In [1]:
```
#!pip install pandas
```

In [2]:
```
#!pip install librosa
```

In [3]:
```
#!pip install numpy
```

In [4]:
```
#!pip install sklearn
```

In [5]:
```
#!pip install joblib
```

In [1]:
```
import pandas as pd
import librosa

#import audiosegment -- only if you need to segment audiofiles
#import moviepy.editor as mp -- only if you need to extract audio from videofiles


import numpy as np


from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


from joblib import dump,load
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
path = "C:\\Users\\Jayashree\\Downloads\\ICFOSS_Project\\ESC-50-master\\ESC-50-master\\meta\\esc50.csv"
```

In [4]:
```
data = pd.read_csv(path)
```

In [5]:
```
data.head()
```

Out[5]:

	filename	fold	target	category	esc10	src_file	take
0	1-100032-A-0.wav	1	0	dog	True	100032	A
1	1-100038-A-14.wav	1	14	chirping_birds	False	100038	A
2	1-100210-A-36.wav	1	36	vacuum_cleaner	False	100210	A
3	1-100210-B-36.wav	1	36	vacuum_cleaner	False	100210	B
4	1-101296-A-19.wav	1	19	thunderstorm	False	101296	A

Feature Extraction and Vectorization

In [6]:
```
#https://librosa.org/doc/0.9.1/feature.html#spectral-features
```

In [7]:
```python
def get_features(file):

    data, sample_rate = librosa.load(file, sr=None)

    result=np.array([])

    #https://librosa.org/doc/0.9.1/generated/librosa.feature.mfcc.html#librosa.feature.mfcc
    mfccs=np.mean(librosa.feature.mfcc(y=data, sr=sample_rate, n_mfcc=40).T, axis=0)
    result=np.hstack((result, mfccs))

    #https://librosa.org/doc/0.9.1/generated/librosa.feature.chroma_stft.html
    stft=np.abs(librosa.stft(data))
    chroma=np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis=0)
    result=np.hstack((result, chroma))

    #https://librosa.org/doc/main/generated/librosa.feature.melspectrogram.html
    mel=np.mean(librosa.feature.melspectrogram(data, sr=sample_rate).T,axis=0)
    result=np.hstack((result, mel))

    return result
```

Making the DataSet

In [8]:
```python
def make_train_data(df): # Loads Audio files from data set and makes train data
    audio_path = 'C:\\Users\\Jayashree\\Downloads\\ICFOSS_Project\\ESC-50-master\\ESC-50-master\\audio\\'
    X = []
    y = []
    for index, row in df.iterrows():
        outname = audio_path+row['filename']
        X.append(get_features(outname)) # extract spectral features and add to X for training
        y.append(row['category']) # add label to the corresponding clip
    return X, y
```

In [9]:
```
%%time
X, y = make_train_data(data) # making data set for training

#Wall time: 3min
```

```
Wall time: 2min 14s
```

In [13]:
```
dt = pd.DataFrame(data=[X,y]).T
```

In [14]:
```
dt['s_id'] = dt[1].factorize()[0]
```

In [15]:
```
dt.to_csv("transformed.csv")
```

In [16]:
```
dump(dt,'transformed.mdl')
```

Out[16]:
```
['transformed.mdl']
```

In [17]:
```
dt.head()
```

Out[17]:

	0	1	s_id
0	[-581.7399291992188, 8.207121849060059, -6.658...	dog	0
1	[-254.93630981445312, 85.8396224975586, -107.1...	chirping_birds	1
2	[-30.461212158203125, 102.50389862060547, -40....	vacuum_cleaner	2
3	[-31.114238739013672, 104.30037689208984, -43....	vacuum_cleaner	2
4	[-466.6221008300781, 144.2613983154297, 22.961...	thunderstorm	3

In [18]:
```
x_train,x_test,y_train,y_test = train_test_split(np.array(X), y, test_size=0.2, random_state=100) # split data for train
```

In [19]:
```
tst = pd.DataFrame(data=y_train) # just to monitor train and test dataset
#tst.iloc[:,-1].value_counts()
```

MLP Classifier

In [22]:
```
######################## Train using MLPClassifier ##############################
```

In [23]:
```
%%time
from sklearn.neural_network import MLPClassifier
mlpModel=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(3000,), learning_rate='adaptive', 
mlpModel.fit(X,y)
```

```
Wall time: 52.8 s
```

Out[23]:
```
MLPClassifier(alpha=0.01, batch_size=256, hidden_layer_sizes=(3000,),
              learning_rate='adaptive', max_iter=10000)
```

In [24]:
```
dump(mlpModel, 'mlpModel.mdl')
```

Out[24]:
```
['mlpModel.mdl']
```

In [25]:
```
t= load('mlpModel.mdl')
```

In [26]:
```
y_pred=t.predict(x_test)
```

In [27]:
```
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
Accuracy: 99.25%
```

DecisionTree Classifier

In [28]:
```
######################## Train using DecisionTreeClassifier ##############################
```

In [29]:
```
from sklearn.tree import DecisionTreeClassifier
dtreeModel = DecisionTreeClassifier()
```

In [30]:
```
%%time
dtreeModel.fit(x_train,y_train)
```

```
Wall time: 398 ms
```

Out[30]:
```
DecisionTreeClassifier()
```

In [31]:
```
dump(dtreeModel, 'dtreeModel.mdl')
```

Out[31]:
```
['dtreeModel.mdl']
```

In [32]:
```
y_pred=dtreeModel.predict(x_test)
```

In [33]:
```
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
Accuracy: 27.75%
```

Parameter Tuning

In [34]:
```
from sklearn.model_selection import GridSearchCV
```

In [35]:
```
parameters = {'criterion':["gini", "entropy", "log_loss"],
              'splitter':["best", "random"],
              'max_depth' : [2, 5, 7, 10],
              }

dTrModel = DecisionTreeClassifier()
```

In [36]:
```
clf0 = GridSearchCV(dTrModel, parameters)
```

In [37]:
```
%%time
clf0.fit(x_train, y_train)
```

```
Wall time: 21.5 s
```

Out[37]:
```
GridSearchCV(estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy', 'log_loss'],
                         'max_depth': [2, 5, 7, 10],
                         'splitter': ['best', 'random']})
```

In [38]:
```
y_hat = clf0.predict(x_test)
```

In [39]:
```
accuracy_score(y_true=y_test, y_pred=y_hat)
```

Out[39]:
```
0.2075
```

In [40]:
```
clf0.best_params_
```

Out[40]:
```
{'criterion': 'gini', 'max_depth': 10, 'splitter': 'random'}
```

SVM Classifier

In [41]:
```
######################## Train using SVM #############################
```

In [42]:
```
from sklearn.svm import SVC
svmModel = SVC(kernel = 'linear', random_state = 1)
```

In [43]:
```
%%time
svmModel.fit(x_train, y_train)
```

```
Wall time: 505 ms
```

Out[43]:
```
SVC(kernel='linear', random_state=1)
```

In [44]:
```
dump(svmModel, 'svmModel.mdl')
```

Out[44]:
```
['svmModel.mdl']
```

In [45]:
```
y_pred=svmModel.predict(x_test)
```

In [46]:
```
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
Accuracy: 44.25%
```

Parameter Tuning

In [47]:
```
parameters = {'C': [0.1, 1, 10, 100],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'gamma':['scale', 'auto'],
              'kernel': ['linear']}

svcModel = SVC()
```

In [48]:
```
clf = GridSearchCV(svcModel, parameters)
```

In [49]:
```
%%time
clf.fit(x_train, y_train)
```

```
Wall time: 14.7 s
```

Out[49]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 1, 10, 100], 'gamma': ['scale', 'auto'],
                         'kernel': ['linear']})
```

In [50]:
```
y_hat = clf.predict(x_test)
```

In [51]:
```
accuracy_score(y_true=y_test, y_pred=y_hat)
```
Out[51]:
```
0.4375
```

In [52]:
```
clf.best_params_
```
Out[52]:
```
{'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}
```

RandomForest Classifier

In [53]:
```
from sklearn.ensemble import RandomForestClassifier
rForestModel = RandomForestClassifier(max_depth=2, random_state=0)
```

In [54]:
```
%%time
rForestModel.fit(x_train, y_train)
```
```
Wall time: 429 ms
```
Out[54]:
```
RandomForestClassifier(max_depth=2, random_state=0)
```

In [55]:
```
y_pred=rForestModel.predict(x_test)
```

In [56]:
```
dump(rForestModel, 'rForestModel.mdl')
```
Out[56]:
```
['rForestModel.mdl']
```

In [57]:
```
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```
```
Accuracy: 15.50%
```

Parameter Tuning - Random Forest

In [58]:
```
parameters = {
             'criterion' : ["gini", "entropy", "log_loss"],
              'max_depth' : [2, 5, 7, 10],
               'n_estimators' : [100, 50, 25, 200]

             }
```

In [59]:
```
ranFrstModel = RandomForestClassifier()
```

In [60]:
```
clf2 = GridSearchCV(ranFrstModel, parameters)
```

In [61]:
```
%%time
clf2.fit(x_train, y_train)
```

```
Wall time: 6min 10s
```

Out[61]:
```
GridSearchCV(estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy', 'log_loss'],
                         'max_depth': [2, 5, 7, 10],
                         'n_estimators': [100, 50, 25, 200]})
```

In [62]:
```
y_pred=clf2.predict(x_test)
```

In [63]:
```
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
Accuracy: 46.50%
```

In [64]:
```
clf2.best_params_
```

Out[64]:
```
{'criterion': 'gini', 'max_depth': 10, 'n_estimators': 200}
```

In [73]:
```
dump(clf2, 'rForestModel_final.mdl')
```

Out[73]:
```
['rForestModel_final.mdl']
```

KNN Classifier

In [20]:
```
from sklearn.neighbors import KNeighborsClassifier
knnModel = KNeighborsClassifier()
```

In [21]:
```
%%time
knnModel.fit(x_train, y_train)
```

```
Wall time: 8.33 ms
```

Out[21]:
```
KNeighborsClassifier()
```

In [22]:
```
dump(knnModel, 'knnModel.mdl')
```

Out[22]:
```
['knnModel.mdl']
```

In [23]:
```
y_pred=knnModel.predict(x_test)
```

In [25]:
```
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
accuracy
```

Out[25]:
```
0.2875
```

In [32]:
```
leaf_size = list(range(1,50))
n_neighbors = list(range(1,30))
p=[1,2]
```

In [33]:
```
hyperparameters = dict(leaf_size=leaf_size, n_neighbors=n_neighbors, p=p)
```

In [35]:
```
knn_2 = KNeighborsClassifier()
```

In [36]:
```
clf_knn = GridSearchCV(knn_2, hyperparameters, cv=10)
```

In [40]:
```
%%time
best_model = clf_knn.fit(x_train,y_train)
```

```
Wall time: 11min 2s
```

In [41]:
```
dump(knnModel, 'knnModel_final.mdl')
```

Out[41]:
```
['knnModel_final.mdl']
```

In [45]:
```
y_pred=clf_knn.predict(x_test)
```

In [46]:
```
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
accuracy
```

Out[46]:
```
0.43
```

SGD Classifier

In [26]:
```
from sklearn.linear_model import SGDClassifier
sgdModel = SGDClassifier()
```

In [27]:
```
%%time
sgdModel.fit(x_train, y_train)
```

```
Wall time: 630 ms
```

Out[27]:
```
SGDClassifier()
```

In [30]:
```
y_pred=sgdModel.predict(x_test)
```

In [31]:
```
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
accuracy
```

Out[31]:
```
0.2075
```

In [47]:
```
parameters = {
'loss' : ['hinge', 'log_loss', 'log', 'modified_huber', 'squared_hinge', 'perceptron', 'squared_error', 'huber', 'epsilo

}
```

In [49]:
```
clf_sgd = GridSearchCV(SGDClassifier(), parameters)
```

In [51]:
```
%%time
clf_sgd.fit(x_train, y_train)
```

```
Wall time: 49.7 s
```

Out[51]:
```
GridSearchCV(estimator=SGDClassifier(),
             param_grid={'loss': ['hinge', 'log_loss', 'log', 'modified_huber',
                                  'squared_hinge', 'perceptron',
                                  'squared_error', 'huber',
                                  'epsilon_insensitive',
                                  'squared_epsilon_insensitive']})
```

In [52]:
```
y_pred=clf_sgd.predict(x_test)
```

In [53]:
```
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
accuracy
```

Out[53]:
```
0.2275
```

Testing the Models

In [65]:
```
######################### External Clip Test Function ###################
```

In [66]:
```
def test(file,model): # split and max count
    #sample_rate=44100
    result = []
    X = []
    outname_silence = file
    X.append(get_features(outname_silence))
    y = model.predict(X)
    return y
```

In [67]:
```
path = 'C:\\Users\\Jayashree\\Downloads\\ICFOSS_Project\\ESC-50-master\\ESC-50-master\\test\\'
```

In [68]:
```
result =  test(path+'t3.wav',mlpModel)
```

In [69]:
```
result[0]
```

Out[69]:
```
'clapping'
```

In [70]:
```
result =  test(path+'t3.wav',rForestModel)
result[0]
```

Out[70]:
```
'sea_waves'
```

In [74]:
```
result =  test(path+'t3.wav',clf2)
result[0]
```

Out[74]:
```
'clapping'
```

In [71]:
```
result =  test(path+'t3.wav',svmModel)
result[0]
```

Out[71]:
```
'clapping'
```

In [72]:
```
result =  test(path+'t3.wav',dtreeModel)
result[0]
```

Out[72]:
```
'clapping'
```