# Solution_A

October 15, 2020

[1]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
import filters, mutual_info, ttest
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
```

[2]:
```
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("")
```

[3]:
```
X1, y1 = data1.iloc[:,:-1], data1.iloc[:,-1]
X2, y2 = data2.iloc[:,1:], data2.iloc[:,0]
X3, y3 = data3.iloc[:,1:], data3.iloc[:,0]
```

[4]:
```
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)
```

[5]:
```
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))
```

```
No. of features in DBCL data: 7070
No. of features in Leukemia data: 5147
No. of features in Lung data: 12600
```

[6]:
```
n1_select = floor(0.2*n1_features)-1
n2_select = floor(0.2*n2_features)-1
n3_select = floor(0.2*n3_features)-1

print("No. of features to be selected from DBCL data (20%): {}".
 ↪format(n1_select))
print("No. of features to be selected from Leukemia data (20%): {}".
 ↪format(n2_select))
print("No. of features to be selected from Lung data (20%): {}".
 ↪format(n3_select))
```

```
No. of features to be selected from DBCL data (20%): 1413
No. of features to be selected from Leukemia data (20%): 1028
No. of features to be selected from Lung data (20%): 2519
```

[7]:
```
def topk_features_mi(X, y, n_features):
    fi = mutual_info.mutual_info_classif(X, y)
    f = X.columns
    fs = [j for i,j in sorted(zip(fi,f), reverse=True)][:n_features]
    fi = [i for i,j in sorted(zip(fi,f), reverse=True)][:n_features]
    Xs = X[fs]

    return Xs, y, fs, fi

def topk_features_f_calssif(X, y, n_features):
    fi = filters.f_classif(X, y)
    fi = fi[0]
    f = X.columns
    fs = [j for i,j in sorted(zip(fi,f), reverse=True)][:n_features]
    fi = [i for i,j in sorted(zip(fi,f), reverse=True)][:n_features]
    Xs = X[fs]

    return Xs, y, fs, fi

def topk_features_t_test(X, y, n_features):
    f,fi = ttest.get_features(X, y)
#     fi = fi[0]
#     print(f)
#     f = X.columns
```

```
    fs = [j for i,j in sorted(zip(fi,f), reverse=True)][:n_features]
    fi = [i for i,j in sorted(zip(fi,f), reverse=True)][:n_features]
    Xs = X[fs]

    return Xs, y, fs, fi

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
```

### 0.1 Feature Selection Using Mutual Information

[8]:
```
X1_train_mi, y1_train_mi, fs1, fi1 = topk_features_mi(X1_train, y1_train,␣
↪n1_select)
X1_test_mi = X1_test[fs1]

X2_train_mi, y2_train_mi, fs2, fi2 = topk_features_mi(X2_train, y2_train,␣
↪n2_select)
X2_test_mi = X2_test[fs2]

X3_train_mi, y3_train_mi, fs3, fi3 = topk_features_mi(X3_train, y3_train,␣
↪n3_select)
X3_test_mi = X3_test[fs3]
```

[9]:
```
knn_results1 = results_kNN(X1_train_mi, y1_train_mi, X1_test_mi, y1_test)
svm_results1 = results_svm(X1_train_mi, y1_train_mi, X1_test_mi, 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")

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")
```

```
kNN Results for DBCL data:
Accuracy: 0.95
Weighted F1-Score: 0.9480286738351253
Confusion Matrix:
[[15  0]
 [ 1  4]]


SVM Results for DBCL data:
Accuracy: 0.95
Weighted F1-Score: 0.9480286738351253
Confusion Matrix:
[[15  0]
 [ 1  4]]
```

[10]:
```
knn_results2 = results_kNN(X2_train_mi, y2_train_mi, X2_test_mi, y2_test)
svm_results2 = results_svm(X2_train_mi, y2_train_mi, X2_test_mi, y2_test)

print("kNN Results for Leukemia data:")
print("Accuracy: {}".format(knn_results2[0]))
print("Weighted F1-Score: {}".format(knn_results2[1]))
print("Confusion Matrix:")
print(knn_results2[2])
print("\n")

print("SVM Results for Leukemia data:")
print("Accuracy: {}".format(svm_results2[0]))
print("Weighted F1-Score: {}".format(svm_results2[1]))
print("Confusion Matrix:")
print(svm_results2[2])
print("\n")
```

```
kNN Results for Leukemia data:
Accuracy: 0.9444444444444444
Weighted F1-Score: 0.9430303030303031
Confusion Matrix:
[[12  0]
 [ 1  5]]


SVM Results for Leukemia data:
Accuracy: 0.9444444444444444
Weighted F1-Score: 0.9430303030303031
Confusion Matrix:
[[12  0]
 [ 1  5]]
```

[11]:
```
knn_results3 = results_kNN(X3_train_mi, y3_train_mi, X3_test_mi, y3_test)
svm_results3 = results_svm(X3_train_mi, y3_train_mi, X3_test_mi, y3_test)

print("kNN Results for Lung data:")
print("Accuracy: {}".format(knn_results3[0]))
print("Weighted F1-Score: {}".format(knn_results3[1]))
print("Confusion Matrix:")
print(knn_results3[2])
print("\n")

print("SVM Results for Lung data:")
print("Accuracy: {}".format(svm_results3[0]))
print("Weighted F1-Score: {}".format(svm_results3[1]))
print("Confusion Matrix:")
print(svm_results3[2])
print("\n")
```

```
kNN Results for Lung data:
Accuracy: 0.8627450980392157
Weighted F1-Score: 0.836078431372549
Confusion Matrix:
[[34  0  1  0  1]
 [ 0  1  0  0  0]
 [ 1  0  4  0  0]
 [ 3  0  0  0  0]
 [ 1  0  0  0  5]]


SVM Results for Lung data:
Accuracy: 0.8235294117647058
Weighted F1-Score: 0.7823281211218664
```

```
Confusion Matrix:
[[35  0  1  0  0]
 [ 0  1  0  0  0]
 [ 1  0  4  0  0]
 [ 3  0  0  0  0]
 [ 4  0  0  0  2]]
```

### 0.2 Feature Selection Using f_classif

[12]:
```
X1_train_f, y1_train_f, fs1, fi1 = topk_features_f_calssif(X1_train, y1_train,␣
 ↪n1_select)
X1_test_f = X1_test[fs1]

X2_train_f, y2_train_f, fs2, fi2 = topk_features_f_calssif(X2_train, y2_train,␣
 ↪n2_select)
X2_test_f = X2_test[fs2]

X3_train_f, y3_train_f, fs3, fi3 = topk_features_f_calssif(X3_train, y3_train,␣
 ↪n3_select)
X3_test_f = X3_test[fs3]
```

[13]:
```
knn_results1 = results_kNN(X1_train_f, y1_train_f, X1_test_f, y1_test)
svm_results1 = results_svm(X1_train_f, y1_train_f, X1_test_f, 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")

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")
```

```
kNN Results for DBCL data:
Accuracy: 1.0
Weighted F1-Score: 1.0
Confusion Matrix:
[[15  0]
 [ 0  5]]
```

```
SVM Results for DBCL data:
Accuracy: 0.9
Weighted F1-Score: 0.890625
Confusion Matrix:
[[15  0]
 [ 2  3]]
```

[14]:
```
knn_results2 = results_kNN(X2_train_f, y2_train_f, X2_test_f, y2_test)
svm_results2 = results_svm(X2_train_f, y2_train_f, X2_test_f, y2_test)

print("kNN Results for Leukemia data:")
print("Accuracy: {}".format(knn_results2[0]))
print("Weighted F1-Score: {}".format(knn_results2[1]))
print("Confusion Matrix:")
print(knn_results2[2])
print("\n")

print("SVM Results for Leukemia data:")
print("Accuracy: {}".format(svm_results2[0]))
print("Weighted F1-Score: {}".format(svm_results2[1]))
print("Confusion Matrix:")
print(svm_results2[2])
print("\n")
```

```
kNN Results for Leukemia data:
Accuracy: 0.9444444444444444
Weighted F1-Score: 0.9430303030303031
Confusion Matrix:
[[12  0]
 [ 1  5]]


SVM Results for Leukemia data:
Accuracy: 0.9444444444444444
Weighted F1-Score: 0.9430303030303031
Confusion Matrix:
[[12  0]
 [ 1  5]]
```

[15]:
```
knn_results3 = results_kNN(X3_train_f, y3_train_f, X3_test_f, y3_test)
svm_results3 = results_svm(X3_train_f, y3_train_f, X3_test_f, y3_test)
```

```
print("kNN Results for Lung data:")
print("Accuracy: {}".format(knn_results3[0]))
print("Weighted F1-Score: {}".format(knn_results3[1]))
print("Confusion Matrix:")
print(knn_results3[2])
print("\n")

print("SVM Results for Lung data:")
print("Accuracy: {}".format(svm_results3[0]))
print("Weighted F1-Score: {}".format(svm_results3[1]))
print("Confusion Matrix:")
print(svm_results3[2])
print("\n")
```

```
kNN Results for Lung data:
Accuracy: 0.8235294117647058
Weighted F1-Score: 0.7843986079280196
Confusion Matrix:
[[34  0  1  0  1]
 [ 0  1  0  0  0]
 [ 0  0  5  0  0]
 [ 3  0  0  0  0]
 [ 4  0  0  0  2]]


SVM Results for Lung data:
Accuracy: 0.8627450980392157
Weighted F1-Score: 0.8338680926916222
Confusion Matrix:
[[35  0  1  0  0]
 [ 0  1  0  0  0]
 [ 1  0  4  0  0]
 [ 3  0  0  0  0]
 [ 2  0  0  0  4]]
```

### 0.3 Feature Selection Using t-test

[16]:
```
X1_train_t, y1_train_t, fs1, fi1 = topk_features_t_test(X1_train, y1_train,␣
↪n1_select)
X1_test_t = X1_test[fs1]

X2_train_t, y2_train_t, fs2, fi2 = topk_features_t_test(X2_train, y2_train,␣
↪n2_select)
```

```
X2_test_t = X2_test[fs2]

X3_train_t, y3_train_t, fs3, fi3 = topk_features_t_test(X3_train, y3_train,␣
 ↪n3_select)
X3_test_t = X3_test[fs3]
```

```
/home/aquarius31/EndSem Notes/ML - Assignments/Assignments/Assignment
2/ttest.py:39: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df["class"] = pd.DataFrame(y)
```

[17]:
```
knn_results1 = results_kNN(X1_train_t, y1_train_t, X1_test_t, y1_test)
svm_results1 = results_svm(X1_train_t, y1_train_t, X1_test_t, 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")

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")
```

```
kNN Results for DBCL data:
Accuracy: 1.0
Weighted F1-Score: 1.0
Confusion Matrix:
[[15  0]
 [ 0  5]]


SVM Results for DBCL data:
Accuracy: 0.9
Weighted F1-Score: 0.890625
Confusion Matrix:
[[15  0]
 [ 2  3]]
```

[18]:
```
knn_results2 = results_kNN(X2_train_t, y2_train_t, X2_test_t, y2_test)
svm_results2 = results_svm(X2_train_t, y2_train_t, X2_test_t, y2_test)

print("kNN Results for Leukemia data:")
print("Accuracy: {}".format(knn_results2[0]))
print("Weighted F1-Score: {}".format(knn_results2[1]))
print("Confusion Matrix:")
print(knn_results2[2])
print("\n")

print("SVM Results for Leukemia data:")
print("Accuracy: {}".format(svm_results2[0]))
print("Weighted F1-Score: {}".format(svm_results2[1]))
print("Confusion Matrix:")
print(svm_results2[2])
print("\n")
```

```
kNN Results for Leukemia data:
Accuracy: 0.9444444444444444
Weighted F1-Score: 0.9430303030303031
Confusion Matrix:
[[12  0]
 [ 1  5]]


SVM Results for Leukemia data:
Accuracy: 0.9444444444444444
Weighted F1-Score: 0.9430303030303031
Confusion Matrix:
[[12  0]
 [ 1  5]]
```

[19]:
```
knn_results3 = results_kNN(X3_train_t, y3_train_t, X3_test_t, y3_test)
svm_results3 = results_svm(X3_train_t, y3_train_t, X3_test_t, y3_test)

print("kNN Results for Lung data:")
print("Accuracy: {}".format(knn_results3[0]))
print("Weighted F1-Score: {}".format(knn_results3[1]))
print("Confusion Matrix:")
print(knn_results3[2])
print("\n")

print("SVM Results for Lung data:")
```

```
print("Accuracy: {}".format(svm_results3[0]))
print("Weighted F1-Score: {}".format(svm_results3[1]))
print("Confusion Matrix:")
print(svm_results3[2])
print("\n")
```

```
kNN Results for Lung data:
Accuracy: 0.8431372549019608
Weighted F1-Score: 0.8181077004606415
Confusion Matrix:
[[33  0  1  0  2]
 [ 0  1  0  0  0]
 [ 1  0  4  0  0]
 [ 3  0  0  0  0]
 [ 1  0  0  0  5]]


SVM Results for Lung data:
Accuracy: 0.803921568627451
Weighted F1-Score: 0.7492997198879552
Confusion Matrix:
[[35  0  1  0  0]
 [ 0  1  0  0  0]
 [ 1  0  4  0  0]
 [ 3  0  0  0  0]
 [ 5  0  0  0  1]]
```

[ ]: