Solution_D

October 15, 2020

[ ]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from math import *
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
```

[ ]:
```
data1 = pd.read_csv("Data/DLBCL.tsv",delimiter="\t", low_memory=False)
data1.drop(data1.index[[0,1]], inplace=True)
data1.dropna(inplace=True)

data2 = pd.read_csv("Data/leukemia.tsv",delimiter="\t", low_memory=False)
data2.drop(data2.index[[0,1]], inplace=True)
data2.dropna(inplace=True)

data3 = pd.read_csv("Data/lung.tsv",delimiter="\t", low_memory=False)
data3.drop(data3.index[[0,1]], inplace=True)
data3.dropna(inplace=True)
print("")
```

[ ]:
```
X1, y1 = data1.iloc[:,:-1], data1.iloc[:,-1]
X2, y2 = data2.iloc[:,1:], data2.iloc[:,0]
X3, y3 = data3.iloc[:,1:], data3.iloc[:,0]
```

[ ]:
```
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.25)
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.25)
X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size=0.25)
```

[ ]:
```
n1_features = len(X1.columns)
n2_features = len(X2.columns)
n3_features = len(X3.columns)

print("No. of features in DBCL data: {}".format(n1_features))
```

```
print("No. of features in Leukemia data: {}".format(n2_features))
print("No. of features in Lung data: {}".format(n3_features))
```

[ ]:
```
n1_select = floor(0.2*n1_features)
n2_select = floor(0.2*n2_features)
n3_select = floor(0.2*n3_features)

print("No. of features to be selected from DBCL data (N/3): {}".
 ↪format(n1_select))
print("No. of features to be selected from Leukemia data (N/3): {}".
 ↪format(n2_select))
print("No. of features to be selected from Lung data (N/3): {}".
 ↪format(n3_select))
```

[ ]:
```
def results_kNN(X_train, y_train, X_test, y_test):
    Clf = KNeighborsClassifier(n_neighbors=3)
    Clf.fit(X_train, y_train)

    y_pred = Clf.predict(X_test)
    acc = accuracy_score(y_test.values, y_pred)
    fscore = f1_score(y_test.values, y_pred, average='weighted')
    cnf_matrix = confusion_matrix(y_test, y_pred)

    return acc, fscore, cnf_matrix

def results_svm(X_train, y_train, X_test, y_test):
    Clf = SVC(kernel='rbf')
    Clf.fit(X_train, y_train)

    y_pred = Clf.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    fscore = f1_score(y_test, y_pred, average='weighted')
    cnf_matrix = confusion_matrix(y_test, y_pred)

    return acc, fscore, cnf_matrix

def sfs(X, y, clf, n_feature):
    f_set = set()

    features = X.columns
    k = 0

    while k < n_feature:
        f_dict = {}

        for feature in features:
            if feature not in f_set:
```

```python
                tmp = f_set.copy()
                tmp.add(feature)

                tmp_list = list(tmp)

                X_tmp = X[tmp_list]

                model = clf.fit(X_tmp, y)
                y_pred = model.predict(X_tmp)

                acc = accuracy_score(y, y_pred)

                f_dict[feature] = acc

        next_f = max(f_dict, key=f_dict.get)
        f_set.add(next_f)
        k = k + 1

    Xf = X[list(f_set)]

    return Xf, list(f_set)

def sbs(X, y, clf, n_feature):
    features = X.columns
    f_set = set(features)
    k = len(features)

    while k > n_feature:
        f_dict = {}

        for feature in features:
            if feature in f_set:
                tmp = f_set.copy()
                tmp.remove(feature)

                tmp_list = list(tmp)

                X_tmp = X[tmp_list]

                model = clf.fit(X_tmp, y)
                y_pred = model.predict(X_tmp)

                acc = accuracy_score(y, y_pred)

                f_dict[feature] = acc

        next_f = max(f_dict, key=f_dict.get)
```

```
        f_set.remove(next_f)
        k = k - 1

    Xf = X[list(f_set)]

    return Xf, list(f_set)
```

0.1 DBCL Dataset

0.1.1 Sequential Forward Selection (SFS)

[ ]:
```
clf1 = KNeighborsClassifier(n_neighbors=3)
X1_train_sfs, fs1 = sfs(X1_train, y1_train, clf1, n1_select)
X1_test_sfs = X1_test[fs1]
```

[ ]:
```
knn_results1 = results_kNN(X1_train_sfs, y1_train, X1_test_sfs, y1_test)

print("kNN Results for DBCL data:")
print("Accuracy: {}".format(knn_results1[0]))
print("Weighted F1-Score: {}".format(knn_results1[1]))
print("Confusion Matrix:")
print(knn_results1[2])
print("\n")
```

[ ]:
```
clf1 = SVC(kernel='rbf')
X1_train_sfs, fs1 = sfs(X1_train, y1_train, clf1, n1_select)
X1_test_sfs = X1_test[fs1]
```

[ ]:
```
svm_results1 = results_svm(X1_train_fs, y1_train, X1_test_fs, y1_test)

print("SVM Results for DBCL data:")
print("Accuracy: {}".format(svm_results1[0]))
print("Weighted F1-Score: {}".format(svm_results1[1]))
print("Confusion Matrix:")
print(svm_results1[2])
print("\n")
```

[ ]:
```

```

0.1.2 Sequential Backward Selection (SBS)

[ ]:
```
clf1 = KNeighborsClassifier(n_neighbors=3)
X1_train_sfs, fs1 = sbs(X1_train, y1_train, clf1, n1_select)
X1_test_sfs = X1_test[fs1]
```

[ ]:
```
knn_results1 = results_kNN(X1_train_sfs, y1_train, X1_test_sfs, y1_test)

print("kNN Results for DBCL data:")
print("Accuracy: {}".format(knn_results1[0]))
print("Weighted F1-Score: {}".format(knn_results1[1]))
print("Confusion Matrix:")
print(knn_results1[2])
print("\n")
```

[ ]:
```
clf1 = SVC(kernel='rbf')
X1_train_sfs, fs1 = sbs(X1_train, y1_train, clf1, n1_select)
X1_test_sfs = X1_test[fs1]
```

[ ]:
```
svm_results1 = results_svm(X1_train_fs, y1_train, X1_test_fs, y1_test)

print("SVM Results for DBCL data:")
print("Accuracy: {}".format(svm_results1[0]))
print("Weighted F1-Score: {}".format(svm_results1[1]))
print("Confusion Matrix:")
print(svm_results1[2])
print("\n")
```

[ ]:
```

```

0.2 Leukemia Dataset

0.2.1 Sequential Forward Selection (SFS)

[ ]:
```
clf2 = KNeighborsClassifier(n_neighbors=3)
X2_train_sfs, fs2 = sfs(X2_train, y2_train, clf2, n2_select)
X2_test_sfs = X2_test[fs2]
```

[ ]:
```
knn_results1 = results_kNN(X2_train_sfs, y2_train, X2_test_sfs, y2_test)

print("kNN Results for Leukemia data:")
print("Accuracy: {}".format(knn_results1[0]))
print("Weighted F1-Score: {}".format(knn_results1[1]))
print("Confusion Matrix:")
print(knn_results1[2])
```

```
print("\n")
```

[ ]:
```
clf2 = SVC(kernel='rbf')
X2_train_sfs, fs2 = sfs(X2_train, y2_train, clf2, n2_select)
X2_test_sfs = X2_test[fs2]
```

[ ]:
```
svm_results1 = results_svm(X2_train_fs, y2_train, X2_test_fs, y2_test)

print("SVM Results for Leukemia data:")
print("Accuracy: {}".format(svm_results1[0]))
print("Weighted F1-Score: {}".format(svm_results1[1]))
print("Confusion Matrix:")
print(svm_results1[2])
print("\n")
```

[ ]:
```

```

0.2.2 Sequential Backward Selection (SBS)

[ ]:
```
clf2 = KNeighborsClassifier(n_neighbors=3)
X2_train_sfs, fs2 = sbs(X2_train, y2_train, clf2, n2_select)
X2_test_sfs = X2_test[fs2]
```

[ ]:
```
knn_results1 = results_kNN(X2_train_sfs, y2_train, X2_test_sfs, y2_test)

print("kNN Results for DBCL data:")
print("Accuracy: {}".format(knn_results1[0]))
print("Weighted F1-Score: {}".format(knn_results1[1]))
print("Confusion Matrix:")
print(knn_results1[2])
print("\n")
```

[ ]:
```
clf2 = SVC(kernel='rbf')
X2_train_sfs, fs2 = sbs(X2_train, y2_train, clf2, n2_select)
X2_test_sfs = X2_test[fs2]
```

[ ]:
```
svm_results1 = results_svm(X2_train_fs, y2_train, X2_test_fs, y2_test)

print("SVM Results for DBCL data:")
print("Accuracy: {}".format(svm_results1[0]))
print("Weighted F1-Score: {}".format(svm_results1[1]))
print("Confusion Matrix:")
print(svm_results1[2])
print("\n")
```

[ ]:
```

```

0.3 Lung Dataset

0.3.1 Sequential Forward Selection (SFS)

[ ]:
```
clf3 = KNeighborsClassifier(n_neighbors=3)
X3_train_sfs, fs3 = sfs(X3_train, y3_train, clf3, n3_select)
X3_test_sfs = X3_test[fs3]
```

[ ]:
```
knn_results1 = results_kNN(X3_train_sfs, y3_train, X3_test_sfs, y3_test)

print("kNN Results for DBCL data:")
print("Accuracy: {}".format(knn_results1[0]))
print("Weighted F1-Score: {}".format(knn_results1[1]))
print("Confusion Matrix:")
print(knn_results1[2])
print("\n")
```

[ ]:
```
clf3 = SVC(kernel='rbf')
X3_train_sfs, fs3 = sfs(X3_train, y3_train, clf3, n3_select)
X3_test_sfs = X3_test[fs3]
```

[ ]:
```
svm_results1 = results_svm(X3_train_fs, y3_train, X3_test_fs, y3_test)

print("SVM Results for DBCL data:")
print("Accuracy: {}".format(svm_results1[0]))
print("Weighted F1-Score: {}".format(svm_results1[1]))
print("Confusion Matrix:")
print(svm_results1[2])
print("\n")
```

[ ]:
```

```

0.3.2 Sequential Backward Selection (SBS)

[ ]:
```
clf3 = KNeighborsClassifier(n_neighbors=3)
X3_train_sfs, fs3 = sbs(X3_train, y3_train, clf3, n3_select)
X3_test_sfs = X3_test[fs3]
```

[ ]:
```
knn_results1 = results_kNN(X3_train_sfs, y3_train, X3_test_sfs, y3_test)

print("kNN Results for DBCL data:")
print("Accuracy: {}".format(knn_results1[0]))
```

```python
print("Weighted F1-Score: {}".format(knn_results1[1]))
print("Confusion Matrix:")
print(knn_results1[2])
print("\n")
```

[ ]:
```python
clf3 = SVC(kernel='rbf')
X3_train_sfs, fs3 = sbs(X3_train, y3_train, clf3, n3_select)
X3_test_sfs = X3_test[fs3]
```

[ ]:
```python
svm_results1 = results_svm(X3_train_fs, y3_train, X3_test_fs, y3_test)

print("SVM Results for DBCL data:")
print("Accuracy: {}".format(svm_results1[0]))
print("Weighted F1-Score: {}".format(svm_results1[1]))
print("Confusion Matrix:")
print(svm_results1[2])
print("\n")
```