Speech Commands Dataset Analysis and Model Development

1. Install Required Libraries

```
!pip install pydub

!apt-get install ffmpeg
```

2. Load the Speech Commands Dataset

```
import tensorflow as tf

import tensorflow_datasets as tfds

# Load the speech commands dataset

dataset, info = tfds.load('speech_commands', with_info=True, as_supervised=True)

train_data, test_data = dataset['train'], dataset['test']
```

3. Explore Dataset Info

```
print(info)
```

4. Check the Shape of the Data

```
for audio, label in train_data.take(1):

    print(f'Audio shape: {audio.shape}, Label: {label.numpy()}')
```

5. Get Unique Labels

```
import numpy as np


# Get the unique labels from the training dataset

label_names = np.unique([label.numpy() for _, label in train_data])

print('Labels in the dataset:', label_names)
```

6. Squeeze Audio Data

```
for audio, label in train_data.take(1):

    squeezed_audio = tf.squeeze(audio)

    print(f'Squeezed audio shape: {squeezed_audio.shape}, Label: {label.numpy()}')
```

7. Display Sample Audio

```
import IPython.display as display


# Get a sample audio and label from the training set
audio, label = next(iter(train_data))
display.display(display.Audio(audio, rate=16000))
```

8. Plot Waveform

```
import matplotlib.pyplot as plt


# Take a sample from the dataset
for audio, label in train_data.take(1):
    squeezed_audio = tf.squeeze(audio)
    audio_data = squeezed_audio.numpy()
    time = np.arange(audio_data.shape[0]) / 16000  # Assuming a sample rate of 16 kHz
    plt.figure(figsize=(10, 4))
    plt.plot(time, audio_data)
    plt.title(f'Waveform of Label: {label.numpy()}')
    plt.xlabel('Time (seconds)')
    plt.ylabel('Amplitude')
    plt.xlim(0, time[-1])  # Set x-axis limit to the duration of the audio
    plt.grid()
    plt.show()
```

9. Define Functions to Plot Waveform and Spectrogram

```
def plot_wave(waveform, label):
    plt.figure(figsize=(10, 3))
    plt.title(label)
    plt.plot(waveform)
    plt.xlim([0, 16000])  # Limit x-axis to the duration of audio
    plt.ylim([-1, 1])  # Amplitude range
    plt.xlabel('Time')
    plt.ylabel('Amplitude')
    plt.grid(True)


def get_spectrogram(waveform):
    waveform = tf.cast(waveform, tf.float32)
    spectrogram = tf.signal.stft(waveform, frame_length=255, frame_step=128)
    spectrogram = tf.abs(spectrogram)
    return spectrogram[..., tf.newaxis]
```

10. Plot Spectrogram

```
def plot_spectrogram(spectrogram, label):
    spectrogram = np.squeeze(spectrogram, axis=-1)
    log_spec = np.log(spectrogram.T + np.finfo(float).eps)
    plt.figure(figsize=(10, 3))
    plt.title(label)
    plt.imshow(log_spec, aspect='auto', origin='lower')
    plt.colorbar(format='%+2.0f dB')
    plt.xlabel('Time')
    plt.ylabel('Frequency')
```

11. Create Spectrogram Dataset

```
def get_spectrogram_dataset(dataset):
    dataset = dataset.map(
        lambda x, y: (get_spectrogram(x), y),
        num_parallel_calls=tf.data.AUTOTUNE
    )
    return dataset
```

12. Build the Model

```
def get_model(input_shape, num_labels):
    model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=input_shape),
        tf.keras.layers.Resizing(64, 64),
        tf.keras.layers.Normalization(),
        tf.keras.layers.Conv2D(64, 3, activation='relu'),
        tf.keras.layers.Conv2D(128, 3, activation='relu'),
        tf.keras.layers.MaxPooling2D(),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(256, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(num_labels, activation='softmax')
    ])
    model.summary()
    return model
```

13. Create and Compile the Model

```
input_shape = (64, 64, 1)
num_labels = len(info.features['label'].names)
model = get_model(input_shape, num_labels)
model.compile(
    optimizer=tf.keras.optimizers.Adam(),
    loss=tf.keras.losses...
)
```

14. Downsize the Dataset

```
# Function to filter and downsize the dataset
```

```
def downsize_dataset(dataset, samples_per_label, label_names):
    ...
```

15. Count Samples in the Datasets

```
# Count samples in the downsized datasets
train_counts = count_samples(downsized_train_set, label_names)
val_counts = count_samples(downsized_val_set, label_names)
print('Training set distribution:', train_counts)
print('Validation set distribution:', val_counts)
```

16. Check Shapes of Labels

```
# Check the shape of labels in the original train set
for audio, label in train_data.take(5):
    print('Original Audio shape:', audio.shape)
    print('Original Label shape:', label.shape)
```

17. Create Spectrogram Dataset from Audio Data

```
# Applying the function on the audio dataset
train_set = get_spectrogram_dataset(downsized_train_set)
validation_set = get_spectrogram_dataset(downsized_val_set)
```

18. Verify Shapes of Example Batches

```
# Verify the shapes of an example batch from the training set
for spectrogram, label in train_set.take(1):
    print('Spectrogram shape:', spectrogram.shape)
    print('Label shape:', label.shape)
```