In [47]:
```
from python_speech_features import mfcc
from python_speech_features import logfbank
import scipy.io.wavfile as wav
import pandas as pd
import numpy as np
from statistics import stdev
import IPython.display as ipd
%matplotlib inline
import matplotlib.pyplot as plt
import librosa.display
import itertools
```

In [48]:
```
import librosa
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import os
from PIL import Image
import pathlib
import csv

# Preprocessing
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.svm import SVC
from sklearn.metrics import recall_score, precision_score, accuracy_score
from sklearn.metrics import confusion_matrix, f1_score, classification_report
from sklearn.model_selection import cross_val_score
```

In [51]:
```
labelencoder = LabelEncoder()
def plot_confusion_matrix(cm, classes,normalize=False,title='Confusion matrix'
,cmap=plt.cm.Reds):
    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45, labelsize=20)
    plt.yticks(tick_marks, classes, labelsize=20)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [3]:
```
header = 'filename rms spectral_centroid spectral_bandwidth rolloff zero_cross
ing_rate'
for i in range(1, 21):
    header += f' mfcc{i}'
header += ' label'
header = header.split()
```

In [4]:
```
file = open('data.csv', 'w', newline='')
with file:
    writer = csv.writer(file)
    writer.writerow(header)
```

In [5]:
```
Instruments = 'flu pia tru org gac voi'.split()
for i in Instruments:
    for filename in os.listdir(f'D:/PGDBA/ISI/CDS/Project/IRMAS-TrainingData/{i}'):
        songname = f'D:/PGDBA/ISI/CDS/Project/IRMAS-TrainingData/{i}/{filename}'
        y, sr = librosa.load(songname, sr =44100)
        rms = librosa.feature.rms(y=y)
        spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
        spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
        rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
        zcr = librosa.feature.zero_crossing_rate(y)
        mfcc = librosa.feature.mfcc(y=y, sr=sr)
        to_append = f'{filename} {np.mean(rms)} {np.mean(spec_cent)} {np.mean(spec_bw)} {np.mean(rolloff)} {np.mean(zcr)}'
        for e in mfcc:
            to_append += f' {np.mean(e)}'
        to_append += f' {i}'
        file = open('data.csv', 'a', newline='')
        with file:
            writer = csv.writer(file)
            writer.writerow(to_append.split())
```

In [7]:
```
df = pd.read_csv('data.csv')
df.head()
```

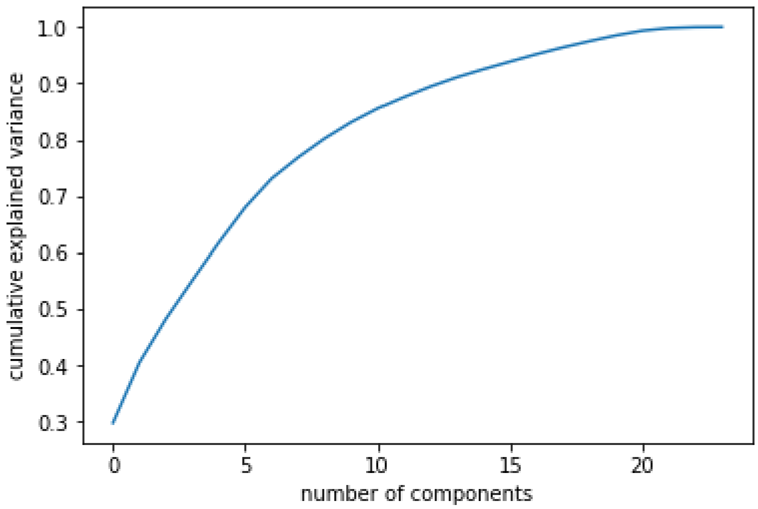
Out[7]:

	filename	rms	spectral_centroid	spectral_bandwidth	rolloff	zero_crossi
0	008__[flu][nod][cla]0393__1.wav	0.041481	1312.170004	1899.675233	1606.260557	0
1	008__[flu][nod][cla]0393__2.wav	0.038519	1111.202613	1564.692193	1589.300042	0
2	008__[flu][nod][cla]0393__3.wav	0.069043	1345.465834	1830.713470	1726.064981	0
3	009__[flu][nod][cou_fol]0410__1.wav	0.110917	3714.597457	2891.937598	6495.544764	0
4	009__[flu][nod][cou_fol]0410__2.wav	0.135875	3558.964511	2848.067932	6224.592219	0

5 rows × 27 columns

In [8]:
```
df.shape
features = df[df.columns[2:26]]
```

In [9]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
import matplotlib.pyplot as plt
X = scale(features)
pca = PCA().fit(X)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()
```



In [10]:
```
# Dropping unneccesary columns
df = df.drop(['filename'],axis=1)
```

In [11]:
```
df1 = shuffle(df)
```

In [12]:
```
df1.shape
```

Out[12]:
```
(3846, 26)
```

In [13]:
```
instru_list = df1.iloc[:, -1]
encoder = LabelEncoder()
y = encoder.fit_transform(instru_list)
```

In [14]:
```
scaler = StandardScaler()
X = scaler.fit_transform(np.array(df1.iloc[:, :-1], dtype = float))
```

In [15]:
```
df.head()
```

Out[15]:

	rms	spectral_centroid	spectral_bandwidth	rolloff	zero_crossing_rate	mfcc1
0	0.041481	1312.170004	1899.675233	1606.260557	0.042741	-380.747925
1	0.038519	1111.202613	1564.692193	1589.300042	0.033182	-392.709167
2	0.069043	1345.465834	1830.713470	1726.064981	0.048355	-326.334808
3	0.110917	3714.597457	2891.937598	6495.544764	0.083345	-278.512115
4	0.135875	3558.964511	2848.067932	6224.592219	0.082947	-271.450378

5 rows × 26 columns

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

SVM

In [62]:
```
svclassifier = SVC(kernel='rbf', C = 10.0, gamma=0.1)
```

In [63]:
```
svclassifier.fit(X_train, y_train)
```

Out[63]:
```
SVC(C=10.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [64]:
```
predicted_labels = svclassifier.predict(X_test)
```

In [65]:
```
print("Recall: ", recall_score(y_test, predicted_labels,average=None))
print("Precision: ", precision_score(y_test, predicted_labels,average=None))
print("F1-Score: ", f1_score(y_test, predicted_labels, average=None))
print("Accuracy: %.2f  ," % accuracy_score(y_test, predicted_labels,normalize=
True), accuracy_score(y_test, predicted_labels,normalize=False) )

print("Number of samples:",y_test.shape[0])
print(confusion_matrix(y_test, predicted_labels))
```

```
Recall:  [0.62886598 0.83870968 0.76923077 0.76315789 0.67226891 0.85135135]
Precision:  [0.62886598 0.78787879 0.78125    0.76821192 0.78431373 0.7875
]
F1-Score:  [0.62886598 0.8125     0.7751938  0.76567657 0.7239819  0.8181818
2]
Accuracy: 0.76  , 587
Number of samples: 770
[[ 61   5   8  12   5   6]
 [  8 104   1   7   0   4]
 [  7   4 100   1   7  11]
 [  7   9  11 116   5   4]
 [ 11   4   4  11  80   9]
 [  3   6   4   4   5 126]]
```

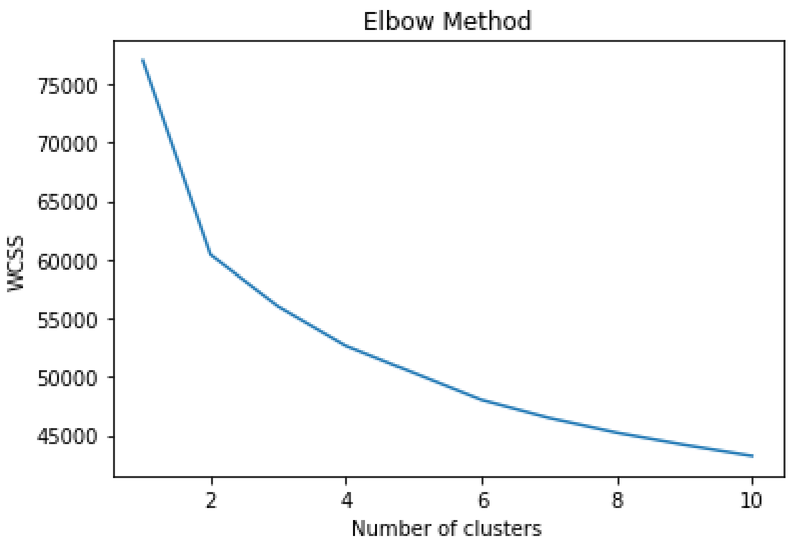
In [73]:
```
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
df_cm = pd.DataFrame(confusion_matrix(y_test, predicted_labels),index=["flu",
"pia", "tru", "org", "gac", "voi"], columns=["flu", "pia", "tru", "org", "gac"
, "voi"])
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.0)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 18},fmt='g')# font size

plt.show()
```

Clustering K means

In [29]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.datasets.samples_generator import make_blobs
from sklearn.cluster import KMeans
```

In [30]:
```
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, r
andom_state=0)
    kmeans.fit(X_train)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [31]:
```
kmeans = KMeans(n_clusters=6, init='k-means++', max_iter=300, n_init=10, rando
m_state=0)
pred_y = kmeans.fit_predict(X_train)
#plt.scatter(X[:,0], X[:,1])
#plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=3
00, c='red')
#plt.show()
```

In [41]:
```
pred_y
```

Out[41]:
```
array([5, 0, 4, ..., 0, 3, 4])
```

In [43]:
```
pd.dataframe[]
```

Out[43]:
```
array([[-1.34948109, -1.43410887, -1.1219222 , ...,  0.26030232,
        -0.00623883, -0.12059796],
       [ 0.86070974, -0.26592214, -0.34241653, ...,  0.29424463,
         0.47737547,  0.26350079],
       [ 0.57794983,  1.5149286 ,  1.43635985, ..., -0.09247361,
         0.28441564, -1.80962332],
       ...,
       [-0.46087121,  0.14777843,  0.28153824, ..., -0.81927998,
         0.02642052,  0.58981847],
       [ 0.74099405,  0.24589329,  0.33580323, ...,  0.44233721,
         0.27648732, -0.77995725],
       [ 0.40495827,  0.90220273,  1.05894706, ...,  1.20126641,
         0.90660586, -0.60152801]])
```

In [45]:
```
imp = pd.DataFrame()
imp['Prediction'] = pred_y
```

In [46]:
```
imp['Data'] = y_train
```

In [49]:
```
imp.to_csv('D:/PGDBA/ISI/CDS/Project/Cluster.csv')
```

LOgistic Regression

In [75]:
```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
#X, y = load_iris(return_X_y=True)
clf = LogisticRegression(random_state=0, solver='lbfgs', multi_class='multinom
ial',max_iter = 2000).fit(X_train, y_train)
```

In [76]:
```
predicted_labels = clf.predict(X_test)
```

In [77]:
```
print("Recall: ", recall_score(y_test, predicted_labels,average=None))
print("Precision: ", precision_score(y_test, predicted_labels,average=None))
print("F1-Score: ", f1_score(y_test, predicted_labels, average=None))
print("Accuracy: %.2f  ," % accuracy_score(y_test, predicted_labels,normalize=
True), accuracy_score(y_test, predicted_labels,normalize=False) )

print("Number of samples:",y_test.shape[0])
print(confusion_matrix(y_test, predicted_labels))
```

```
Recall:  [0.39175258 0.58870968 0.53076923 0.57236842 0.4789916  0.61486486]
Precision:  [0.58461538 0.5530303  0.44230769 0.62589928 0.57575758 0.5083798
9]
F1-Score:  [0.4691358  0.5703125  0.48251748 0.59793814 0.52293578 0.5565749
2]
Accuracy: 0.54  , 415
Number of samples: 770
[[38 10 15 16  3 15]
 [ 3 73 16 12  6 14]
 [ 2 14 69  8 11 26]
 [ 8 14 23 87 11  9]
 [14  2  9 13 57 24]
 [ 0 19 24  3 11 91]]
```

In [78]:
```
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
df_cm = pd.DataFrame(confusion_matrix(y_test, predicted_labels),index=["flu",
"pia", "tru", "org", "gac", "voi"], columns=["flu", "pia", "tru", "org", "gac"
, "voi"])
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.0)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 18},fmt='g')# font size

plt.show()
```

LGBM

In [79]:
```
from lightgbm import LGBMClassifier
m=LGBMClassifier()
m.fit(X_train, y_train)
```

Out[79]:
```
LGBMClassifier(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
               importance_type='split', learning_rate=0.1, max_depth=-1,
               min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
               n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,
               random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
               subsample=1.0, subsample_for_bin=200000, subsample_freq=0)
```

In [80]:
```
predicted_labels = m.predict(X_test)
```

In [81]:
```
print("Recall: ", recall_score(y_test, predicted_labels,average=None))
print("Precision: ", precision_score(y_test, predicted_labels,average=None))
print("F1-Score: ", f1_score(y_test, predicted_labels, average=None))
print("Accuracy: %.2f  ," % accuracy_score(y_test, predicted_labels,normalize=
True), accuracy_score(y_test, predicted_labels,normalize=False) )

print("Number of samples:",y_test.shape[0])
print(confusion_matrix(y_test, predicted_labels))
```

```
Recall:  [0.58762887 0.73387097 0.66923077 0.68421053 0.53781513 0.73648649]
Precision:  [0.65517241 0.69465649 0.58783784 0.72727273 0.71910112 0.6337209
3]
F1-Score:  [0.61956522 0.71372549 0.62589928 0.70508475 0.61538462 0.68125
]
Accuracy: 0.66  , 512
Number of samples: 770
[[ 57   7  13   9   2   9]
 [  6  91   6   9   1  11]
 [  3   9  87   4   8  19]
 [ 10  11  11 104   9   7]
 [  9   2  16  11  64  17]
 [  2  11  15   6   5 109]]
```

In [82]:
```
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
df_cm = pd.DataFrame(confusion_matrix(y_test, predicted_labels),index=["flu",
"pia", "tru", "org", "gac", "voi"], columns=["flu", "pia", "tru", "org", "gac"
, "voi"])
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.0)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 18},fmt='g')# font size

plt.show()
```

RAndomForest

In [83]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
clf = RandomForestClassifier(n_estimators = 150, random_state = 7000)
clf.fit(X_train, y_train)
```

Out[83]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=150, n_jobs=None,
            oob_score=False, random_state=7000, verbose=0,
            warm_start=False)
```

In [84]:
```
predicted_labels = clf.predict(X_test)
```

In [85]:
```
print("Recall: ", recall_score(y_test, predicted_labels,average=None))
print("Precision: ", precision_score(y_test, predicted_labels,average=None))
print("F1-Score: ", f1_score(y_test, predicted_labels, average=None))
print("Accuracy: %.2f  ," % accuracy_score(y_test, predicted_labels,normalize=
True), accuracy_score(y_test, predicted_labels,normalize=False) )

print("Number of samples:",y_test.shape[0])
print(confusion_matrix(y_test, predicted_labels))
```

```
Recall:  [0.48453608 0.75       0.72307692 0.71710526 0.53781513 0.7972973 ]
Precision:  [0.72307692 0.72093023 0.61437908 0.73154362 0.74418605 0.6276595
7]
F1-Score:  [0.58024691 0.73517787 0.66431095 0.72425249 0.62439024 0.7023809
5]
Accuracy: 0.68  , 525
Number of samples: 770
[[ 47  13  13  14   1   9]
 [  4  93   6  10   0  11]
 [  3   7  94   0   6  20]
 [  5   7  14 109   7  10]
 [  4   3  15  13  64  20]
 [  2   6  11   3   8 118]]
```

In [86]:
```
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
df_cm = pd.DataFrame(confusion_matrix(y_test, predicted_labels),index=["flu",
"pia", "tru", "org", "gac", "voi"], columns=["flu", "pia", "tru", "org", "gac"
, "voi"])
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.0)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 18},fmt='g')# font size

plt.show()
```

Decision Tree

In [93]:
```
from sklearn import tree
#Assumed you have, X (predictor) and Y (target) for training data set and x_te
st(predictor) of test_dataset
# Create tree object
model = tree.DecisionTreeClassifier(criterion='gini') # for classification, he
re you can change the algorithm as gini or entropy (information gain) by defau
lt it is gini
# model = tree.DecisionTreeRegressor() for regression
# Train the model using the training sets and check score
model.fit(X_train, y_train)
#Predict Output
predicted_labels = model.predict(X_test)
```

In [94]:
```
print("Recall: ", recall_score(y_test, predicted_labels,average=None))
print("Precision: ", precision_score(y_test, predicted_labels,average=None))
print("F1-Score: ", f1_score(y_test, predicted_labels, average=None))
print("Accuracy: %.2f  ," % accuracy_score(y_test, predicted_labels,normalize=
True), accuracy_score(y_test, predicted_labels,normalize=False) )

print("Number of samples:",y_test.shape[0])
print(confusion_matrix(y_test, predicted_labels))
```

```
Recall:  [0.44329897 0.54032258 0.46153846 0.56578947 0.50420168 0.5472973 ]
Precision:  [0.42574257 0.52755906 0.49586777 0.6013986  0.52173913 0.4969325
2]
F1-Score:  [0.43434343 0.53386454 0.47808765 0.58305085 0.51282051 0.5209003
2]
Accuracy: 0.52  , 397
Number of samples: 770
[[43 10 12 11  9 12]
 [15 67  6 18  0 18]
 [ 7 14 60 10 18 21]
 [19  9  9 86 14 15]
 [11  7 14 11 60 16]
 [ 6 20 20  7 14 81]]
```

In [95]:
```
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
df_cm = pd.DataFrame(confusion_matrix(y_test, predicted_labels),index=["flu",
"pia", "tru", "org", "gac", "voi"], columns=["flu", "pia", "tru", "org", "gac"
, "voi"])
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.0)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 18},fmt='g')# font size

plt.show()
```

Xgboost

In [98]:
```
import xgboost as xgb
XGBvlassifier = xgb.XGBClassifier(n_estimators=300, max_depth= 70, learning_ra
te=0.1)

XGBvlassifier.fit(X_train, y_train)
predicted_labels = XGBvlassifier.predict(X_test)
print("Recall: ", recall_score(y_test, predicted_labels,average=None))
print("Precision: ", precision_score(y_test, predicted_labels,average=None))
print("F1-Score: ", f1_score(y_test, predicted_labels, average=None))
print("Accuracy: %.2f  ," % accuracy_score(y_test, predicted_labels,normalize=
True), accuracy_score(y_test, predicted_labels,normalize=False) )

print("Number of samples:",y_test.shape[0])

confusion_matrix(y_test, predicted_labels)
```

```
Recall:  [0.58762887 0.70967742 0.69230769 0.71710526 0.52941176 0.74324324]
Precision:  [0.65517241 0.71544715 0.58441558 0.7124183  0.75       0.6508875
7]
F1-Score:  [0.61956522 0.71255061 0.63380282 0.7147541  0.62068966 0.6940063
1]
Accuracy: 0.67  , 517
Number of samples: 770
```

Out[98]:
```
array([[ 57,   8,  16,  10,   1,   5],
       [  6,  88,   5,  12,   1,  12],
       [  4,   9,  90,   2,   9,  16],
       [ 10,   7,  13, 109,   6,   7],
       [  8,   3,  14,  12,  63,  19],
       [  2,   8,  16,   8,   4, 110]], dtype=int64)
```

In [99]:
```
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
df_cm = pd.DataFrame(confusion_matrix(y_test, predicted_labels),index=["flu",
"pia", "tru", "org", "gac", "voi"], columns=["flu", "pia", "tru", "org", "gac"
, "voi"])
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.0)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 18},fmt='g')# font size

plt.show()
```

In [ ]: