COMPUTER PROJECT 2

PATTERN RECOGNITION

(ECEN 649)

SPRING 2017

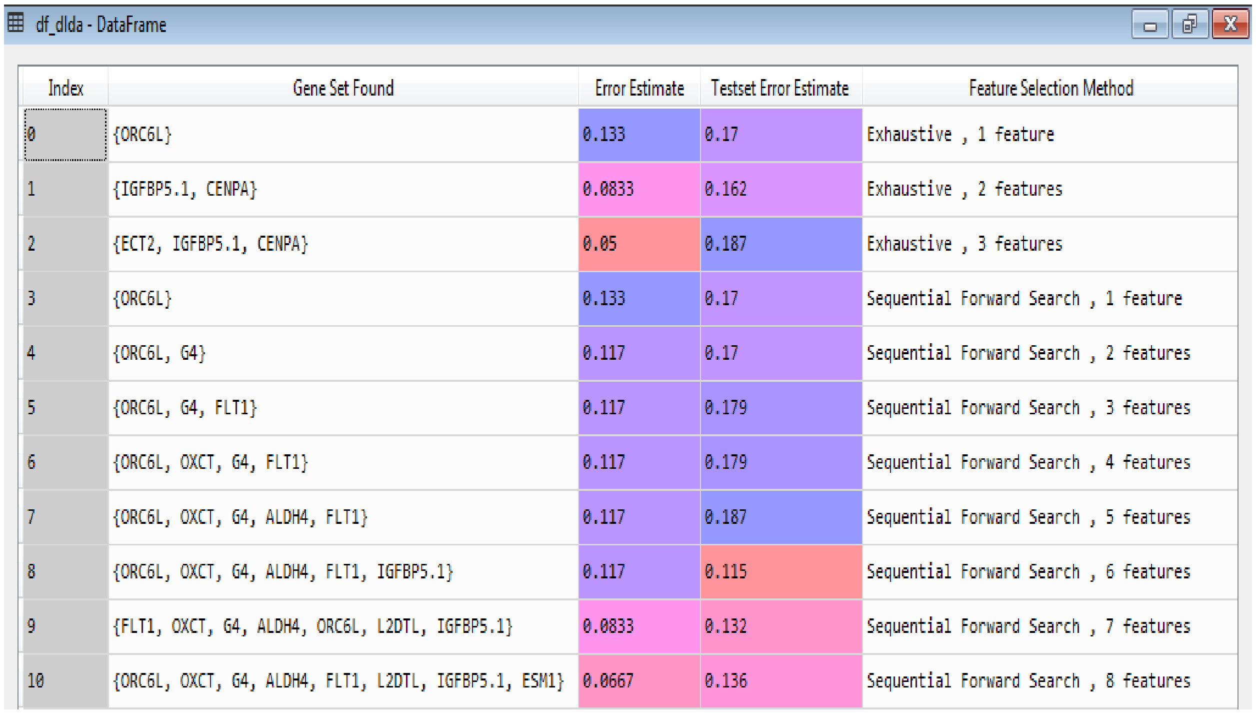
- ISHAN D KHURJEKAR

UIN : 725007495

DEPT OF ELECTRICAL ENGINEERING

Results :

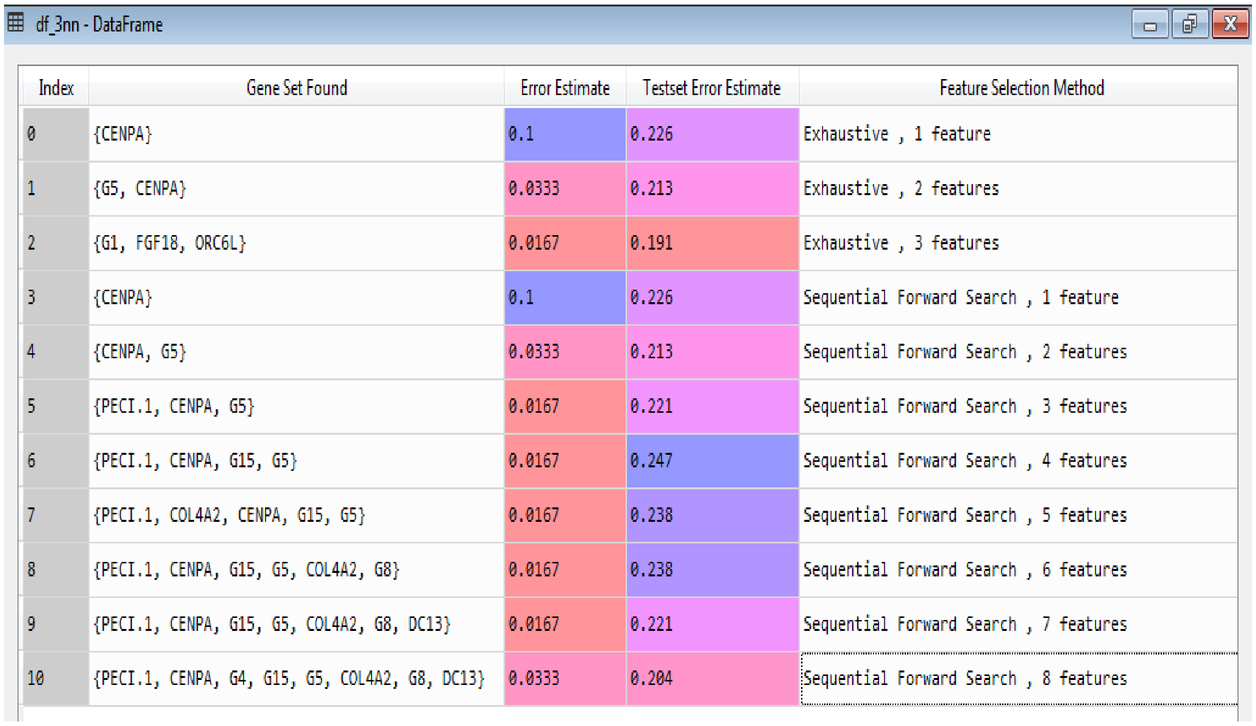
DLDA :



df_dlda - DataFrame

Index	Gene Set Found	Error Estimate	Testset Error Estimate	Feature Selection Method
0	{ORC6L}	0.133	0.17	Exhaustive , 1 feature
1	{IGFBP5.1, CENPA}	0.0833	0.162	Exhaustive , 2 features
2	{ECT2, IGFBP5.1, CENPA}	0.05	0.187	Exhaustive , 3 features
3	{ORC6L}	0.133	0.17	Sequential Forward Search , 1 feature
4	{ORC6L, G4}	0.117	0.17	Sequential Forward Search , 2 features
5	{ORC6L, G4, FLT1}	0.117	0.179	Sequential Forward Search , 3 features
6	{ORC6L, OXCT, G4, FLT1}	0.117	0.179	Sequential Forward Search , 4 features
7	{ORC6L, OXCT, G4, ALDH4, FLT1}	0.117	0.187	Sequential Forward Search , 5 features
8	{ORC6L, OXCT, G4, ALDH4, FLT1, IGFBP5.1}	0.117	0.115	Sequential Forward Search , 6 features
9	{FLT1, OXCT, G4, ALDH4, ORC6L, L2DTL, IGFBP5.1}	0.0833	0.132	Sequential Forward Search , 7 features
10	{ORC6L, OXCT, G4, ALDH4, FLT1, L2DTL, IGFBP5.1, ESM1}	0.0667	0.136	Sequential Forward Search , 8 features

3NN :



df_3nn - DataFrame

Index	Gene Set Found	Error Estimate	Testset Error Estimate	Feature Selection Method
0	{CENPA}	0.1	0.226	Exhaustive , 1 feature
1	{G5, CENPA}	0.0333	0.213	Exhaustive , 2 features
2	{G1, FGF18, ORC6L}	0.0167	0.191	Exhaustive , 3 features
3	{CENPA}	0.1	0.226	Sequential Forward Search , 1 feature
4	{CENPA, G5}	0.0333	0.213	Sequential Forward Search , 2 features
5	{PECI.1, CENPA, G5}	0.0167	0.221	Sequential Forward Search , 3 features
6	{PECI.1, CENPA, G15, G5}	0.0167	0.247	Sequential Forward Search , 4 features
7	{PECI.1, COL4A2, CENPA, G15, G5}	0.0167	0.238	Sequential Forward Search , 5 features
8	{PECI.1, CENPA, G15, G5, COL4A2, G8}	0.0167	0.238	Sequential Forward Search , 6 features
9	{PECI.1, CENPA, G15, G5, COL4A2, G8, DC13}	0.0167	0.221	Sequential Forward Search , 7 features
10	{PECI.1, CENPA, G4, G15, G5, COL4A2, G8, DC13}	0.0333	0.204	Sequential Forward Search , 8 features

Code :

1.) Code for exhaustive search

```
import itertools
import pandas as pd
import numpy as np
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn import neighbors
#columns =['Gene Set Found','Error Estimate','Testset Error Estimate ']
#index=range(11)
#df = pd.DataFrame(index=index, columns=columns)
#df = df.fillna(0)
#Read relevant data as data frame
training = pd.read_table('C:\Training_Data.txt')
n=training.columns
testing  = pd.read_table('C:\Testing_Data.txt')
feature_raw = np.array(training.iloc[:,0:71])
output_vector = np.array(training.iloc[:,-1])
feature_test = np.array(testing.iloc[:,0:71])
output_test = np.array(testing.iloc[:,-1])
feature_size = 3
#function for creating all possible subsets of required size
def findsubsets(S,m):
    return set(itertools.combinations(S, m))
feature_space = findsubsets(range(71),feature_size)
feature_space = np.array(list(feature_space))
```

```
b=[]
#find the optimal feature
for ip in feature_space:
    x = feature_raw[:,ip].reshape((feature_raw.shape[0],feature_size))
    clf = LinearDiscriminantAnalysis(solver='lsqr', shrinkage=1).fit(x,output_vector)
    #clf = neighbors.KNeighborsClassifier(3).fit(x,output_vector)
    a=clf.score(x, output_vector)
    b.append(1-a)
b = np.array(b)
error_estimate = min(b)
index = np.argmin(b)
ip_optimal = feature_raw[:,feature_space[index]].reshape((feature_raw.shape[0],feature_size))
#clf_optimal = neighbors.KNeighborsClassifier(3).fit(ip_optimal,output_vector)
clf_optimal = LinearDiscriminantAnalysis(solver='lsqr', shrinkage=1).fit(ip_optimal,output_vector)
x = feature_test[:,feature_space[index]].reshape((feature_test.shape[0],feature_size))
output_pred = clf_optimal.predict(x)
acc = float((output_test == output_pred).sum()) / output_pred.shape[0]
testset_error = 1-acc
```

2.) Sequential Forward Search

```
import pandas as pd

import numpy as np

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

from sklearn.neighbors import KNeighborsClassifier

from mlxtend.feature_selection import SequentialFeatureSelector as SFS

training = pd.read_table('C:\Training_Data.txt')

n=training.columns

testing  = pd.read_table('C:\Testing_Data.txt')

feature_raw = np.array(training.iloc[:,0:71])

output_vector = np.array(training.iloc[:,-1])

feature_test = np.array(testing.iloc[:,0:71])

output_test = np.array(testing.iloc[:,-1])

#lda = LinearDiscriminantAnalysis(solver='lsqr', shrinkage=1)

knn = KNeighborsClassifier(n_neighbors=3)

sfs1 = SFS(knn, k_features=1, forward=True, floating=False, verbose=2,scoring='accuracy',cv=0)

sfs1 = sfs1.fit(feature_raw, output_vector)

error_estimate = 1-sfs1.k_score_

feature_train = feature_raw[:,sfs1.k_feature_idx_]

knn.fit(feature_train,output_vector)

feature_testing = feature_test[:,sfs1.k_feature_idx_]

output_pred = knn.predict(feature_testing)

acc = float((output_test == output_pred).sum()) / output_pred.shape[0]

testset_error = 1-acc
```

Conclusions:

For both the classification rules, (DLDA and 3NN) and the feature selection methods (sequential forward and exhaustive) the re-substitution error estimate is less than the test set error estimate. We can state that the re-substitution error estimate is optimistically biased.

In almost all cases the re-substitution error estimate goes on decreasing with decreasing test set error estimate (as feature set size increases). Also, The gene sets found by the two feature selection methods are different.

For both the classification rules, in the case of sequential forward search method we observe that the test set error estimate goes on decreasing as we increase the number of. On the other hand, no clear trend can be seen for exhaustive search method (might be as we have restricted the feature set size to 3 while in case of sequential forward search we are going up to the size of 8).

We cannot conclusively say whether having more samples or taking more features will improve the performance. Although increasing the feature set size even more will allow us to observe whether any kind of peaking phenomenon happens later on.

We observe that both the classification rules give re-substitution error estimates that are very close to zero.

We can try other rules that in combination with re-sub error that might result in lesser bias. We can also try a different criterion in place of re-sub error.