```
# File: emotion_recognition.py

import numpy as np
import librosa
import librosa.display
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout
from tensorflow.keras.utils import to_categorical

# Step 1: Generate Synthetic Audio Features
def generate_synthetic_audio_features(num_samples=1000, num_mfcc=13, max_length=50):
    np.random.seed(42)
    features = []
    labels = []

    emotions = ['happy', 'sad', 'angry', 'neutral']  # Emotion categories
    for _ in range(num_samples):
        mfcc = np.random.rand(num_mfcc, max_length)  # Random MFCC-like feature matrix
        emotion = np.random.choice(emotions)  # Random emotion label
        features.append(mfcc)
        labels.append(emotion)

    features = np.array(features)
    labels = np.array(labels)
    return features, labels

# Step 2: Preprocess Data
def preprocess_data(features, labels):
    # Normalize features
    features = features / np.max(features)

    # Encode labels to integers
    encoder = LabelEncoder()
    labels = encoder.fit_transform(labels)
    labels = to_categorical(labels)  # One-hot encoding for multi-class classification

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)
    return X_train, X_test, y_train, y_test, encoder

# Step 3: Build the CNN Model
def build_model(input_shape, num_classes):
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
        MaxPooling2D((2, 2)),
        Dropout(0.2),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Dropout(0.2),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.3),
        Dense(num_classes, activation='softmax')
    ])
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model

# Step 4: Train and Evaluate the Model
def train_and_evaluate_model(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))
    test_loss, test_accuracy = model.evaluate(X_test, y_test)
    print(f"Test Accuracy: {test_accuracy:.2f}")
    return model

# Main Function
if __name__ == "__main__":
    # Generate synthetic data
    features, labels = generate_synthetic_audio_features()

    # Preprocess the data
    X_train, X_test, y_train, y_test, encoder = preprocess_data(features, labels)

    # Reshape data for CNN input (Add channel dimension)
```

```
X_train = X_train[..., np.newaxis]
X_test = X_test[..., np.newaxis]

# Build model
input_shape = X_train.shape[1:]  # (num_mfcc, max_length, 1)
num_classes = len(encoder.classes_)
model = build_model(input_shape, num_classes)

# Train and evaluate model
trained_model = train_and_evaluate_model(model, X_train, X_test, y_train, y_test)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 4s 38ms/step - accuracy: 0.2625 - loss: 1.3971 - val_accuracy: 0.2850 - val_loss: 1.3866
Epoch 2/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.2800 - loss: 1.3872 - val_accuracy: 0.2800 - val_loss: 1.3874
Epoch 3/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.2534 - loss: 1.3869 - val_accuracy: 0.2800 - val_loss: 1.3836
Epoch 4/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - accuracy: 0.2718 - loss: 1.3826 - val_accuracy: 0.2800 - val_loss: 1.3857
Epoch 5/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - accuracy: 0.3067 - loss: 1.3793 - val_accuracy: 0.2800 - val_loss: 1.3857
Epoch 6/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - accuracy: 0.2766 - loss: 1.3844 - val_accuracy: 0.2850 - val_loss: 1.3856
Epoch 7/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.3155 - loss: 1.3795 - val_accuracy: 0.2800 - val_loss: 1.3865
Epoch 8/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.2866 - loss: 1.3801 - val_accuracy: 0.2800 - val_loss: 1.3839
Epoch 9/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.2806 - loss: 1.3769 - val_accuracy: 0.2800 - val_loss: 1.3853
Epoch 10/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.3019 - loss: 1.3767 - val_accuracy: 0.2800 - val_loss: 1.3843
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.2894 - loss: 1.3849
Test Accuracy: 0.28
```