SPEECH EMOTION RECOGNITION.

```
# Import necessary libraries
import os
import pandas as pd
import librosa
import librosa.display
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import OneHotEncoder
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout

# Load dataset
paths = []
labels = []
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        paths.append(os.path.join(dirname, filename))
        label = filename.split('_')[-1]
        label = label.split('.')[0]
        labels.append(label.lower())

print('Dataset is Loaded')

# Create a Pandas DataFrame
df = pd.DataFrame()
df['speech'] = paths
df['label'] = labels

# Display the first few rows of the DataFrame
print(df.head())

# Display the label distribution
print(df['label'].value_counts())
sns.countplot(x='label', data=df)

# Define functions for waveform and spectrogram plots
def waveplot(data, sr, emotion):
    plt.figure(figsize=(10,4))
    plt.title(emotion, size=20)
    plt.plot(data)
    plt.xlabel('Time')
    plt.ylabel('Amplitude')
    plt.show()

def spectogram(data, sr, emotion):
    x = librosa.stft(data)
    xdb = librosa.amplitude_to_db(abs(x))
    plt.figure(figsize=(12,4))
    plt.title(emotion, size=20)
    librosa.display.specshow(xdb, sr=sr, x_axis='time', y_axis='hz')
    plt.colorbar()
    plt.show()

# Load and visualize audio data for each emotion
emotions = ['fear', 'angry', 'disgust', 'happy', 'sad', 'ps', 'neutral']
```

```
for emotion in emotions:
    path = np.array(df['speech'][df['label']==emotion])[0]
    data, sampling_rate = librosa.load(path)
    waveplot(data, sampling_rate, emotion)
    spectogram(data, sampling_rate, emotion)
    # Audio(path)  # Not sure what this line does, so I commented it out

# Extract MFCC features from audio data
def extract_mfcc(filename):
    y, sr = librosa.load(filename, duration=3, offset=0.5)
    mfcc = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40).T, axis=0)
    return mfcc

x_mfcc = df['speech'].apply(lambda x: extract_mfcc(x))
x = [x for x in x_mfcc]
x = np.array(x)
x = np.expand_dims(x, -1)

# One-hot encode labels
enc = OneHotEncoder()
y = enc.fit_transform(df[['label']])
y = np.array(y)

# Define the LSTM model
model = Sequential([
    LSTM(123, return_sequences=False, input_shape=(40,1)),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(7, activation='softmax')
])

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Print the model summary
print(model.summary())

# Train the model
history = model.fit(x, y, validation_split=0.2, epochs=100, batch_size=512, shuffle=True

# Plot the training and validation accuracy
epochs = list(range(100))
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
plt.plot(epochs, acc, label='train accuracy')
plt.plot(epochs, val_acc, label='val accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()

# Plot the training and validation loss
epochs = list(range(100))
loss = history.history['loss']
val_loss = history.history['val_loss']
plt.plot(epochs, loss, label='train loss')
plt.plot(epochs, val_loss, label='val loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

Speech Recognition Model Documentation

Overview

This project aims to build a speech recognition model that classifies audio recordings into different emotional categories. The model uses audio features extracted from the recordings to train a Long Short-Term Memory (LSTM) neural network.

Project Structure

The project consists of several key components:

1. **Data Loading and Preparation**
2. **Audio Visualization**
3. **Feature Extraction**
4. **Model Definition and Training**
5. **Performance Evaluation**

1. Data Loading and Preparation

The first step in the project involves loading the audio dataset from a specified directory. The audio files are traversed using the `os.walk` method, and the paths and labels are extracted based on the filenames.

```
paths = []
labels = []
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        paths.append(os.path.join(dirname, filename))
        label = filename.split('_')[-1]
        label = label.split('.')[0]
        labels.append(label.lower())
```

- **Paths**: List of audio file paths.
- **Labels**: List of corresponding emotional labels derived from the filenames.

A Pandas DataFrame is created to store the paths and labels for easier manipulation and analysis.

2. Audio Visualization

The project includes functions to visualize the audio data through waveform and spectrogram plots. The `waveplot` function displays the amplitude of the audio signal over time, while the `spectogram` function shows the frequency spectrum.

```
def waveplot(data, sr, emotion):
    # Function to plot the waveform of the audio data
    ...

def spectogram(data, sr, emotion):
    # Function to plot the spectrogram of the audio data
    ...
```

These visualizations help in understanding the characteristics of the audio signals associated with different emotions.

3. Feature Extraction

To prepare the audio data for training, Mel-frequency cepstral coefficients (MFCCs) are extracted. MFCCs are commonly used features in speech and audio processing that capture the short-term power spectrum of sound.

```
def extract_mfcc(filename):
    y, sr = librosa.load(filename, duration=3, offset=0.5)
    mfcc = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40).T, axis=0)
    return mfcc
```

- **Input**: Path to the audio file.
- **Output**: A NumPy array of MFCC features.

4. Model Definition and Training

The core of the project is the LSTM model, which is defined using Keras. The model architecture consists of several layers:

- An LSTM layer for sequence processing.
- Dense layers for classification.
- Dropout layers to prevent overfitting.

```
model = Sequential([
    LSTM(123, return_sequences=False, input_shape=(40,1)),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(7, activation='softmax')
])
```

The model is compiled with categorical crossentropy loss and the Adam optimizer, and trained on the extracted features.

5. Performance Evaluation

The model's performance is evaluated through accuracy and loss metrics during training. Plots are generated to visualize the training and validation accuracy and loss over epochs.

```
# Plotting accuracy and loss
plt.plot(epochs, acc, label='train accuracy')
plt.plot(epochs, val_acc, label='val accuracy')
...
```

Conclusion

This project demonstrates the process of building a speech recognition model using audio data. By extracting meaningful features and training a robust LSTM model, the project aims to classify emotions in speech effectively. Future work could involve experimenting with different model architectures, hyperparameter tuning, and expanding the dataset for improved performance