temp_e:
                index1 = i
                index2 = j
                temp_e = e
    x0_ = np.vstack((x0[:,index1],x0[:,index2])).T
    x1_ = np.vstack((x1[:,index1],x1[:,index2])).T
    x0_test_ = np.vstack((x0_test[:,index1],x0_test[:,index2])).T
    x1_test_ = np.vstack((x1_test[:,index1],x1_test[:,index2])).T


    x_test_ = np.vstack((X_test[:,index1],X_test[:,index2])).T

    e0_train  = 1 - clf.score(x0_,y0)
    e1_train  = 1 - clf.score(x1_,y1)
    e0_test  = 1 - clf.score(x0_test_,y0_test)
    e1_test  = 1 - clf.score(x1_test_,y1_test)

    e1_total  = 1 - clf.score(x_test_,Y_test)


    return temp_e,index1,index2,e0_train,e1_train,e0_test,e1_test,e1_total
```

In [8]:

```
def fs_sfs3(ind1,ind2,X,Y,clf,x0,x0_test,y0,y0_test,x1,x1_test,y1,y1_test,X_test,Y_test):
    c = X.shape[1]

    temp_e = np.inf
    e = 0
    for i in range(c):
        if (i != ind1 and i != ind2):
            x = np.vstack((X[:,ind1], X[:,ind2], X[:,i])).T
            y = Y
            Xx_train, Xx_test, yy_train, yy_test = train_test_split(x,y, test_size=0.0,
random_state=0)
            e = 1 - clf.fit(Xx_train, yy_train).score(Xx_train, yy_train)
            #print(e)
            if e < temp_e:
                index = i
                temp_e = e
    x0_ = np.vstack((x0[:,ind1],x0[:,ind2],x0[:,index])).T
    x1_ = np.vstack((x1[:,ind1],x1[:,ind2],x1[:,index])).T
    x0_test_ = np.vstack((x0_test[:,ind1],x0_test[:,ind2],x0_test[:,index])).T
    x1_test_ = np.vstack((x1_test[:,ind1],x1_test[:,ind2],x1_test[:,index])).T
    x_test_ = np.vstack((X_test[:,ind1],X_test[:,ind2],X_test[:,index])).T

    e0_train  = 1 - clf.score(x0_,y0)
    e1_train  = 1 - clf.score(x1_,y1)
    e0_test  = 1 - clf.score(x0_test_,y0_test)
    e1_test  = 1 - clf.score(x1_test_,y1_test)

    e1_total  = 1 - clf.score(x_test_,Y_test)




    return temp_e,index,e0_train,e1_train,e0_test,e1_test,e1_total
```

In [9]:

```
def fs_sfs4(ind1,ind2,ind3,X,Y,clf,x0,x0_test,y0,y0_test,x1,x1_test,y1,y1_test,X_test,Y_test):
    c = X.shape[1]

    temp_e = np.inf
    e = 0
    for i in range(c):
        if (i != ind1 and i != ind2 and i != ind3):
            x = np.vstack((X[:,ind1], X[:,ind2], X[:,ind3],X[:,i])).T
            y = Y
            Xx_train, Xx_test, yy_train, yy_test = train_test_split(x,y, test_size=0.0,
random_state=0)
            e = 1 - clf.fit(Xx_train, yy_train).score(Xx_train, yy_train)
            #print(e)
            if e < temp_e:
                index = i
                temp_e = e

    x0_ = np.vstack((x0[:,ind1],x0[:,ind2],x0[:,ind3],x0[:,index])).T
    x1_ = np.vstack((x1[:,ind1],x1[:,ind2],x1[:,ind3],x1[:,index])).T


    x0_test_ = np.vstack((x0_test[:,ind1],x0_test[:,ind2],x0_test[:,ind3],x0_test[:,index])).T
    x1_test_ = np.vstack((x1_test[:,ind1],x1_test[:,ind2],x1_test[:,ind3],x1_test[:,index])).T

    x_test_ = np.vstack((X_test[:,ind1],X_test[:,ind2],X_test[:,ind3],X_test[:,index])).T

    e0_train  = 1 - clf.score(x0_,y0)
    e1_train  = 1 - clf.score(x1_,y1)
    e0_test  = 1 - clf.score(x0_test_,y0_test)
    e1_test  = 1 - clf.score(x1_test_,y1_test)

    e1_total  = 1 - clf.score(x_test_,Y_test)
    return temp_e,index,e0_train,e1_train,e0_test,e1_test,e1_total
```

In [10]:

```
def fs_sfs5(ind1,ind2,ind3,ind4,X,Y,clf,x0,x0_test,y0,y0_test,x1,x1_test,y1,y1_test,X_test,Y_test):
```

```
    c = X.shape[1]

    temp_e = np.inf
    e = 0
    for i in range(c):
        if (i != ind1 and i != ind2 and i != ind3 and i != ind4):
            x = np.vstack((X[:,ind1], X[:,ind2], X[:,ind3],X[:,ind4],X[:,i])).T
            y = Y
            Xx_train, Xx_test, yy_train, yy_test = train_test_split(x,y, test_size=0.0,
random_state=0)
            e = 1 - clf.fit(Xx_train, yy_train).score(Xx_train, yy_train)
            #print(e)
            if e < temp_e:
                index = i
                temp_e = e
    x0_ = np.vstack((x0[:,ind1],x0[:,ind2],x0[:,ind3],x0[:,ind4],x0[:,index])).T
    x1_ = np.vstack((x1[:,ind1],x1[:,ind2],x1[:,ind3],x1[:,ind4],x1[:,index])).T
    e0_train  = 1 - clf.score(x0_,y0)
    e1_train  = 1 - clf.score(x1_,y1)

    x0_test_ = np.vstack((x0_test[:,ind1],x0_test[:,ind2],x0_test[:,ind3],x0_test[:,ind4],x0_test[
:,index])).T
    x1_test_ = np.vstack((x1_test[:,ind1],x1_test[:,ind2],x1_test[:,ind3],x1_test[:,ind4],x1_test[
:,index])).T
    e0_test  = 1 - clf.score(x0_test_,y0_test)
    e1_test  = 1 - clf.score(x1_test_,y1_test)

    x_test_ =
np.vstack((X_test[:,ind1],X_test[:,ind2],X_test[:,ind3],X_test[:,ind4],X_test[:,index])).T
    e1_total  = 1 - clf.score(x_test_,Y_test)
    return temp_e,index,e0_train,e1_train,e0_test,e1_test,e1_total
```

1) Linear SVM

1.1 Linear SVM NO feature selection

In [11]:

```
clf_svmL = SVC(kernel='linear', C=1)
Xn_train, Xn_test, yn_train, yn_test = train_test_split(X,Y, test_size=0.0, random_state=0)
e = 1 - clf_svmL.fit(Xn_train, yn_train).score(Xn_train, yn_train)
e0_train_svm2_n = 1 - clf_svmL.score(x0, y0)
e1_train_svm2_n = 1 - clf_svmL.score(x1, y1)
e0_test_svm2_n  = 1 - clf_svmL.score(x0_test, y0_test)
e1_test_svm2_n  = 1 - clf_svmL.score(x1_test, y1_test)
e1_total_svm2_n = 1 - clf_svmL.score(X_test, Y_test)
print('Resubstitution error = ',e)
print("Train Error on class 0",e0_train_svm2_n)
print("Train Error on class 1",e1_train_svm2_n)
print("Test Error on class 0",e0_test_svm2_n)
print("Test Error on class 1",e1_test_svm2_n)
print("Test Error Total",e1_total_svm2_n)
```

```
Resubstitution error =  0.050000000000000044
Train Error on class 0 0.07499999999999996
Train Error on class 1 0.025000000000000022
Test Error on class 0 0.3846153846153846
Test Error on class 1 0.3295454545454546
Test Error Total 0.3395348837209302
```

1.2 Linear SVM 2-feature selection extraction

In [12]:

```
e,feature1_svmL,feature2_svmL,e0_train_svm2_2,e1_train_svm2_2,e0_test_svm2_2,e1_test_svm2_2,e1_tota
l_svm2_2  = fs_es(X,Y,clf_svmL,x0,x0_test,y0,
                                                                                y0_test,x1,x1_test
,y1_test,X_test,Y_test)
print('Resubstitution error = ',e,'\nFeature 1 = ', feature1_svmL,' Feature 2 = ',feature2_svmL)
print("Train Error on class 0",e0_train_svm2_2)
print("Train Error on class 1",e1_train_svm2_2)
```

```
print("Train Error on class 1",e1_train_svm2_2)
print("Test Error on class 0",e0_test_svm2_2)
print("Test Error on class 1",e1_test_svm2_2)
print("Test Error Total",e1_total_svm2_2)
```

```
Resubstitution error =  0.21250000000000002
Feature 1 =  3  Feature 2 =  19
Train Error on class 0 0.32499999999999996
Train Error on class 1 0.25
Test Error on class 0 0.41025641025641024
Test Error on class 1 0.36931818181818177
Test Error Total 0.3767441860465116
```

1.3 Linear SVM 3-feature selection sequential forward search

In [13]:

```
e,feature3_svmL,e0_train_svm2_3,e1_train_svm2_3,e0_test_svm2_3,e1_test_svm2_3,e1_total_svm2_3 = fs_
sfs3(feature1_svmL,feature2_svmL,X,Y,clf_svmL,

x0,x0_test,y0,y0_test,x1,x1_test,y1,y1_test,X_test,Y_test)
print('Resubstitution error = ',e,'\nFeature 1 = ', feature1_svmL,' Feature 2 = ',feature2_svmL,'
Feature 3 = ',feature3_svmL)
print("Train Error on class 0",e0_train_svm2_3)
print("Train Error on class 1",e1_train_svm2_3)
print("Test Error on class 0",e0_test_svm2_3)
print("Test Error on class 1",e1_test_svm2_3)
print("Test Error Total",e1_total_svm2_3)
```

```
Resubstitution error =  0.19999999999999996
Feature 1 =  3  Feature 2 =  19  Feature 3 =  27
Train Error on class 0 0.050000000000000044
Train Error on class 1 0.44999999999999996
Test Error on class 0 0.15384615384615385
Test Error on class 1 0.5738636363636364
Test Error Total 0.4976744186046511
```

1.4 Linear SVM 4-feature selection sequential forward search

In [14]:

```
e,feature4_svmL,e0_train_svm2_4,e1_train_svm2_4,e0_test_svm2_4,e1_test_svm2_4,e1_total_svm2_4 = fs_
sfs4(feature1_svmL,feature2_svmL,feature3_svmL,X,Y,clf_svmL,

x0,x0_test,y0,y0_test,x1,x1_test,y1,y1_test,X_test,Y_test)
print('Resubstitution error = ',e,'\nFeature 1 = ', feature1_svmL,' Feature 2 = ',feature2_svmL,'
Feature 3 = ',feature3_svmL,' Feature 4 = ',feature4_svmL)
print("Train Error on class 0",e0_train_svm2_4)
print("Train Error on class 1",e1_train_svm2_4)
print("Test Error on class 0",e0_test_svm2_4)
print("Test Error on class 1",e1_test_svm2_4)
print("Test Error Total",e1_total_svm2_4)
```

```
Resubstitution error =  0.1875
Feature 1 =  3  Feature 2 =  19  Feature 3 =  27  Feature 4 =  5
Train Error on class 0 0.050000000000000044
Train Error on class 1 0.42500000000000004
Test Error on class 0 0.1282051282051282
Test Error on class 1 0.5454545454545454
Test Error Total 0.469767441860465115
```

1.5 Linear SVM 5-feature selection sequential forward search

In [15]:

```
e,feature5_svmL,e0_train_svm2_5,e1_train_svm2_5,e0_test_svm2_5,e1_test_svm2_5,e1_total_svm2_5 = fs_
sfs5(feature1_svmL,feature2_svmL,feature3_svmL,feature4_svmL,X,Y,clf_svmL,

x0,x0_test,y0,y0_test,x1,x1_test,y1,y1_test,X_test,Y_test)
```

```
print('Resubstitution error = ',e,'\nFeature 1 = ', feature1_svmL,' Feature 2 = ',feature2_svmL,'
Feature 3 = ',feature3_svmL,' Feature 4 = ',feature4_svmL,' Feature 5 = ',feature5_svmL)
print("Train Error on class 0",e0_train_svm2_5)
print("Train Error on class 1",e1_train_svm2_5)
print("Test Error on class 0",e0_test_svm2_5)
print("Test Error on class 1",e1_test_svm2_5)
print("Test Error Total",e1_total_svm2_5)
```

```
Resubstitution error =  0.1875
Feature 1 =  3  Feature 2 =  19  Feature 3 =  27  Feature 4 =  5  Feature 5 =  44
Train Error on class 0 0.09999999999999998
Train Error on class 1 0.375
Test Error on class 0 0.15384615384615385
Test Error on class 1 0.5284090909090908
Test Error Total 0.4604651162790697
```

2) Non Linear SVM

2.1 Non Linear SVM no feature selection

In [16]:

```
clf_svmNL = SVC(gamma ='auto' ,kernel='rbf', C=10)
Xn_train, Xn_test, yn_train, yn_test = train_test_split(X,Y, test_size=0.0, random_state=0)
e = 1 - clf_svmNL.fit(Xn_train, yn_train).score(Xn_train, yn_train)
e0_trainsvmNL_n = 1-clf_svmNL.score(x0, y0)
e1_trainsvmNL_n = 1-clf_svmNL.score(x1, y1)
e0_testsvmNL_n  = 1-clf_svmNL.score(x0_test, y0_test)
e1_testsvmNL_n  = 1-clf_svmNL.score(x1_test, y1_test)
e1_totalsvmNL_n = 1-clf_svmNL.score(X_test, Y_test)
print('Resubstitution error = ',e)
print("Train Error on class 0",e0_trainsvmNL_n)
print("Train Error on class 1",e1_trainsvmNL_n)
print("Test Error on class 0",e0_testsvmNL_n)
print("Test Error on class 1",e1_testsvmNL_n)
print("Test Error Total",e1_totalsvmNL_n)
```

```
Resubstitution error =  0.0625
Train Error on class 0 0.09999999999999998
Train Error on class 1 0.025000000000000022
Test Error on class 0 0.3846153846153846
Test Error on class 1 0.3352272727272727
Test Error Total 0.3441860465116279
```

2.2 Non-Linear SVM 2-feature selection extraction

In [17]:

```
e,feature1_svmNL,feature2_svmNL,e0_train_svmNL_2,e1_train_svmNL_2,e0_test_svmNL_2,e1_test_svmNL_2,e
1_total_svmNL_2  = fs_es(X,Y,clf_svmNL,x0,x0_test,y0,
                                                                    y0_test,x1,x1_test
,y1_test,X_test,Y_test)
print('Resubstitution error = ',e,'\nFeature 1 = ', feature1_svmNL,' Feature 2 = ',feature2_svmNL)
print("Train Error on class 0",e0_train_svmNL_2)
print("Train Error on class 1",e1_train_svmNL_2)
print("Test Error on class 0",e0_test_svmNL_2)
print("Test Error on class 1",e1_test_svmNL_2)
print("Test Error Total",e1_total_svmNL_2)
```

```
Resubstitution error =  0.21250000000000002
Feature 1 =  12  Feature 2 =  48
Train Error on class 0 0.32499999999999996
Train Error on class 1 0.42500000000000004
Test Error on class 0 0.17948717948717952
Test Error on class 1 0.4602272727272727
Test Error Total 0.40930232558139534
```

2.3 Non Linear SVM 3-feature selection sequential forward search

In [18]:

```
e,feature3_svmNL,e0_train_svmNL_3,e1_train_svmNL_3,e0_test_svmNL_3,e1_test_svmNL_3,e1_total_svmNL_3
= fs_sfs3(feature1_svmNL,feature2_svmNL,X,Y,clf_svmNL,

x0,x0_test,y0,y0_test,x1,x1_test,y1,y1_test,X_test,Y_test)
print('Resubstitution error = ',e,'\nFeature 1 = ', feature1_svmNL,' Feature 2 = ',feature2_svmNL,
' Feature 3 = ',feature3_svmNL)
print("Train Error on class 0",e0_train_svmNL_3)
print("Train Error on class 1",e1_train_svmNL_3)
print("Test Error on class 0",e0_test_svmNL_3)
print("Test Error on class 1",e1_test_svmNL_3)
print("Test Error Total",e1_total_svmNL_3)
```

```
Resubstitution error =  0.17500000000000004
Feature 1 =  12  Feature 2 =  48  Feature 3 =  32
Train Error on class 0 0.625
Train Error on class 1 0.275
Test Error on class 0 0.641025641025641
Test Error on class 1 0.36363636363636365
Test Error Total 0.413953488372093
```

2.4 Non Linear SVM 4-feature selection sequential forward search

In [19]:

```
e
,feature4_svmNL,e0_train_svmNL_4,e1_train_svmNL_4,e0_test_svmNL_4,e1_test_svmNL_4,e1_total_svmNL_4
= fs_sfs4(feature1_svmNL,feature2_svmNL,feature3_svmNL,X,Y,clf_svmNL,

x0,x0_test,y0,y0_test,x1,x1_test,y1,y1_test,X_test,Y_test)
print('Resubstitution error = ',e,'\nFeature 1 = ', feature1_svmNL,' Feature 2 = ',feature2_svmNL,
' Feature 3 = ',feature3_svmNL,' Feature 4 = ',feature4_svmNL)
print("Train Error on class 0",e0_train_svmNL_4)
print("Train Error on class 1",e1_train_svmNL_4)
print("Test Error on class 0",e0_test_svmNL_4)
print("Test Error on class 1",e1_test_svmNL_4)
print("Test Error Total",e1_total_svmNL_4)
```

```
Resubstitution error =  0.15000000000000002
Feature 1 =  12  Feature 2 =  48  Feature 3 =  32  Feature 4 =  1
Train Error on class 0 0.09999999999999998
Train Error on class 1 0.25
Test Error on class 0 0.23076923076923073
Test Error on class 1 0.48863636363636365
Test Error Total 0.4418604651162791
```

2.5 Non Linear SVM 5-feature selection sequential forward search

In [20]:

```
e
,feature5_svmNL,e0_train_svmNL_5,e1_train_svmNL_5,e0_test_svmNL_5,e1_test_svmNL_5,e1_total_svmNL_5
= fs_sfs5(feature1_svmNL,feature2_svmNL,feature3_svmNL,feature4_svmNL,X,Y,clf_svmNL,

x0,x0_test,y0,y0_test,x1,x1_test,y1,y1_test,X_test,Y_test)
print('Resubstitution error = ',e,'\nFeature 1 = ', feature1_svmNL,' Feature 2 = ',feature2_svmNL,
' Feature 3 = ',feature3_svmNL,' Feature 4 = ',feature4_svmNL,' Feature 5 = ',feature5_svmNL)
print("Train Error on class 0",e0_train_svmNL_5)
print("Train Error on class 1",e1_train_svmNL_5)
print("Test Error on class 0",e0_test_svmNL_5)
print("Test Error on class 1",e1_test_svmNL_5)
print("Test Error Total",e1_total_svmNL_5)
```

```
Resubstitution error =  0.13749999999999996
Feature 1 =  12  Feature 2 =  48  Feature 3 =  32  Feature 4 =  1  Feature 5 =  24
Train Error on class 0 0.125
Train Error on class 1 0.25
```

```
Test Error on class 0 0.2564102564102564
Test Error on class 1 0.44318181818181823
Test Error Total 0.40930232558139534
```

3 Neural Network

3.1 Neural Network No feature selection

In [21]:

```
clf_NN = MLPClassifier((5,5),solver = 'lbfgs',activation='relu',
                       learning_rate= 'constant',learning_rate_init=0.001,
                      random_state=0,alpha=0.0001,)
Xn_train, Xn_test, yn_train, yn_test = train_test_split(X,Y, test_size=0.0, random_state=0)
e = 1 - clf_NN.fit(Xn_train, yn_train).score(Xn_train, yn_train)
e0_train_NN_n = 1 - clf_NN.score(x0, y0)
e1_train_NN_n = 1 - clf_NN.score(x1, y1)
e0_test_NN_n  = 1 - clf_NN.score(x0_test, y0_test)
e1_test_NN_n  = 1 - clf_NN.score(x1_test, y1_test)
e1_total_NN_n = 1 - clf_NN.score(X_test, Y_test)
print('Resubstitution error = ',e)
print("Train Error on class 0",e0_train_NN_n)
print("Train Error on class 1",e1_train_NN_n)
print("Test Error on class 0",e0_test_NN_n)
print("Test Error on class 1",e1_test_NN_n)
print("Test Error Total",e1_total_NN_n)
```

```
Resubstitution error =  0.0
Train Error on class 0 0.0
Train Error on class 1 0.0
Test Error on class 0 0.33333333333333337
Test Error on class 1 0.34090909090909094
Test Error Total 0.3395348837209302
```

3.2 Neural Network 2-feature selection extraction search

In [22]:

```
e,feature1_nn,feature2_nn,e0_train_nn_2,e1_train_nn_2,e0_test_nn_2,e1_test_nn_2,e1_total_nn_2  = fs
_es(X,Y,clf_NN,x0,x0_test,y0,
                                                                                    y0_test,x1,x1_test
,y1_test,X_test,Y_test)
print('Resubstitution error = ',e,'\nFeature 1 = ', feature1_nn,' Feature 2 = ',feature2_nn)
print("Train Error on class 0",e0_train_nn_2)
print("Train Error on class 1",e1_train_nn_2)
print("Test Error on class 0",e0_test_nn_2)
print("Test Error on class 1",e1_test_nn_2)
print("Test Error Total",e1_total_nn_2)
```

```
Resubstitution error =  0.15000000000000002
Feature 1 =  59  Feature 2 =  67
Train Error on class 0 0.42500000000000004
Train Error on class 1 0.22499999999999998
Test Error on class 0 0.41025641025641024
Test Error on class 1 0.3977272727272727
Test Error Total 0.4
```

Neural Network 3-feature selection sequential forward search

In [23]:

```
e,feature3_nn,e0_train_nn_3,e1_train_nn_3,e0_test_nn_3,e1_test_nn_3,e1_total_nn_3 =
fs_sfs3(feature1_nn,feature2_nn,X,Y,clf_NN,

x0,x0_test,y0,y0_test,x1,x1_test,y1,y1_test,X_test,Y_test)
print('Resubstitution error = ',e,'\nFeature 1 = ', feature1_nn,' Feature 2 = ',feature2_nn,'
Feature 3 = ',feature3_nn)
print("Train Error on class 0",e0_train_nn_3)
```

```
print("Train Error on class 1",e1_train_nn_3)
print("Test Error on class 0",e0_test_nn_3)
print("Test Error on class 1",e1_test_nn_3)
print("Test Error Total",e1_total_nn_3)
```

```
Resubstitution error =  0.125
Feature 1 =  59  Feature 2 =  67  Feature 3 =  51
Train Error on class 0 0.375
Train Error on class 1 0.22499999999999998
Test Error on class 0 0.3846153846153846
Test Error on class 1 0.38636363636363635
Test Error Total 0.38604651162790693
```

Neural Network 4-feature selection sequential forward search

In [24]:

```
e,feature4_nn,e0_train_nn_4,e1_train_nn_4,e0_test_nn_4,e1_test_nn_4,e1_total_nn_4 =
fs_sfs4(feature1_nn,feature2_nn,feature3_nn,X,Y,clf_NN,

x0,x0_test,y0,y0_test,x1,x1_test,y1,y1_test,X_test,Y_test)
print('Resubstitution error = ',e,'\nFeature 1 = ', feature1_nn,' Feature 2 = ',feature2_nn,'
Feature 3 = ',feature3_nn,' Feature 4 = ',feature4_nn)
print("Train Error on class 0",e0_train_nn_4)
print("Train Error on class 1",e1_train_nn_4)
print("Test Error on class 0",e0_test_nn_4)
print("Test Error on class 1",e1_test_nn_4)
print("Test Error Total",e1_total_nn_4)
```

```
Resubstitution error =  0.03749999999999998
Feature 1 =  59  Feature 2 =  67  Feature 3 =  51  Feature 4 =  19
Train Error on class 0 0.25
Train Error on class 1 0.44999999999999996
Test Error on class 0 0.3076923076923077
Test Error on class 1 0.46590909090909094
Test Error Total 0.4372093023255814
```

Neural network 5-feature selection sequential forward search

In [25]:

```
e,feature5_nn,e0_train_nn_5,e1_train_nn_5,e0_test_nn_5,e1_test_nn_5,e1_total_nn_5 =
fs_sfs5(feature1_nn,feature2_nn,feature3_nn,feature4_nn,X,Y,clf_NN,

x0,x0_test,y0,y0_test,x1,x1_test,y1,y1_test,X_test,Y_test)
print('Resubstitution error = ',e,'\nFeature 1 = ', feature1_nn,' Feature 2 = ',feature2_nn,'
Feature 3 = ',feature3_nn,' Feature 4 = ',feature4_nn,' Feature 5 = ',feature5_nn)
print("Train Error on class 0",e0_train_nn_5)
print("Train Error on class 1",e1_train_nn_5)
print("Test Error on class 0",e0_test_nn_5)
print("Test Error on class 1",e1_test_nn_5)
print("Test Error Total",e1_total_nn_5)
```

```
Resubstitution error =  0.0
Feature 1 =  59  Feature 2 =  67  Feature 3 =  51  Feature 4 =  19  Feature 5 =  30
Train Error on class 0 0.15000000000000002
Train Error on class 1 0.19999999999999996
Test Error on class 0 0.3846153846153846
Test Error on class 1 0.4147727272727273
Test Error Total 0.40930232558139534
```

5					Train			Test		
			Feature	Selected Features	Total (Resub)	class 0 error	class 1 error	class 0 error	class 1 error	Total
Classifier	Linear SVM	Without Selection	No	NO	0.05	0.07	0.02	0.38	0.32	0.33
		Exhaustive	2	3,19	0.212	0.32	0.25	0.41	0.36	0.37
		Sequential	3	27	0.19	0.05	0.44	0.15	0.57	0.49
		Forward	4	5	0.18	0.05	0.42	0.12	0.54	0.46
		Search	5	44	0.18	0.09	0.375	0.15	0.52	0.46
	Non Linear Svm	Without Selection	No	NO	0.0625	0.09	0.02	0.38	0.33	0.34
		Exhaustive	2	12,48	0.21	0.32	0.42	0.17	0.46	0.4
		Sequential	3	32	0.17	0.62	0.27	0.64	0.36	0.41
		Forward	4	1	0.15	0.09	0.25	0.23	0.48	0.44
		Search	5	5	0.13	0.12	0.25	0.25	0.44	0.4
	Neural Network	Without Selection	No	No	0	0	0	0.33	0.34	0.34
		Exhaustive	2	59,67	0.1	0.42	0.22	0.41	0.39	0.4
		Sequential	3	51	0.12	0.37	0.22	0.38	0.38	0.38
		Forward	4	19	0.03	0.25	0.44	0.3	0.46	0.43
		Search	5	5	0	0.15	0.19	0.38	0.41	0.4

The resubstitution error seems to be low for almost all the cases. The specific 0 and 1 class resub. error is also low for almost all the cases. We can also observe no majority has been observed for feature selections amongst all three classifiers. There is also an obvious interpretation that even for class 0 and 1 the test error is higher than the train error. The neural networks 0 and 1 class test error seems to be almost equal but for both the SVM the 1 class error is high for most of the cases which can be because of non-equal data sizes for class 1 and 0 in the test set. Based on the error rates no classifier seems to be capable of making good predictions and the error rates seems to oscillate with increasing and decreasing features i.e. a clear trend of error w.r.t to number of features cannot be observed with any classifier. Hence, the sequential forward search can be further used to optimize the number of features to be used. With small testing and training data, we can see that the test total error is least when no feature is selected but this might not be the case with large training data or if try to find different combinations of larger features. This can also be seen as a fact that when no feature is selected, the training resubstitution error is the lowest but it quite high for testing set; in other words lacking consistency.

Question2

In [1]:

```
import os
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import cv2
import ntpath
import keras
import sys
from sklearn import metrics
from sklearn import preprocessing
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.svm import LinearSVC,SVC,NuSVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.preprocessing import StandardScaler
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input,VGG16
from keras.models import Sequential,load_model,Model
from keras.utils import to_categorical
%matplotlib inline
```

```
Using TensorFlow backend.
```

In [2]:

```
datadir = 'micrograph'
columns = ['micrograph_id','path','micron_bar','micron_bar_units','micron_bar_px',
           'magnification','detector','sample_key','contributor_key','primary_microconstituent']
data = pd.read_csv('micrograph.csv')
data.head()
```

Out[2]:

	micrograph_id	path	micron_bar	micron_bar_units	micron_bar_px	magnification	detector	sample_key	contributor_key
0	1	micrograph1.tif	5	um	129	4910x	SE	42.0	2
1	2	micrograph2.tif	10	um	103	1964X	SE	18.0	2
2	4	micrograph4.tif	10	um	129	NaN	SE	35.0	2
3	5	micrograph5.tif	5	um	129	4910X	SE	10.0	2
4	6	micrograph6.tif	20	um	124	1178X	SE	29.0	2

In [3]:

```
p = 0
n = 0
s = 0
Xp = [];Xpt = [];
Yp = [];Ypt = [];
Xn = [];Xnt = [];
Yn = [];Ynt = [];
Xs = [];Xst = [];
Ys = [];Yst = [];
Xmt= []
Ymt = []
data_dir = 'micrograph'
for i in range(len(data)):
    indexed_data = data.iloc[i]
    if(indexed_data[9]=='pearlite'and p < 100):
        p = p+1
        x = indexed_data[1]
        Xp.append(os.path.join(data_dir, x.strip()))
        Yp.append(indexed_data[9])

    if(indexed_data[9]=='pearlite'and p >= 100):
        x = indexed_data[1]
        Xpt.append(os.path.join(data_dir, x.strip()))
        Ypt.append(indexed_data[9])
```

```
Xp = np.asarray(Xp)
Yp = np.asarray(Yp)

for i in range(len(data)):
    indexed_data = data.iloc[i]
    if(indexed_data[9]=='network'and n < 100):
        n = n+1
        x = indexed_data[1]
        Xn.append(os.path.join(data_dir, x.strip()))
        Yn.append(indexed_data[9])
    if(indexed_data[9]=='network'and n >= 100):
        x = indexed_data[1]
        Xnt.append(os.path.join(data_dir, x.strip()))
        Ynt.append(indexed_data[9])
Xn = np.asarray(Xn)
Yn = np.asarray(Yn)

for i in range(len(data)):
    indexed_data = data.iloc[i]
    if(indexed_data[9]=='spheroidite'and s < 100):
        s = s+1
        x = indexed_data[1]
        Xs.append(os.path.join(data_dir, x.strip()))
        Ys.append(indexed_data[9])
    if(indexed_data[9]=='spheroidite'and s >= 100):
        x = indexed_data[1]
        Xst.append(os.path.join(data_dir, x.strip()))
        Yst.append(indexed_data[9])

Xs = np.asarray(Xs)
Ys = np.asarray(Ys)

for i in range(len(data)):
    indexed_data = data.iloc[i]
    if(indexed_data[9]=='spheroidite'or indexed_data[9]=='network' or
       indexed_data[9]=='pearlite' or indexed_data[9]=='martensite' or
       indexed_data[9]=='widmanstatten'):
        pass
    else:
        x = indexed_data[1]
        Xmt.append(os.path.join(data_dir, x.strip()))
        Ymt.append(indexed_data[9])
```

In [5]:

```
def datagen(Xp,Xs,Xn,Yp,Ys,Yn,Xpt,Xst,Xnt,Ypt,Yst,Ynt):
    image_path = []
    label = []


    X1 = np.concatenate((Xp,Xs))
    Y1 = np.concatenate((Yp,Ys))
    X2 = np.concatenate((Xp,Xn))
    Y2 = np.concatenate((Yp,Yn))
    X3 = np.concatenate((Xs,Xn))
    Y3 = np.concatenate((Ys,Yn))


    Xr = np.concatenate((Xp,Xs,Xn))
    Yr = np.concatenate((Yp,Ys,Yn))

    X1t = np.concatenate((Xpt,Xst))
    Y1t = np.concatenate((Ypt,Yst))
    X2t = np.concatenate((Xpt,Xnt))
    Y2t = np.concatenate((Ypt,Ynt))
    X3t = np.concatenate((Xst,Xnt))
    Y3t = np.concatenate((Yst,Ynt))

    Xe = np.concatenate((Xpt,Xst,Xnt))
    Ye = np.concatenate((Ypt,Yst,Ynt))


    return X1,Y1,X2,Y2,X3,Y3,Xr,Yr,X1t,Y1t,X2t,Y2t,X3t,Y3t,Xe,Ye

X1,Y1,X2,Y2,X3,Y3,Xr,Yr,X1t,Y1t,X2t,Y2t,X3t,Y3t,Xe,Ye = datagen(Xp,Xs,Xn,Yp,Ys,Yn,Xpt,Xst,Xnt,Ypt,Y
st,Ynt)
```

In [6]:

```
#le = preprocessing.LabelEncoder()
#le.fit(['pearlite','spheroidite','network'])
#Y1 = le.transform(Y1)
##Y1 = to_categorical(Y1)
##le.fit(['pearlite',])
#Y2 = le.transform(Y2)
##Y2 = to_categorical(Y2)
##le.fit(['spheroidite','network'])
#Y3 = le.transform(Y3)
##Y3 = to_categorical(Y3)
#Y1t_ = le.transform(Yt)
#le.fit((['pearlite+spheroidite',]))
```

In [7]:

```
def pre_processor(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    x = image.img_to_array(img)
    x = x[0:484,:,:]
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    return x

def generator(X):
    x = []
    for i in range(len(X)):
        x.append(pre_processor(X[i]))
    return x
```

In [8]:

```
X1 = generator(X1)
X2 = generator(X2)
X3 = generator(X3)
Xr = generator(Xr)

X1t = generator(X1t)
X2t = generator(X2t)
X3t = generator(X3t)

Xe = generator(Xe)
```

In [33]:

```
#single labels
Xpt = generator(Xpt)
Xst = generator(Xst)
Xnt = generator(Xnt)
```

In [9]:

```
base_model = VGG16(weights='imagenet', include_top=False)
```

In [10]:

```
def exctractor(X,model):
    x = []
    for i in range(len(X)):
        x.append(model.predict(X[i]))
    return np.array(x)
```

In [ ]:

```
model_M1 = Model(inputs=base_model.input, outputs=base_model.get_layer('block1_pool').output)
model_M2 = Model(inputs=base_model.input, outputs=base_model.get_layer('block2_pool').output)
model_M3 = Model(inputs=base_model.input, outputs=base_model.get_layer('block3_pool').output)
model_M4 = Model(inputs=base_model.input, outputs=base_model.get_layer('block4_pool').output)
```

```
model_M5 = Model(inputs=base_model.input, outputs=base_model.get_layer('block5_pool').output)
```

In [11]:

```
X1M1 = exctractor(X1,model_M1)
X1M2 = exctractor(X1,model_M2)
X1M3 = exctractor(X1,model_M3)
X1M4 = exctractor(X1,model_M4)
X1M5 = exctractor(X1,model_M5)
```

In [12]:

```
X2M1 = exctractor(X2,model_M1)
X2M2 = exctractor(X2,model_M2)
X2M3 = exctractor(X2,model_M3)
X2M4 = exctractor(X2,model_M4)
X2M5 = exctractor(X2,model_M5)
```

In [13]:

```
X3M1 = exctractor(X3,model_M1)
X3M2 = exctractor(X3,model_M2)
X3M3 = exctractor(X3,model_M3)
X3M4 = exctractor(X3,model_M4)
X3M5 = exctractor(X3,model_M5)
```

In [47]:

```
X_train1_ = exctractor(Xr,model_M1)
X_train2_ = exctractor(Xr,model_M2)
X_train3_ = exctractor(Xr,model_M3)
X_train4_ = exctractor(Xr,model_M4)
X_train5_ = exctractor(Xr,model_M5)
```

In [16]:

```
def reshapper(X):
    x = []
    for i in range(len(X)):
        v = []
        a = X[i].reshape(X.shape[4],-1)
        for c in range(len(a)):
            v.append(sum(a[c])/(X.shape[2]*X.shape[3]))
        x.append(np.array(v).T)
    return np.array(x)
```

In [17]:

```
X1M1_ = reshapper(X1M1)
X1M2_ = reshapper(X1M2)
X1M3_ = reshapper(X1M3)
X1M4_ = reshapper(X1M4)
X1M5_ = reshapper(X1M5)
```

In [18]:

```
X2M1_ = reshapper(X2M1)
X2M2_ = reshapper(X2M2)
X2M3_ = reshapper(X2M3)
X2M4_ = reshapper(X2M4)
X2M5_ = reshapper(X2M5)
```

In [19]:

```
X3M1_ = reshapper(X3M1)
X3M2_ = reshapper(X3M2)
X3M3_ = reshapper(X3M3)
X3M4_ = reshapper(X3M4)
X3M5_ = reshapper(X3M5)
```

In [48]:

```
X_train1 = reshapper(X_train1_)
X_train2 = reshapper(X_train2_)
X_train3 = reshapper(X_train3_)
X_train4 = reshapper(X_train4_)
X_train5 = reshapper(X_train5_)
```

In [52]:

```
clf1 = SVC(gamma ='auto' ,kernel ='rbf', C = 10)

scores_X1M1 = cross_val_score(clf1, X1M1_, Y1, cv=10)
scores_X1M2 = cross_val_score(clf1, X1M2_, Y1, cv=10)
scores_X1M3 = cross_val_score(clf1, X1M3_, Y1, cv=10)
scores_X1M4 = cross_val_score(clf1, X1M4_, Y1, cv=10)
scores_X1M5 = cross_val_score(clf1, X1M5_, Y1, cv=10)
print('CV error on layer 1',1-sum(scores_X1M1)/10)
print('CV error on layer 2',1-sum(scores_X1M2)/10)
print('CV error on layer 3',1-sum(scores_X1M3)/10)
print('CV error on layer 4',1-sum(scores_X1M4)/10)
print('CV error on layer 5',1-sum(scores_X1M5)/10)
```

```
CV error on layer 1 0.49
CV error on layer 2 0.49
CV error on layer 3 0.49
CV error on layer 4 0.5
CV error on layer 5 0.13000000000000012
```

In [53]:

```
clf2 = SVC(gamma ='auto' ,kernel ='rbf', C = 10)
scores_X2M1 = cross_val_score(clf2, X2M1_, Y2, cv=10)
scores_X2M2 = cross_val_score(clf2, X2M2_, Y2, cv=10)
scores_X2M3 = cross_val_score(clf2, X2M3_, Y2, cv=10)
scores_X2M4 = cross_val_score(clf2, X2M4_, Y2, cv=10)
scores_X2M5 = cross_val_score(clf2, X2M5_, Y2, cv=10)
print('CV error on layer 1',1-sum(scores_X2M1)/10)
print('CV error on layer 2',1-sum(scores_X2M2)/10)
print('CV error on layer 3',1-sum(scores_X2M3)/10)
print('CV error on layer 4',1-sum(scores_X2M4)/10)
print('CV error on layer 5',1-sum(scores_X2M5)/10)
```

```
CV error on layer 1 0.495
CV error on layer 2 0.49
CV error on layer 3 0.4850000000000001
CV error on layer 4 0.495
CV error on layer 5 0.019999999999999907
```

In [54]:

```
clf3 = SVC(gamma ='auto' ,kernel ='rbf', C = 10)
scores_X3M1 = cross_val_score(clf3, X3M1_, Y3, cv=10)
scores_X3M2 = cross_val_score(clf3, X3M2_, Y3, cv=10)
scores_X3M3 = cross_val_score(clf3, X3M3_, Y3, cv=10)
scores_X3M4 = cross_val_score(clf3, X3M4_, Y3, cv=10)
scores_X3M5 = cross_val_score(clf3, X3M5_, Y3, cv=10)
print('CV error on layer 1',1-sum(scores_X3M1)/10)
print('CV error on layer 2',1-sum(scores_X3M2)/10)
print('CV error on layer 3',1-sum(scores_X3M3)/10)
print('CV error on layer 4',1-sum(scores_X3M4)/10)
print('CV error on layer 5',1-sum(scores_X3M5)/10)
```

```
CV error on layer 1 0.5
CV error on layer 2 0.49
CV error on layer 3 0.4850000000000001
CV error on layer 4 0.4850000000000001
CV error on layer 5 0.05500000000000005
```

In [55]:

```
clf = SVC(gamma ='auto' ,kernel ='rbf', C = 10)
clfm = OneVsOneClassifier(clf)
scores_Xtest1 = cross_val_score(clfm, X_train1, Yr, cv=10)
scores_Xtest2 = cross_val_score(clfm, X_train2, Yr, cv=10)
scores_Xtest3 = cross_val_score(clfm, X_train3, Yr, cv=10)
scores_Xtest4 = cross_val_score(clfm, X_train4, Yr, cv=10)
scores_Xtest5 = cross_val_score(clfm, X_train5, Yr, cv=10)
print('CV error on layer 1',1-sum(scores_Xtest1)/10)
print('CV error on layer 2',1-sum(scores_Xtest2)/10)
print('CV error on layer 3',1-sum(scores_Xtest3)/10)
print('CV error on layer 4',1-sum(scores_Xtest4)/10)
print('CV error on layer 5',1-sum(scores_Xtest5)/10)
```

```
CV error on layer 1 0.6599999999999999
CV error on layer 2 0.6533333333333333
CV error on layer 3 0.6499999999999999
CV error on layer 4 0.6566666666666666
CV error on layer 5 0.1233333333333334
```

The final Maxpool layer gives the least error and hence we will use only the final layer i.e Model5 or block_pool5

In [15]:

```
X_test1 = exctractor(X1t,model_M5)
X_test2 = exctractor(X2t,model_M5)
X_test3 = exctractor(X3t,model_M5)
X_test = exctractor(Xe,model_M5)
```

In [23]:

```
X_test1 = reshapper(X_test1)
X_test2 = reshapper(X_test2)
X_test3 = reshapper(X_test3)
X_test = reshapper(X_test)
```

In [35]:

```
Xpt = exctractor(Xpt,model_M5)
Xst = exctractor(Xst,model_M5)
Xnt = exctractor(Xnt,model_M5)
```

In [36]:

```
Xpt = reshapper(Xpt)
Xst = reshapper(Xst)
Xnt = reshapper(Xnt)
```

In [50]:

```
#test with 1 labels test set


e11 = clf1.fit(X1M5_, Y1).score(Xpt,Ypt)
e12 = clf1.fit(X1M5_, Y1).score(Xst,Yst)

e21 = clf1.fit(X2M5_, Y2).score(Xpt,Ypt)
e22 = clf1.fit(X2M5_, Y2).score(Xnt,Ynt)


e31 = clf3.fit(X3M5_, Y3).score(Xnt,Ynt)
e32 = clf3.fit(X3M5_, Y3).score(Xst,Yst)

ept = clfm.fit(X_train5,Yr).score(Xpt,Ypt)
est = clfm.fit(X_train5,Yr).score(Xst,Yst)
ent = clfm.fit(X_train5,Yr).score(Xnt,Ynt)
```

```
print('Error on class 1 (pearlite)',1-e11)
print('Error on class 1 (spherodite)',1-e12)

print('Error on class 2 (pearlite)',1-e21)
print('Error on class 2 (network)',1-e22)

print('Error on class 3 (network)',1-e31)
print('Error on class 3 (spherodite)',1-e32)

print('Error on multiclass (pearlite)',1-ept)
print('Error on multiclass (spherodite)',1-est)
print('Error on multiclass (network)',1-ent)
```

```
Error on class 1 (pearlite) 0.07999999999999996
Error on class 1 (spherodite) 0.20727272727272728
Error on class 2 (pearlite) 0.040000000000000036
Error on class 2 (network) 0.06194690265486724
Error on class 3 (network) 0.08849557522123896
Error on class 3 (spherodite) 0.032727272727272716
Error on multiclass (pearlite) 0.07999999999999996
Error on multiclass (spherodite) 0.24
Error on multiclass (network) 0.09734513274336287
```

In [51]:

```
#test with 2 and 3 labels test set


e1 = clf1.fit(X1M5_, Y1).score(X_test1,Y1t)
e2 = clf2.fit(X2M5_, Y2).score(X_test2,Y2t)
e3 = clf3.fit(X3M5_, Y3).score(X_test3,Y3t)

et = clfm.fit(X_train5,Yr).score(X_test,Ye)

epst = clfm.fit(X_train,Yr).score(X_test1,Y1t)
epnt = clfm.fit(X_train,Yr).score(X_test2,Y2t)
esnt = clfm.fit(X_train,Yr).score(X_test3,Y3t)
print('Error on class 1 (pearlite & spherodite)',1-e1)
print('Error on class 2 (pearlite & network)',1-e2)
print('Error on class 3 (spherodite & network)',1-e3)
print('Error on multiclass (all)',1-et)
print('Error on multiclass (pearlite & spherodite)',1-epst)
print('Error on multiclass (pearlite & network)',1-epnt)
print('Error on multiclass (spherodite & network)',1-esnt)
```

```
Error on class 1 (pearlite & spherodite) 0.19666666666666666
Error on class 2 (pearlite & network) 0.05797101449275366
Error on class 3 (spherodite & network) 0.0489690721649485
Error on multiclass (all) 0.19128329297820823
Error on multiclass (pearlite & spherodite) 0.22666666666666668
Error on multiclass (pearlite & network) 0.09420289855072461
Error on multiclass (spherodite & network) 0.19845360824742264
```

VGG16 is one of the most sophisticated CNN architecture which is widely used in machine learning for various problems like object detection, image classification, etc.

It is trained on millions of images, in other words the network has weights which are ideally suited for feature extractions. When an image is passed through VGG model, with every consecutive layer, more and more image features which are crucial for the recognition of that image are highlighted.

In our case, as it is also tested with cross-validation score, we can confidently say that the 5^{th} max-pooling layer generates the features with almost ideal requirements. The test error also seems reasonable when checked with unseen images which were passed through the 5^{th} maxpool layer.

The test error can be further reduced if feature selection is carried out on the features extracted from the 5^{th} maxpool layer.

CV error on layer 1 class 1 0.49 Class1(pearlite & spherodite)

CV error on layer 2 class 1 0.49

CV error on layer 3 class 1 0.49

CV error on layer 4 class 1 0.5

CV error on layer 5 class 1 0.13

CV error on layer 1 class 2 0.495 Class2(pearlite & network)

CV error on layer 2 class 2 0.49

CV error on layer 3 class 2 0.485

CV error on layer 4 class 2 0.495

CV error on layer 5 class 2 0.019

CV error on layer 1 class 3 0.5 Class 3 (spherodite & network)

CV error on layer 2 class 3 0.49

CV error on layer 3 class 3 0.48

CV error on layer 4 class 3 0.485

CV error on layer 5 class 3 0.019

CV error on layer 1 multiclass 0.659

CV error on layer 2 multiclass 0.653

CV error on layer 3 multiclass 0.649

CV error on layer 4 multiclass 0.656

CV error on layer 5 multiclass 0.123

#Test with 1 labels test set

Error on class 1 (pearlite) 0.07999999999999996

Error on class 1 (spherodite) 0.20727272727272728

Error on class 2 (pearlite) 0.040000000000000036

Error on class 2 (network) 0.06194690265486724

Error on class 3 (network) 0.08849557522123896

Error on class 3 (spherodite) 0.032727272727272716

Error on multiclass (pearlite) 0.07999999999999996

Error on multiclass (spherodite) 0.24

Error on multiclass (network) 0.09734513274336287

#Test with 2 and 3 labels test set

Error on class 1 (pearlite & spherodite) 0.19666666666666666

Error on class 2 (pearlite & network) 0.05797101449275366

Error on class 3 (spherodite & network) 0.0489690721649485

Error on multiclass (all) 0.19128329297820823

Error on multiclass (pearlite & spherodite) 0.22666666666666668

Error on multiclass (pearlite & network) 0.09420289855072461

Error on multiclass (spherodite & network) 0.19845360824742264

The error is reasonably low for all the classifiers.