main.py

```
# Basic Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
pd.plotting.register_matplotlib_converters()
#%matplotlib inline
import seaborn as sns

# Specific Libraries
import os
import librosa
import librosa.display

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

from tqdm import tqdm, trange
from tqdm.auto import tqdm
```

```
import tensorflow as tf

from tensorflow.keras.utils import to_categorical

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense , Activation , Dropout

import IPython.display as ipd


df = pd.read_csv("C:/Shivani/Project-
Audio_Classification/UrbanSound8K/UrbanSound8K.csv")

df.head()


dat1, sampling_rate1 = librosa.load('C:/Shivani/Project-
Audio_Classification/UrbanSound8K/childern_playing.wav')

dat2, sampling_rate2 = librosa.load('C:/Shivani/Project-
Audio_Classification/UrbanSound8K/dog_bark.wav')


plt.figure(figsize=(20, 20))

D = librosa.amplitude_to_db(np.abs(librosa.stft(dat1)), ref=np.max)

plt.subplot(4, 2, 1)

librosa.display.specshow(D, y_axis='linear')

plt.colorbar(format='%+2.0f dB')

plt.title('Linear-frequency power spectrogram')


plt.figure(figsize=(20, 20))

D = librosa.amplitude_to_db(np.abs(librosa.stft(dat2)), ref=np.max)
```

```
plt.subplot(4, 2, 1)

librosa.display.specshow(D, y_axis='linear')

plt.colorbar(format='%+2.0f dB')

plt.title('Linear-frequency power spectrogram')


arr = np.array(df["slice_file_name"])

fold = np.array(df["fold"])

cla = np.array(df["class"])


for i in range(192, 197, 2):

    path = 'C:/Shivani/Project-Audio_Classification/UrbanSound8K/audio/fold' + str(fold[i]) +
'/' + arr[i]

    data, sampling_rate = librosa.load(path)

    plt.figure(figsize=(10, 10))

    D = librosa.amplitude_to_db(np.abs(librosa.stft(data)), ref=np.max)

    plt.subplot(4, 2, 1)

    librosa.display.specshow(D, y_axis='linear')

    plt.colorbar(format='%+2.0f dB')

    plt.title(cla[i])


def features_extract(file):

    sample,sample_rate = librosa.load(file_name,res_type='kaiser_fast')

    feature = librosa.feature.mfcc(y=sample,sr=sample_rate,n_mfcc=50)

    scaled_feature = np.mean(feature.T,axis=0)

    return scaled_feature
```

```
extracted = []
path = 'C:/Shivani/Project-Audio_Classification/UrbanSound8K/audio/'

for index_num,row in tqdm(df.iterrows()):
    file_name = os.path.join(os.path.abspath(path),'fold'+str(row["fold"])+'/',str(row['slice_file_name']))
    final_class_labels = row['class']
    data= features_extract(file_name)
    extracted.append([data,final_class_labels])

ext_df = pd.DataFrame(extracted,columns=['feature','class'])
ext_df

x = np.array(ext_df['feature'].tolist())
y = np.array(ext_df['class'].tolist())

le = LabelEncoder()

y = to_categorical(le.fit_transform(y))

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1, random_state = 42)

print("Number of training samples = ", x_train.shape[0])
print("Number of testing samples = ",x_test.shape[0])
```

```
num_labels = y.shape[1]
model = Sequential()

model.add(Dense(128, input_shape=(50,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(128))

model.add(Dense(num_labels))
model.add(Activation('softmax'))
model.summary()

model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss=tf.keras.losses.CategoricalCrossentropy(),
              metrics=['accuracy'])

model.fit(
```

```
        x_train,
        y_train,
        batch_size=60,
        epochs=200,
        validation_data=(x_test, y_test),
       )

model.save('C:/Shivani/Project-Audio_Classification/model.h5')

test_accuracy = model.evaluate(x_test, y_test, verbose=0)
print(test_accuracy[1])

def extract_feature(file_name):
    audio_data, sample_rate = librosa.load(file_name, res_type='kaiser_fast')
    fea = librosa.feature.mfcc(y=audio_data, sr=sample_rate, n_mfcc=50)
    scaled = np.mean(fea.T,axis=0)
    return np.array([scaled])

def print_prediction(file_name):
    pred_fea = extract_feature(file_name)
    pred_vector = np.argmax(model.predict(pred_fea), axis=-1)
    pred_class = le.inverse_transform(pred_vector)
    print("The predicted class is:", pred_class[0], '\n')
```

df

print_prediction('C:/Shivani/Project-Audio_Classification/UrbanSound8K/audio/fold2/100652-3-0-2.wav')

ipd.Audio('C:/Shivani/Project-Audio_Classification/UrbanSound8K/audio/fold2/100652-3-0-2.wav')

classes.py

```
import numpy as np

# Load the integer labels for the training dataset

train_labels = [0,1,2,3,4,5,6,7,8,9]  # replace [...] with your integer labels


# Define the corresponding string names for the labels

class_names = ['air_conditioner', 'car_horn', 'children_playing', 'dog_bark', 'drilling', 'engine_idling', 'gun_shot', 'jackhammer', 'siren', 'street_music']  # replace with your class names


# Create a dictionary to map integer labels to string names

label_map = dict(zip(range(len(class_names)), class_names))


# Convert the integer labels to string names

train_class_names = np.vectorize(label_map.get)(train_labels)
```

```
# Save the class names to a numpy array file
save_path = 'C:/Shivani/Project-Audio_Classification/classes.npy'
np.save(save_path, np.array(train_class_names))
```

GUI.py

```
import tkinter as tk
from tkinter import filedialog
import os
import librosa
import librosa.display
import numpy as np
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import load_model
from sklearn.preprocessing import LabelEncoder

# Load trained model
model = load_model('C:/Shivani/Project-Audio_Classification/model.h5')

# Load label encoder
le = LabelEncoder()
le.classes_ = np.load('C:/Shivani/Project-Audio_Classification/classes.npy')

# Define function to extract features from audio file
```

```
def extract_feature(file_name):
    audio_data, sample_rate = librosa.load(file_name, res_type='kaiser_fast')
    fea = librosa.feature.mfcc(y=audio_data, sr=sample_rate, n_mfcc=50)
    scaled = np.mean(fea.T, axis=0)
    return np.array([scaled])

# Define function to predict class of audio file
def predict_class(file_name):
    pred_fea = extract_feature(file_name)
    pred_vector = np.argmax(model.predict(pred_fea), axis=-1)
    pred_class = le.inverse_transform(pred_vector)
    return pred_class[0]

# Define function to open file dialog and select audio file
def browse_file():
    global file_path
    file_path = filedialog.askopenfilename()
    file_path_label.config(text=file_path)

# Define function to classify selected audio file and display result
def classify_file():
    if file_path:
        pred_class = predict_class(file_path)
        result_label.config(text="Predicted class: {}".format(pred_class))
```

```
    else:
        result_label.config(text="Please select an audio file first.")

# Create GUI window
root = tk.Tk()
root.title("Audio Classification")

# Load background image
bg_img = tk.PhotoImage(file="C:/Shivani_Proj_Doc/background.png")

# Create a Label widget to display the background image
bg_label = tk.Label(root, image=bg_img)
bg_label.place(x=0, y=0, relwidth=1, relheight=1)

# Set styling options
root.configure(bg='#f5f5f5')
root.geometry('700x350')
root.resizable(False, False)

# Create frame for file selection and classification
frame = tk.Frame(root, bg='#f5f5f5')
frame.pack(pady=40)

# Create file selection button
```

```
browse_button = tk.Button(root, text="Select audio file", command=browse_file,
bg='#4682b4', fg='white', font=('Arial', 12, 'bold'), pady=10)

browse_button.pack(padx=10)


# Create label to display selected file path

file_path_label = tk.Label(root, text="" , bg='#f5f5f5', font=('Arial', 12))

file_path_label.pack()


# Create classification button

classify_button = tk.Button(root, text="Classify audio file", command=classify_file,
bg='#4682b4', fg='white', font=('Arial', 12, 'bold'), pady=10)

classify_button.pack(pady=10)


# Create label to display classification result

result_label = tk.Label(root, text="", bg='#f5f5f5', font=('Arial', 16,'bold'))

result_label.pack()


# Start GUI event loop

root.mainloop()
```