In [61]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from scipy.spatial.distance import pdist, squareform
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram

from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils
# from keras.utils import to_categorical
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
```

In [7]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [34]:
```
# read data
df = pd.read_csv('demographicsLifeExpectancy.csv')
print(df.head())

# to find out any na in dataset
df.isna().sum()
```

```
  HeartDisease    BMI Smoking AlcoholDrinking Stroke  PhysicalHealth  \
0           No  16.60     Yes              No     No             3.0
1           No  20.34      No              No    Yes             0.0
2           No  26.58     Yes              No     No            20.0
3           No  24.21      No              No     No             0.0
4           No  23.71      No              No     No            28.0

   MentalHealth DiffWalking     Sex  AgeCategory   Race Diabetic  \
0          30.0          No  Female        55-59  White      Yes
1           0.0          No  Female  80 or older  White       No
2          30.0          No    Male        65-69  White      Yes
3           0.0          No  Female        75-79  White       No
4           0.0         Yes  Female        40-44  White       No

  PhysicalActivity  GenHealth  SleepTime Asthma KidneyDisease SkinCancer
0              Yes  Very good        5.0    Yes            No        Yes
1              Yes  Very good        7.0     No            No         No
2              Yes       Fair        8.0    Yes            No         No
3               No       Good        6.0     No            No        Yes
4              Yes  Very good        8.0     No            No         No
```

Out[34]:
```
HeartDisease        0
BMI                 0
Smoking             0
AlcoholDrinking     0
Stroke              0
PhysicalHealth      0
MentalHealth        0
DiffWalking         0
Sex                 0
AgeCategory         0
Race                0
Diabetic            0
PhysicalActivity    0
GenHealth           0
SleepTime           0
Asthma              0
KidneyDisease       0
SkinCancer          0
dtype: int64
```

In [35]:
```
# unique value for each column
for col in df.columns:
    print(col, df[col].unique())
```

```
HeartDisease ['No' 'Yes']
BMI [16.6  20.34 26.58 ... 62.42 51.46 46.56]
Smoking ['Yes' 'No']
AlcoholDrinking ['No' 'Yes']
Stroke ['No' 'Yes']
PhysicalHealth [ 3.  0. 20. 28.  6. 15.  5. 30.  7.  1.  2. 21.  4. 10. 14. 18.  8. 25.
 16. 29. 27. 17. 24. 12. 23. 26. 22. 19.  9. 13. 11.]
MentalHealth [30.  0.  2.  5. 15.  8.  4.  3. 10. 14. 20.  1.  7. 24.  9. 28. 16. 12.
  6. 25. 17. 18. 21. 29. 22. 13. 23. 27. 26. 11. 19.]
DiffWalking ['No' 'Yes']
Sex ['Female' 'Male']
AgeCategory ['55-59' '80 or older' '65-69' '75-79' '40-44' '70-74' '60-64' '50-54'
 '45-49' '18-24' '35-39' '30-34' '25-29']
Race ['White' 'Black' 'Asian' 'American Indian/Alaskan Native' 'Other'
 'Hispanic']
Diabetic ['Yes' 'No' 'No, borderline diabetes' 'Yes (during pregnancy)']
PhysicalActivity ['Yes' 'No']
GenHealth ['Very good' 'Fair' 'Good' 'Poor' 'Excellent']
SleepTime [ 5.  7.  8.  6. 12.  4.  9. 10. 15.  3.  2.  1. 16. 18. 14. 20. 11. 13.
```

```
 17. 24. 19. 21. 22. 23.]
Asthma ['Yes' 'No']
KidneyDisease ['No' 'Yes']
SkinCancer ['Yes' 'No']
```

In [53]:

```
# transfer categories to numbers
df_num = df
df_num['HeartDisease'] = df_num['HeartDisease'].astype('category').cat.codes
df_num['Smoking'] = df_num['Smoking'].astype('category').cat.codes
df_num['AlcoholDrinking'] = df_num['AlcoholDrinking'].astype('category').cat.codes
df_num['Stroke'] = df_num['Stroke'].astype('category').cat.codes
df_num['DiffWalking'] = df_num['DiffWalking'].astype('category').cat.codes
df_num['Sex'] = df_num['Sex'].astype('category').cat.codes
df_num['AgeCategory'] = df_num['AgeCategory'].astype('category').cat.codes
df_num['Race'] = df_num['Race'].astype('category').cat.codes
df_num['Diabetic'] = df_num['Diabetic'].astype('category').cat.codes
df_num['PhysicalActivity'] = df_num['PhysicalActivity'].astype('category').cat.codes
df_num['GenHealth'] = df_num['GenHealth'].astype('category').cat.codes
df_num['Asthma'] = df_num['Asthma'].astype('category').cat.codes
df_num['KidneyDisease'] = df_num['KidneyDisease'].astype('category').cat.codes
df_num['SkinCancer'] = df_num['SkinCancer'].astype('category').cat.codes

print(df_num.shape)
df_num.head()
```

```
(319795, 18)
```

Out[53]:

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabetic	PhysicalActivity	GenHealth	SleepTime	Asthm
0	0	16.60	1	0	0	3.0	30.0	0	0	7	5	2	1	4	5.0	
1	0	20.34	0	0	1	0.0	0.0	0	0	12	5	0	1	4	7.0	
2	0	26.58	1	0	0	20.0	30.0	0	1	9	5	2	1	1	8.0	
3	0	24.21	0	0	0	0.0	0.0	0	0	11	5	0	0	2	6.0	
4	0	23.71	0	0	0	28.0	0.0	1	0	4	5	0	1	4	8.0	

In [39]:

```
# check the transformations
for col in df_num.columns:
    print(col, df_num[col].unique())
```

```
HeartDisease [0 1]
BMI [16.6  20.34 26.58 ... 62.42 51.46 46.56]
Smoking [1 0]
AlcoholDrinking [0 1]
Stroke [0 1]
PhysicalHealth [ 3.  0. 20. 28.  6. 15.  5. 30.  7.  1.  2. 21.  4. 10. 14. 18.  8. 25.
 16. 29. 27. 17. 24. 12. 23. 26. 22. 19.  9. 13. 11.]
MentalHealth [30.  0.  2.  5. 15.  8.  4.  3. 10. 14. 20.  1.  7. 24.  9. 28. 16. 12.
  6. 25. 17. 18. 21. 29. 22. 13. 23. 27. 26. 11. 19.]
DiffWalking [0 1]
Sex [0 1]
AgeCategory [ 7 12  9 11  4 10  8  6  5  0  3  2  1]
Race [5 2 1 0 4 3]
Diabetic [2 0 1 3]
PhysicalActivity [1 0]
GenHealth [4 1 2 3 0]
SleepTime [ 5.  7.  8.  6. 12.  4.  9. 10. 15.  3.  2.  1. 16. 18. 14. 20. 11. 13.
 17. 24. 19. 21. 22. 23.]
Asthma [1 0]
KidneyDisease [0 1]
SkinCancer [1 0]
```

In [40]:

```
# visulize of the dataset
sns.pairplot(df_num)
plt.tight_layout()
plt.show()
```

In [51]:
```
# seperate X and y
X = df_num.loc[:, df_num.columns != 'Smoking']
y = df_num['Smoking']
```

In [52]:
```
X.shape, y.shape
```

Out[52]:
```
((319795, 17), (319795,))
```

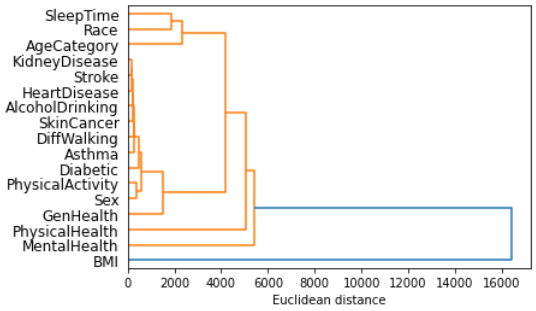
In [60]:
```
# Performing hierarchical clustering on a distance matrix for X
feature_clusters = linkage(X.T.values, method='complete', metric='euclidean')
pd.DataFrame(feature_clusters,
             columns=['row label 1', 'row label 2',
                      'distance', 'no. of items in clust.'],
             index=['cluster %d' % (i + 1)
                    for i in range(feature_clusters.shape[0])])
```

Out[60]:

	row label 1	row label 2	distance	no. of items in clust.
cluster 1	3.0	15.0	144.450684	2.0
cluster 2	0.0	17.0	179.560575	3.0
cluster 3	2.0	18.0	216.490185	4.0
cluster 4	16.0	19.0	218.636685	5.0
cluster 5	6.0	20.0	249.212761	6.0

	row label 1	row label 2	distance	no. of items in clust.
cluster 6	14.0	21.0	259.992308	7.0
cluster 7	7.0	11.0	397.266913	2.0
cluster 8	10.0	22.0	456.439481	8.0
cluster 9	23.0	24.0	563.564548	10.0
cluster 10	12.0	25.0	1513.914463	11.0
cluster 11	9.0	13.0	1849.674836	2.0
cluster 12	8.0	27.0	2348.883565	3.0
cluster 13	26.0	28.0	4176.245802	14.0
cluster 14	4.0	29.0	5075.633458	15.0
cluster 15	5.0	30.0	5414.009789	16.0
cluster 16	1.0	31.0	16395.526558	17.0

In [65]:
```
# plot dendragram
feature_dendr = dendrogram(feature_clusters, labels=X.columns, orientation='right')
plt.xlabel('Euclidean distance')
plt.show()
```



In [72]:
```
# select independent features based on clustering result
X_s = X[['SleepTime','Race','AgeCategory','Diabetic','Sex','GenHealth','PhysicalHealth','MentalHealth','BMI']]
print(X_s.shape)
X_s.head()
```

```
(319795, 9)
```

Out[72]:

	SleepTime	Race	AgeCategory	Diabetic	Sex	GenHealth	PhysicalHealth	MentalHealth	BMI
0	5.0	5	7	2	0	4	3.0	30.0	16.60
1	7.0	5	12	0	0	4	0.0	0.0	20.34
2	8.0	5	9	2	1	1	20.0	30.0	26.58
3	6.0	5	11	0	0	2	0.0	0.0	24.21
4	8.0	5	4	0	0	4	28.0	0.0	23.71

In [74]:
```
# split into train/test subsets
X_train, X_test, y_train, y_test = train_test_split(X_s, y,
                                                    random_state=1000,
                                                    stratify=y,
                                                    test_size=0.1)
```

In [75]:
```
# standard the features
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```

In [76]:
```
# logistic regression model
# create a Logistic Regression instance
lr = LogisticRegression()

# Fit the data
lr.fit(X_train_std,y_train)

# get predictions
y_pred = lr.predict(X_test_std)

# calculate accuracy on test data
print('Accuracy: %.2f' %accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.62
```

In [77]:
```
# decision tree
crts = ["gini", "entropy"]
for crt in crts:
    tree_rotat = DecisionTreeClassifier(criterion=crt, random_state=1)
    tree_rotat.fit(X_train_std, y_train)
```

```
    y_pred = tree_rotat.predict(X_test_std)
    print(f'Accuracy for Decision Tree with criteria as {crt} Index is: {accuracy_score(y_test,y_pred)*100}')
```

```
Accuracy for Decision Tree with criteria as gini Index is: 56.682301438398994
Accuracy for Decision Tree with criteria as entropy Index is: 56.682301438398994
```

In [78]:
```
# select independent features based on clustering result
X_s2 = X[['MentalHealth','BMI']]
print(X_s2.shape)
X_s2.head()
```

```
(319795, 2)
```

Out[78]:

	MentalHealth	BMI
0	30.0	16.60
1	0.0	20.34
2	30.0	26.58
3	0.0	24.21
4	0.0	23.71

In [79]:
```
# split into train/test subsets
X_train, X_test, y_train, y_test = train_test_split(X_s2, y,
                                                    random_state=1000,
                                                    stratify=y,
                                                    test_size=0.1)
```

In [80]:
```
# standard the features
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```

In [81]:
```
# logistic regression model
# create a Logistic Regression instance
lr = LogisticRegression()

# Fit the data
lr.fit(X_train_std,y_train)

# get predictions
y_pred = lr.predict(X_test_std)

# calculate accuracy on test data
print('Accuracy: %.2f' %accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.59
```

In [82]:
```
# decision tree
crts = ["gini", "entropy"]
for crt in crts:
    tree_rotat = DecisionTreeClassifier(criterion=crt, random_state=1)
    tree_rotat.fit(X_train_std, y_train)
    y_pred = tree_rotat.predict(X_test_std)
    print(f'Accuracy for Decision Tree with criteria as {crt} Index is: {accuracy_score(y_test,y_pred)*100}')
```

```
Accuracy for Decision Tree with criteria as gini Index is: 58.26766729205753
Accuracy for Decision Tree with criteria as entropy Index is: 58.24577861163227
```

In [ ]:
```
```