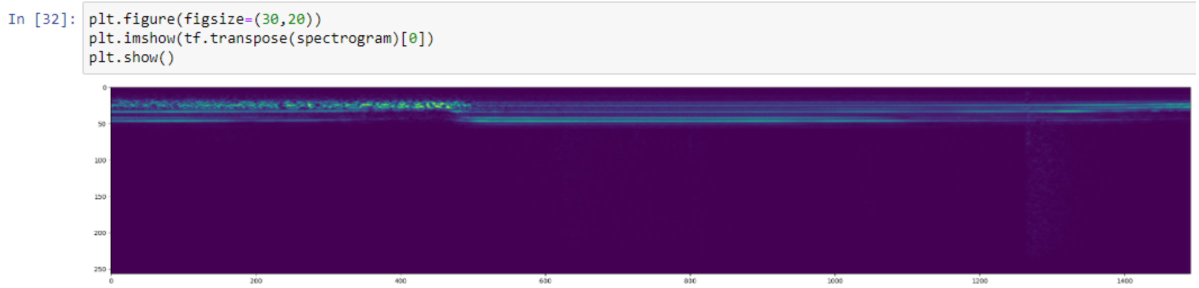
2.3 SNAPSHOTS OF PROJECT

The following libraries are used in this method

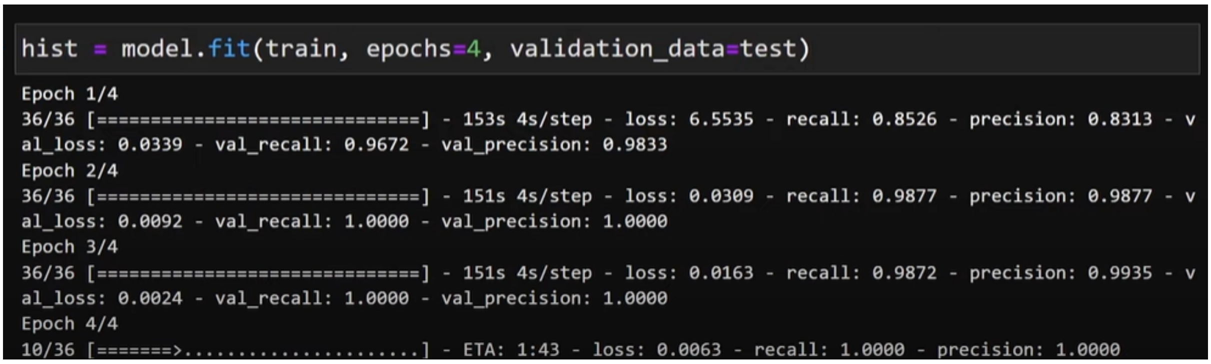
- Keras
- Scipy
- Numpy
- Scikit-learn

SPECTROGRAM:

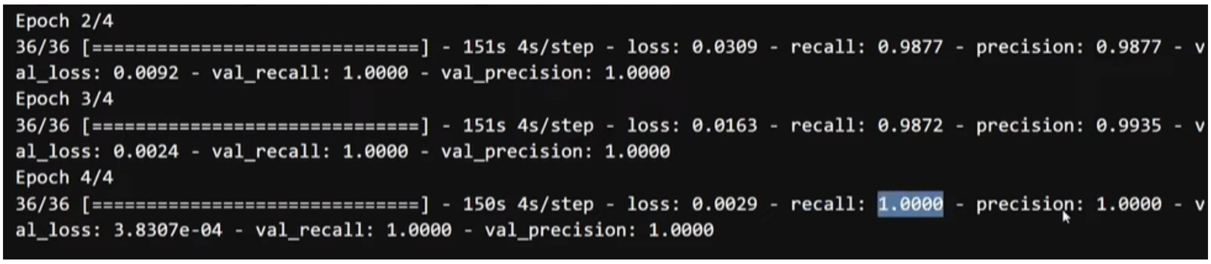


```
In [32]: plt.figure(figsize=(30,20))
         plt.imshow(tf.transpose(spectrogram)[0])
         plt.show()
```

MODEL TRAINING AND HISTOGRAMS:

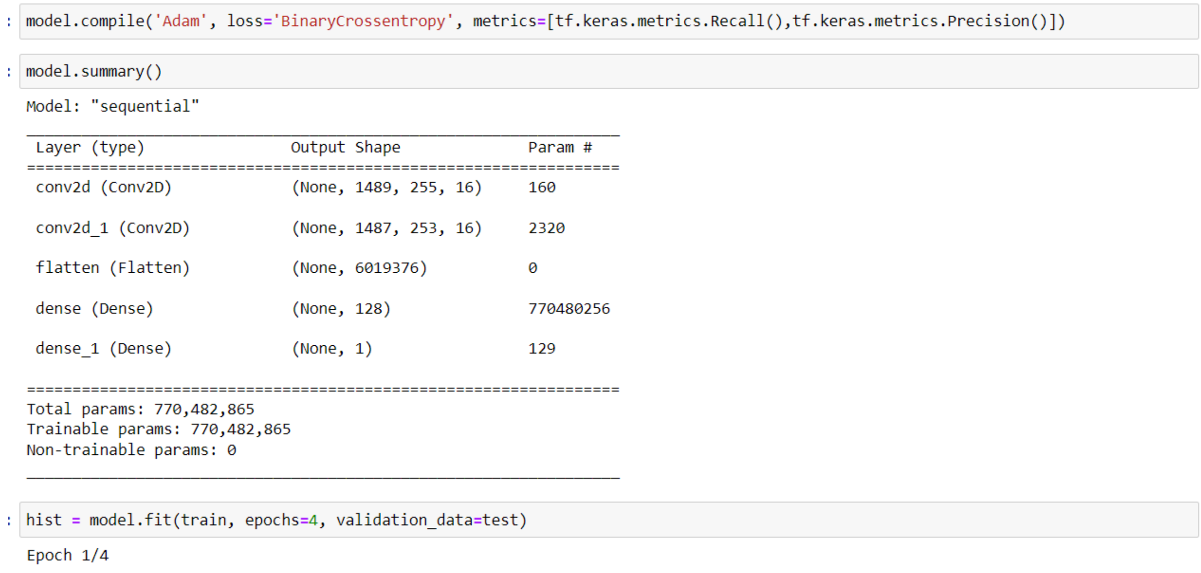


```
hist = model.fit(train, epochs=4, validation_data=test)
Epoch 1/4
36/36 [==============================] - 153s 4s/step - loss: 6.5535 - recall: 0.8526 - precision: 0.8313 - v
al_loss: 0.0339 - val_recall: 0.9672 - val_precision: 0.9833
Epoch 2/4
36/36 [==============================] - 151s 4s/step - loss: 0.0309 - recall: 0.9877 - precision: 0.9877 - v
al_loss: 0.0092 - val_recall: 1.0000 - val_precision: 1.0000
Epoch 3/4
36/36 [==============================] - 151s 4s/step - loss: 0.0163 - recall: 0.9872 - precision: 0.9935 - v
al_loss: 0.0024 - val_recall: 1.0000 - val_precision: 1.0000
Epoch 4/4
10/36 [=======>......................] - ETA: 1:43 - loss: 0.0063 - recall: 1.0000 - precision: 1.0000
```



```
Epoch 2/4
36/36 [==============================] - 151s 4s/step - loss: 0.0309 - recall: 0.9877 - precision: 0.9877 - v
al_loss: 0.0092 - val_recall: 1.0000 - val_precision: 1.0000
Epoch 3/4
36/36 [==============================] - 151s 4s/step - loss: 0.0163 - recall: 0.9872 - precision: 0.9935 - v
al_loss: 0.0024 - val_recall: 1.0000 - val_precision: 1.0000
Epoch 4/4
36/36 [==============================] - 150s 4s/step - loss: 0.0029 - recall: 1.0000 - precision: 1.0000 - v
al_loss: 3.8307e-04 - val_recall: 1.0000 - val_precision: 1.0000
```

DEEP LEARNING LAYERS:



```
: model.compile('Adam', loss='BinaryCrossentropy', metrics=[tf.keras.metrics.Recall(),tf.keras.metrics.Precision()])
```

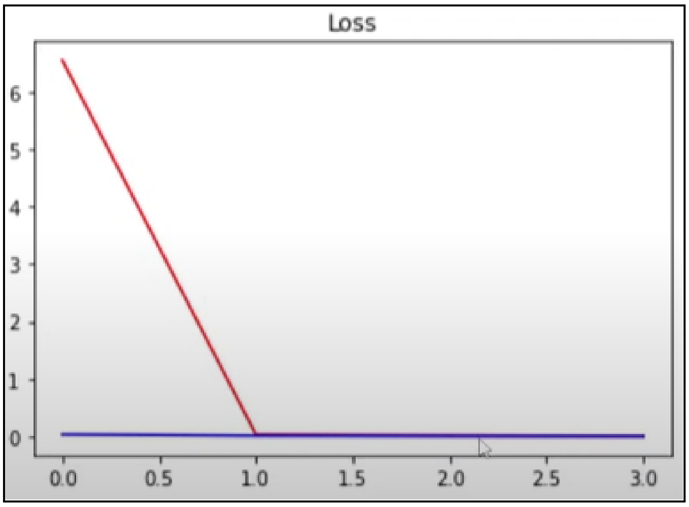
```
: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 1489, 255, 16)     160

 conv2d_1 (Conv2D)           (None, 1487, 253, 16)     2320

 flatten (Flatten)           (None, 6019376)           0

 dense (Dense)               (None, 128)               770480256

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 770,482,865
Trainable params: 770,482,865
Non-trainable params: 0
_________________________________________________________________
```

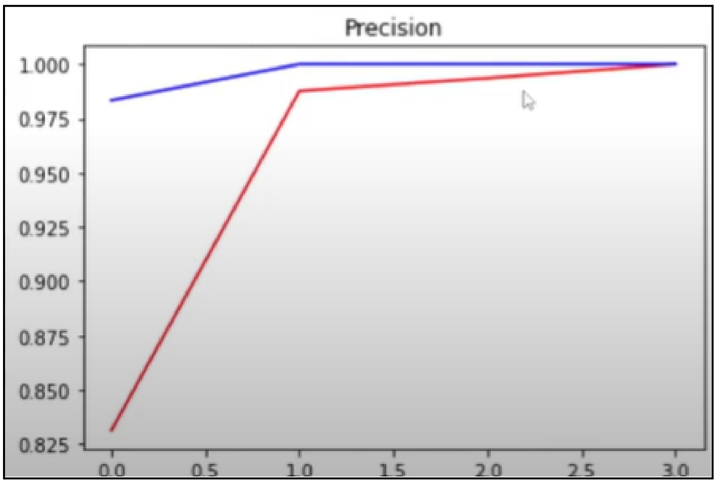
```
: hist = model.fit(train, epochs=4, validation_data=test)
```

```
Epoch 1/4
```

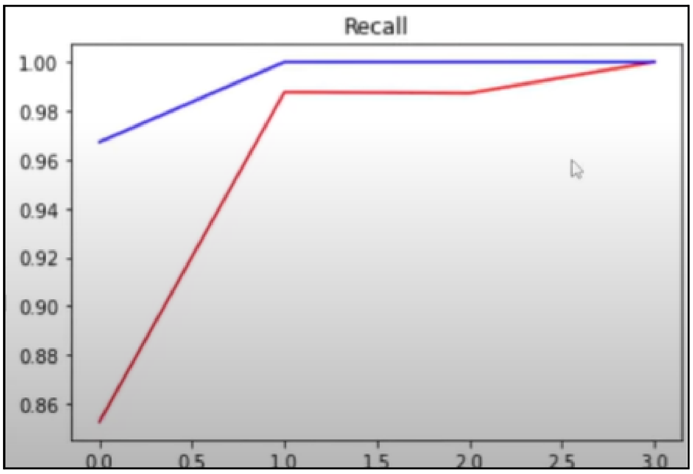
LOSS PLOT:

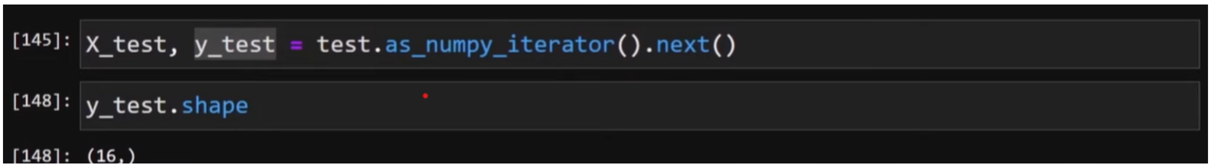


PRECISION PLOT:

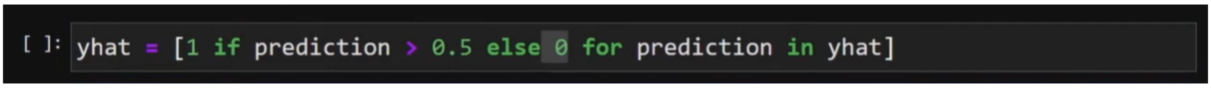


RECALL PLOT:

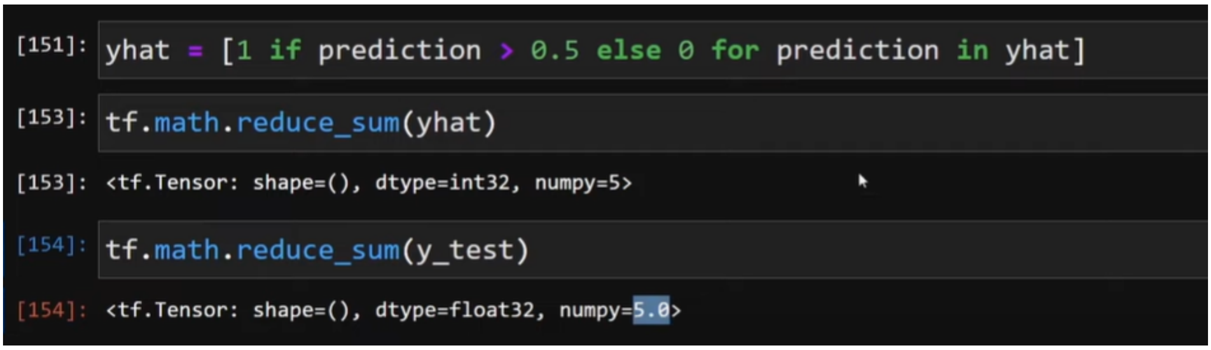




```
[145]: X_test, y_test = test.as_numpy_iterator().next()

[148]: y_test.shape

[148]: (16,)
```

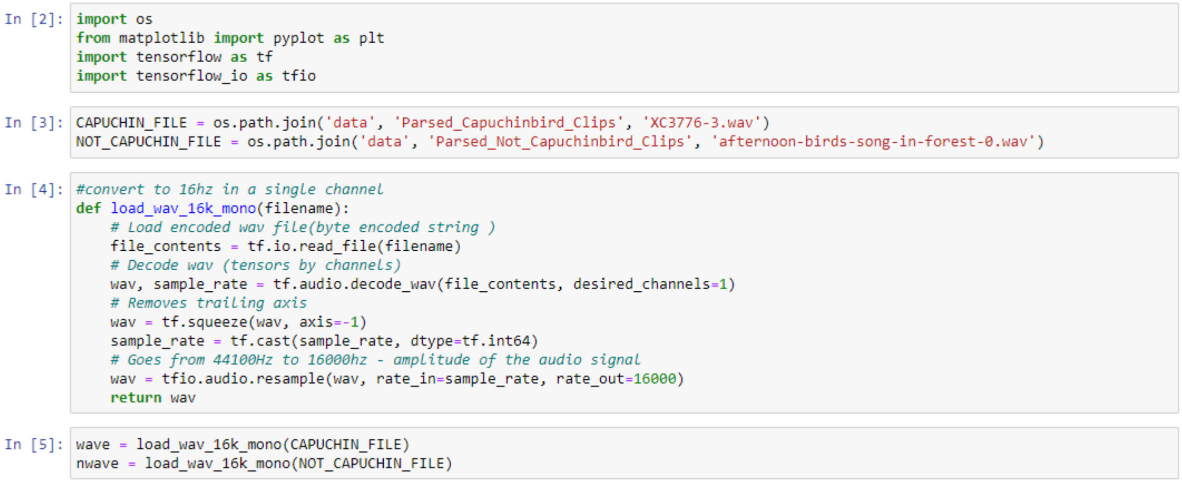


```
[ ]: yhat = [1 if prediction > 0.5 else 0 for prediction in yhat]
```

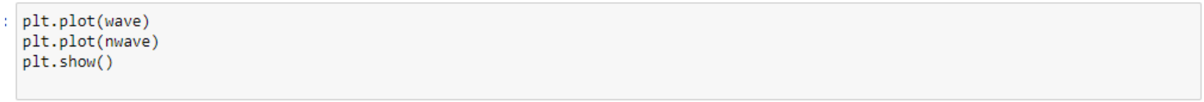


```
[151]: yhat = [1 if prediction > 0.5 else 0 for prediction in yhat]

[153]: tf.math.reduce_sum(yhat)

[153]: <tf.Tensor: shape=(), dtype=int32, numpy=5>

[154]: tf.math.reduce_sum(y_test)

[154]: <tf.Tensor: shape=(), dtype=float32, numpy=5.0>
```

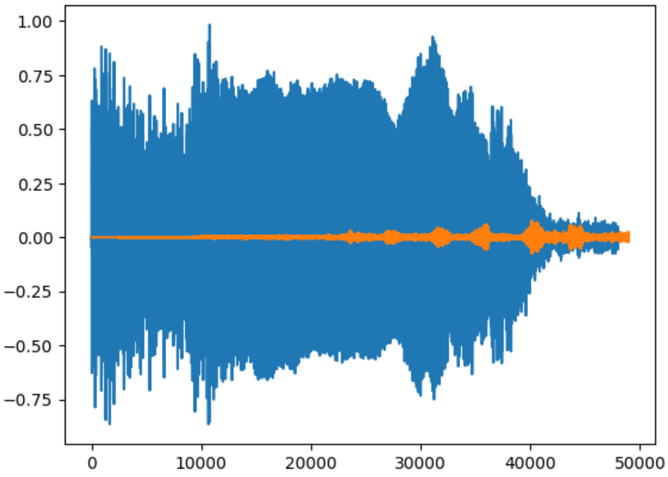
2.4 CODE SNIPPETS

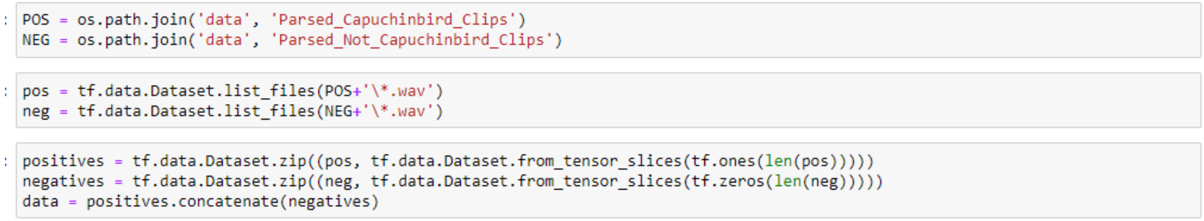


```
In [2]: import os
        from matplotlib import pyplot as plt
        import tensorflow as tf
        import tensorflow_io as tfio

In [3]: CAPUCHIN_FILE = os.path.join('data', 'Parsed_Capuchinbird_Clips', 'XC3776-3.wav')
        NOT_CAPUCHIN_FILE = os.path.join('data', 'Parsed_Not_Capuchinbird_Clips', 'afternoon-birds-song-in-forest-0.wav')

In [4]: #convert to 16hz in a single channel
        def load_wav_16k_mono(filename):
            # Load encoded wav file(byte encoded string )
            file_contents = tf.io.read_file(filename)
            # Decode wav (tensors by channels)
            wav, sample_rate = tf.audio.decode_wav(file_contents, desired_channels=1)
            # Removes trailing axis
            wav = tf.squeeze(wav, axis=-1)
            sample_rate = tf.cast(sample_rate, dtype=tf.int64)
            # Goes from 44100Hz to 16000hz - amplitude of the audio signal
            wav = tfio.audio.resample(wav, rate_in=sample_rate, rate_out=16000)
            return wav

In [5]: wave = load_wav_16k_mono(CAPUCHIN_FILE)
        nwave = load_wav_16k_mono(NOT_CAPUCHIN_FILE)
```

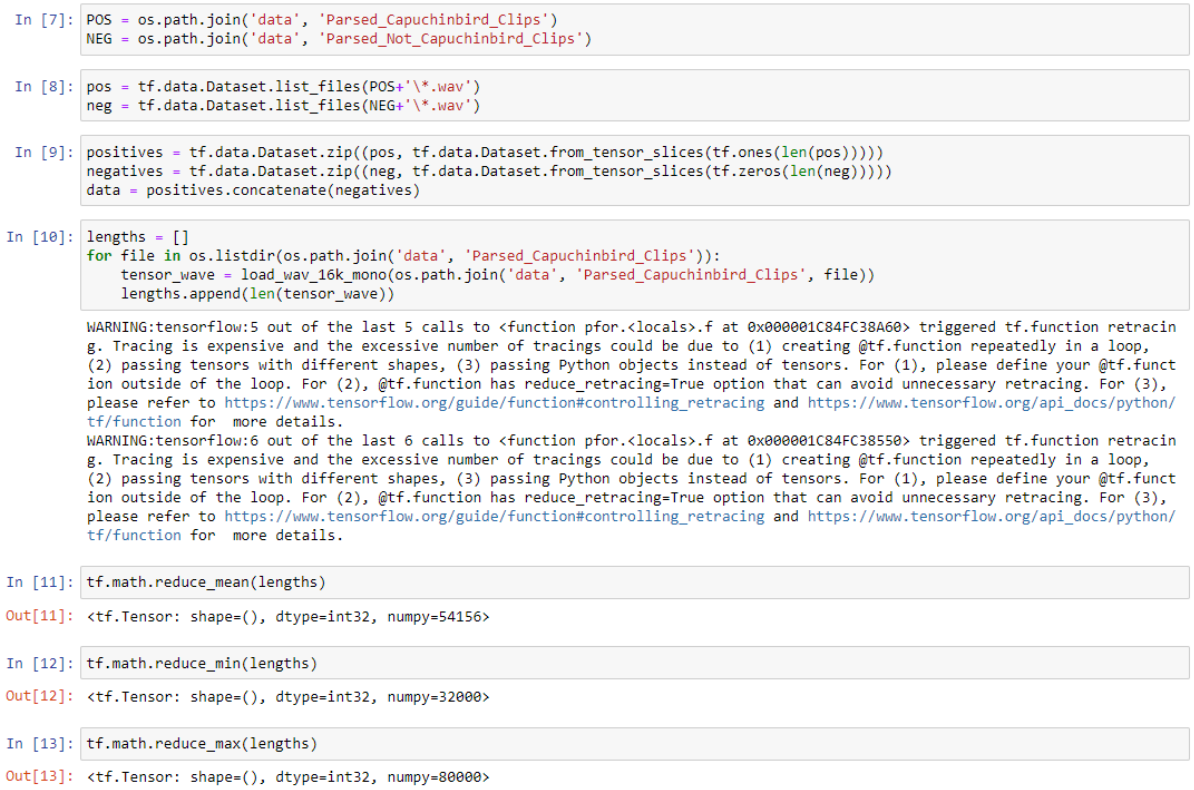


```
: plt.plot(wave)
  plt.plot(nwave)
  plt.show()
```

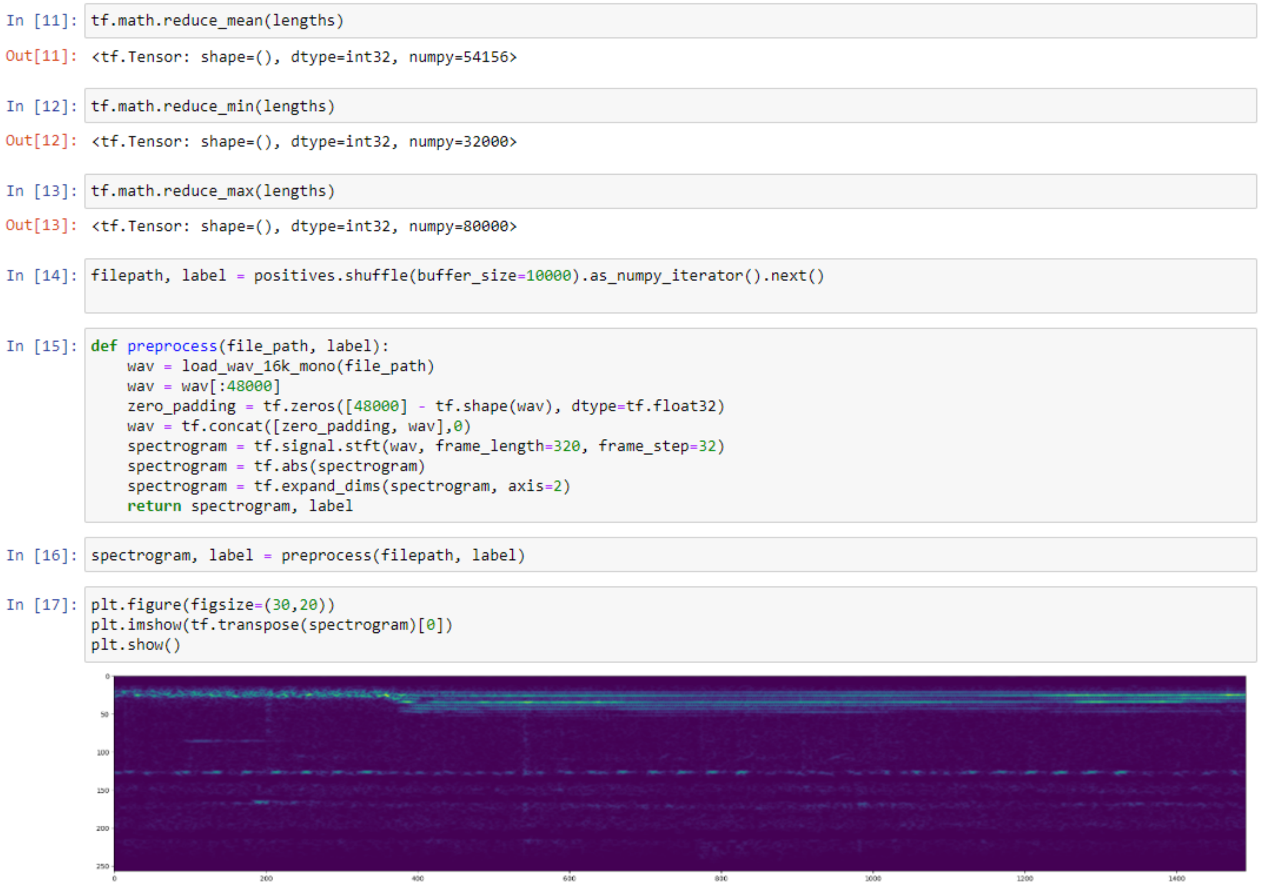




```
: POS = os.path.join('data', 'Parsed_Capuchinbird_Clips')
  NEG = os.path.join('data', 'Parsed_Not_Capuchinbird_Clips')

: pos = tf.data.Dataset.list_files(POS+'\*.wav')
  neg = tf.data.Dataset.list_files(NEG+'\*.wav')

: positives = tf.data.Dataset.zip((pos, tf.data.Dataset.from_tensor_slices(tf.ones(len(pos)))))
  negatives = tf.data.Dataset.zip((neg, tf.data.Dataset.from_tensor_slices(tf.zeros(len(neg)))))
  data = positives.concatenate(negatives)
```

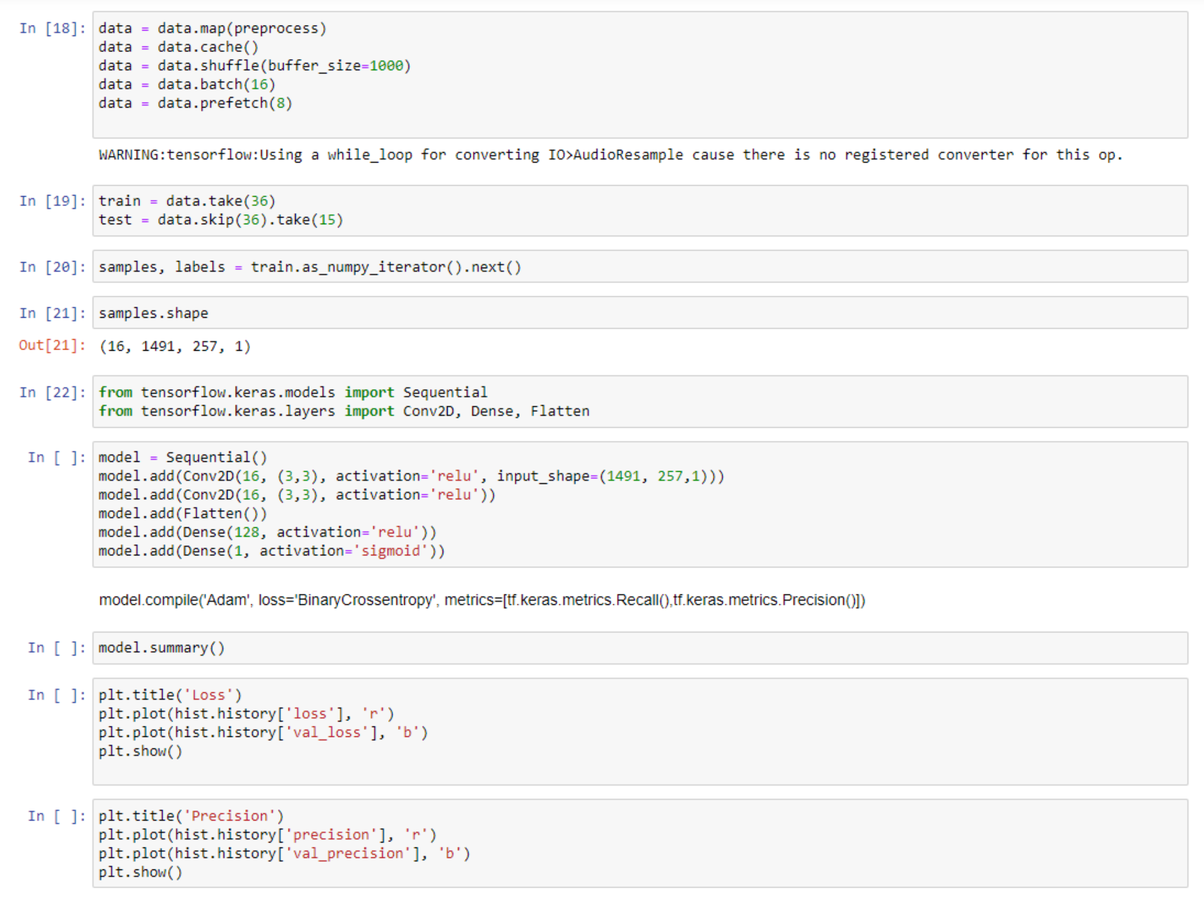


```
In [7]:  POS = os.path.join('data', 'Parsed_Capuchinbird_Clips')
         NEG = os.path.join('data', 'Parsed_Not_Capuchinbird_Clips')

In [8]:  pos = tf.data.Dataset.list_files(POS+'\*.wav')
         neg = tf.data.Dataset.list_files(NEG+'\*.wav')

In [9]:  positives = tf.data.Dataset.zip((pos, tf.data.Dataset.from_tensor_slices(tf.ones(len(pos)))))
         negatives = tf.data.Dataset.zip((neg, tf.data.Dataset.from_tensor_slices(tf.zeros(len(neg)))))
         data = positives.concatenate(negatives)

In [10]: lengths = []
         for file in os.listdir(os.path.join('data', 'Parsed_Capuchinbird_Clips')):
             tensor_wave = load_wav_16k_mono(os.path.join('data', 'Parsed_Capuchinbird_Clips', file))
             lengths.append(len(tensor_wave))

         WARNING:tensorflow:5 out of the last 5 calls to <function pfor.<locals>.f at 0x000001C84FC38A60> triggered tf.function retracin
         g. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop,
         (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.funct
         ion outside of the loop. For (2), @tf.function has reduce_retracing=True option that can avoid unnecessary retracing. For (3),
         please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/
         tf/function for  more details.
         WARNING:tensorflow:6 out of the last 6 calls to <function pfor.<locals>.f at 0x000001C84FC38550> triggered tf.function retracin
         g. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop,
         (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.funct
         ion outside of the loop. For (2), @tf.function has reduce_retracing=True option that can avoid unnecessary retracing. For (3),
         please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/
         tf/function for  more details.

In [11]: tf.math.reduce_mean(lengths)

Out[11]: <tf.Tensor: shape=(), dtype=int32, numpy=54156>

In [12]: tf.math.reduce_min(lengths)

Out[12]: <tf.Tensor: shape=(), dtype=int32, numpy=32000>

In [13]: tf.math.reduce_max(lengths)

Out[13]: <tf.Tensor: shape=(), dtype=int32, numpy=80000>
```

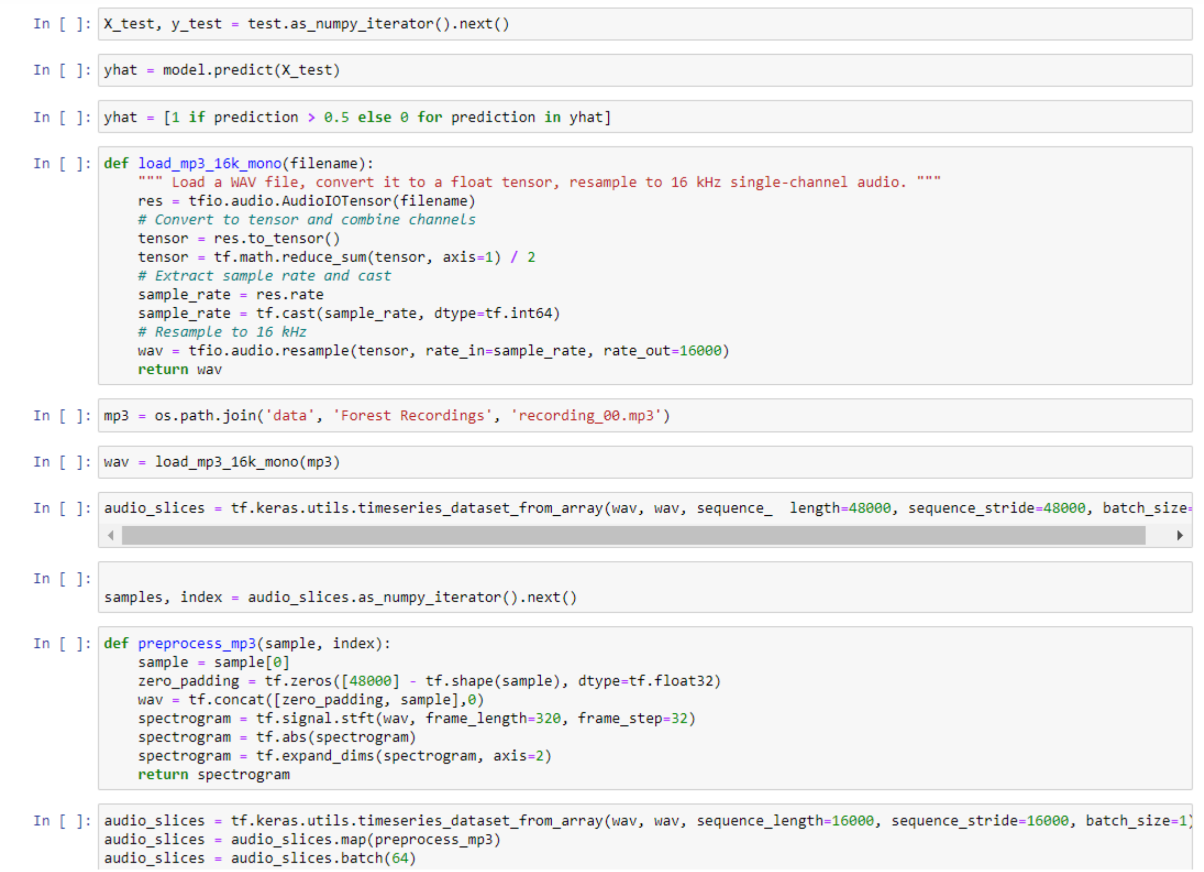


```
In [11]: tf.math.reduce_mean(lengths)

Out[11]: <tf.Tensor: shape=(), dtype=int32, numpy=54156>

In [12]: tf.math.reduce_min(lengths)

Out[12]: <tf.Tensor: shape=(), dtype=int32, numpy=32000>

In [13]: tf.math.reduce_max(lengths)

Out[13]: <tf.Tensor: shape=(), dtype=int32, numpy=80000>
```

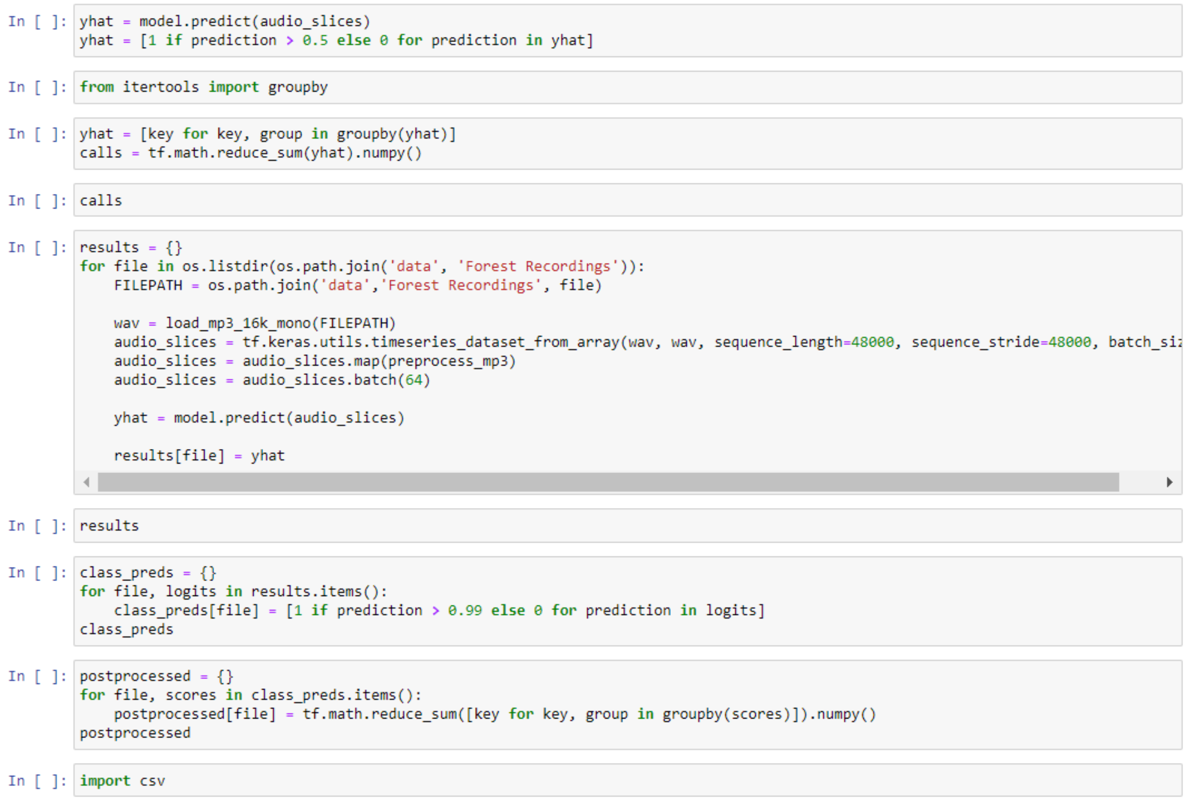
```
In [14]: filepath, label = positives.shuffle(buffer_size=10000).as_numpy_iterator().next()

In [15]: def preprocess(file_path, label):
             wav = load_wav_16k_mono(file_path)
             wav = wav[:48000]
             zero_padding = tf.zeros([48000] - tf.shape(wav), dtype=tf.float32)
             wav = tf.concat([zero_padding, wav],0)
             spectrogram = tf.signal.stft(wav, frame_length=320, frame_step=32)
             spectrogram = tf.abs(spectrogram)
             spectrogram = tf.expand_dims(spectrogram, axis=2)
             return spectrogram, label

In [16]: spectrogram, label = preprocess(filepath, label)

In [17]: plt.figure(figsize=(30,20))
         plt.imshow(tf.transpose(spectrogram)[0])
         plt.show()
```



```
In [18]: data = data.map(preprocess)
         data = data.cache()
         data = data.shuffle(buffer_size=1000)
         data = data.batch(16)
         data = data.prefetch(8)

         WARNING:tensorflow:Using a while_loop for converting IO>AudioResample cause there is no registered converter for this op.

In [19]: train = data.take(36)
         test = data.skip(36).take(15)

In [20]: samples, labels = train.as_numpy_iterator().next()

In [21]: samples.shape

Out[21]: (16, 1491, 257, 1)

In [22]: from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Conv2D, Dense, Flatten

In [ ]: model = Sequential()
        model.add(Conv2D(16, (3,3), activation='relu', input_shape=(1491, 257,1)))
        model.add(Conv2D(16, (3,3), activation='relu'))
        model.add(Flatten())
        model.add(Dense(128, activation='relu'))
        model.add(Dense(1, activation='sigmoid'))

        model.compile('Adam', loss='BinaryCrossentropy', metrics=[tf.keras.metrics.Recall(),tf.keras.metrics.Precision()])

In [ ]: model.summary()

In [ ]: plt.title('Loss')
        plt.plot(hist.history['loss'], 'r')
        plt.plot(hist.history['val_loss'], 'b')
        plt.show()

In [ ]: plt.title('Precision')
        plt.plot(hist.history['precision'], 'r')
        plt.plot(hist.history['val_precision'], 'b')
        plt.show()
```

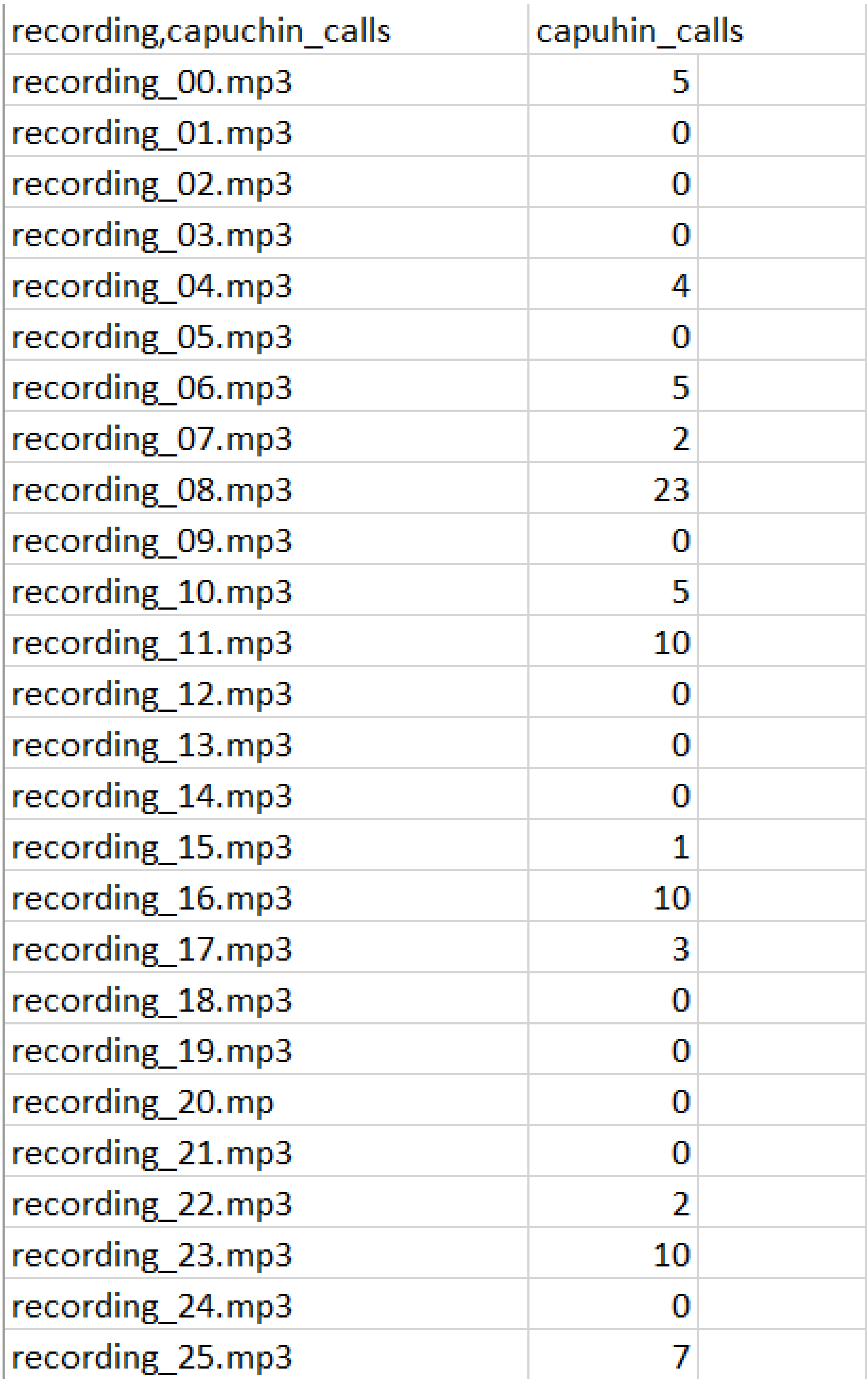


```
In [ ]: X_test, y_test = test.as_numpy_iterator().next()

In [ ]: yhat = model.predict(X_test)

In [ ]: yhat = [1 if prediction > 0.5 else 0 for prediction in yhat]

In [ ]: def load_mp3_16k_mono(filename):
            """ Load a WAV file, convert it to a float tensor, resample to 16 kHz single-channel audio. """
            res = tfio.audio.AudioIOTensor(filename)
            # Convert to tensor and combine channels
            tensor = res.to_tensor()
            tensor = tf.math.reduce_sum(tensor, axis=1) / 2
            # Extract sample rate and cast
            sample_rate = res.rate
            sample_rate = tf.cast(sample_rate, dtype=tf.int64)
            # Resample to 16 kHz
            wav = tfio.audio.resample(tensor, rate_in=sample_rate, rate_out=16000)
            return wav

In [ ]: mp3 = os.path.join('data', 'Forest Recordings', 'recording_00.mp3')

In [ ]: wav = load_mp3_16k_mono(mp3)

In [ ]: audio_slices = tf.keras.utils.timeseries_dataset_from_array(wav, wav, sequence_  length=48000, sequence_stride=48000, batch_size=

In [ ]:
        samples, index = audio_slices.as_numpy_iterator().next()

In [ ]: def preprocess_mp3(sample, index):
            sample = sample[0]
            zero_padding = tf.zeros([48000] - tf.shape(sample), dtype=tf.float32)
            wav = tf.concat([zero_padding, sample],0)
            spectrogram = tf.signal.stft(wav, frame_length=320, frame_step=32)
            spectrogram = tf.abs(spectrogram)
            spectrogram = tf.expand_dims(spectrogram, axis=2)
            return spectrogram

In [ ]: audio_slices = tf.keras.utils.timeseries_dataset_from_array(wav, wav, sequence_length=16000, sequence_stride=16000, batch_size=1)
        audio_slices = audio_slices.map(preprocess_mp3)
        audio_slices = audio_slices.batch(64)
```

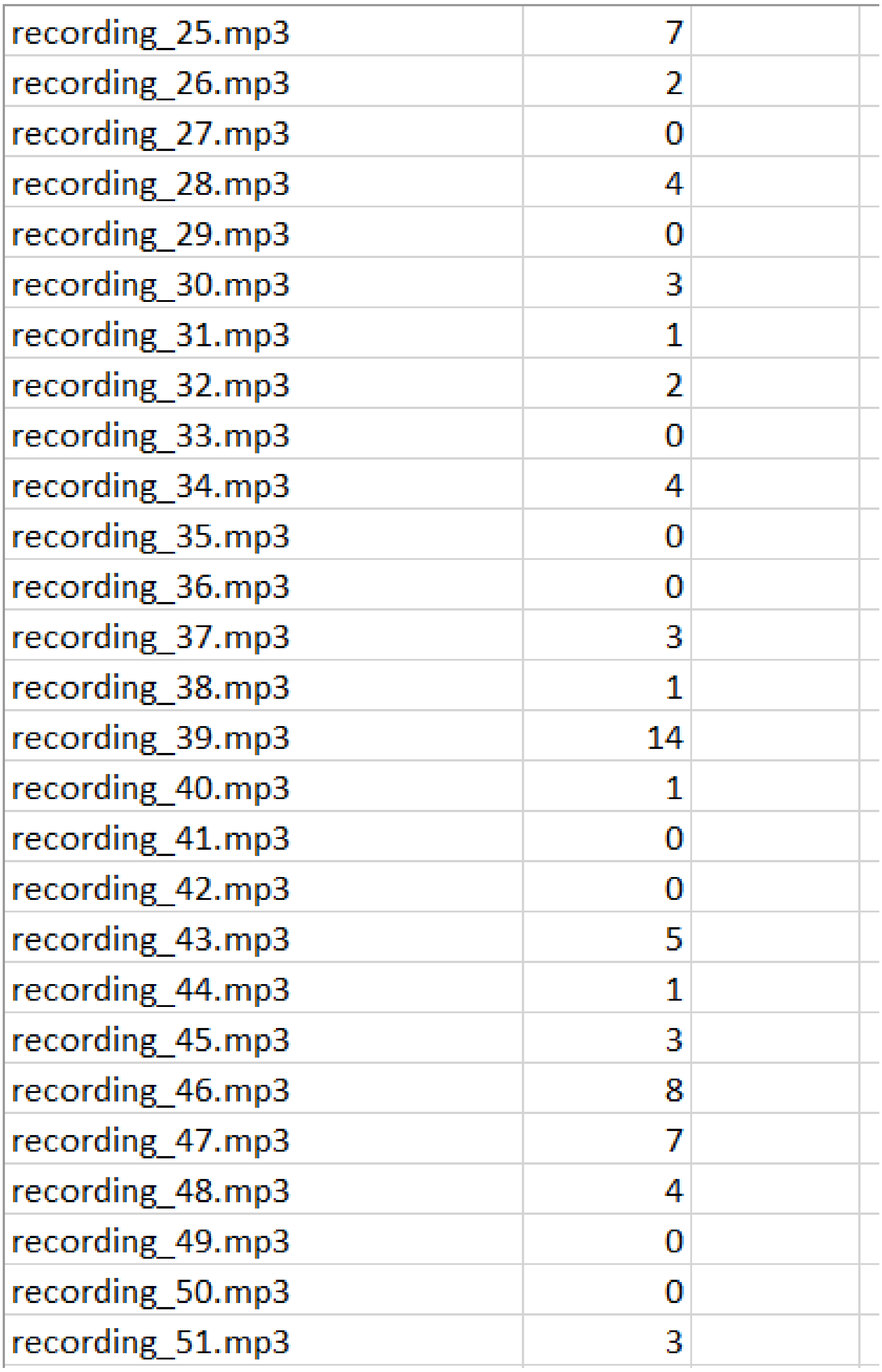


```
In [ ]: yhat = model.predict(audio_slices)
        yhat = [1 if prediction > 0.5 else 0 for prediction in yhat]

In [ ]: from itertools import groupby

In [ ]: yhat = [key for key, group in groupby(yhat)]
        calls = tf.math.reduce_sum(yhat).numpy()

In [ ]: calls

In [ ]: results = {}
        for file in os.listdir(os.path.join('data', 'Forest Recordings')):
            FILEPATH = os.path.join('data','Forest Recordings', file)

            wav = load_mp3_16k_mono(FILEPATH)
            audio_slices = tf.keras.utils.timeseries_dataset_from_array(wav, wav, sequence_length=48000, sequence_stride=48000, batch_siz
            audio_slices = audio_slices.map(preprocess_mp3)
            audio_slices = audio_slices.batch(64)

            yhat = model.predict(audio_slices)

            results[file] = yhat

In [ ]: results

In [ ]: class_preds = {}
        for file, logits in results.items():
            class_preds[file] = [1 if prediction > 0.99 else 0 for prediction in logits]
        class_preds

In [ ]: postprocessed = {}
        for file, scores in class_preds.items():
            postprocessed[file] = tf.math.reduce_sum([key for key, group in groupby(scores)]).numpy()
        postprocessed

In [ ]: import csv
```

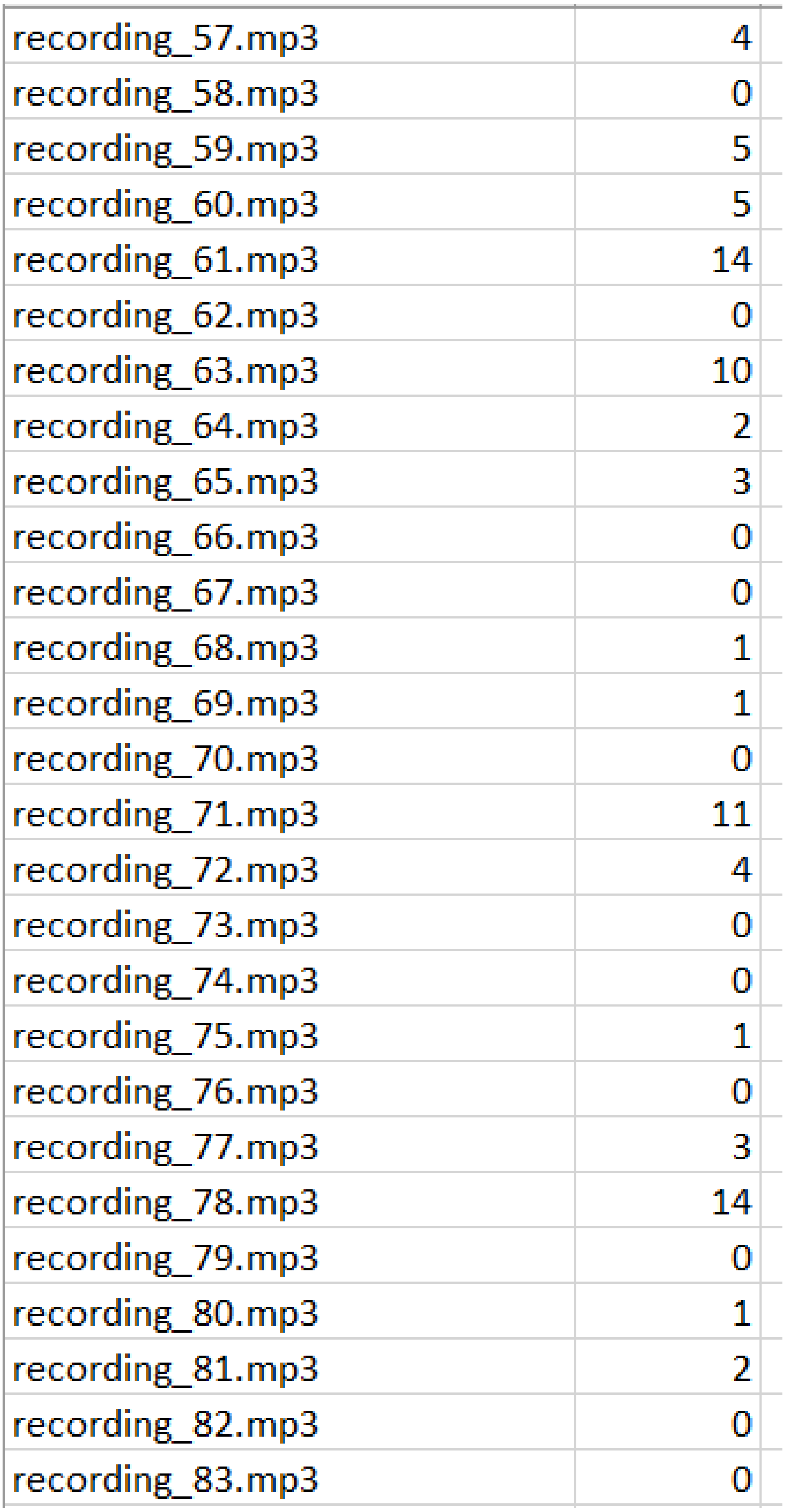
OUTPUT CSV FILE:



recording,capuchin_calls	capuhin_calls
recording_00.mp3	5
recording_01.mp3	0
recording_02.mp3	0
recording_03.mp3	0
recording_04.mp3	4
recording_05.mp3	0
recording_06.mp3	5
recording_07.mp3	2
recording_08.mp3	23
recording_09.mp3	0
recording_10.mp3	5
recording_11.mp3	10
recording_12.mp3	0
recording_13.mp3	0
recording_14.mp3	0
recording_15.mp3	1
recording_16.mp3	10
recording_17.mp3	3
recording_18.mp3	0
recording_19.mp3	0
recording_20.mp	0
recording_21.mp3	0
recording_22.mp3	2
recording_23.mp3	10
recording_24.mp3	0
recording_25.mp3	7



recording_25.mp3	7
recording_26.mp3	2
recording_27.mp3	0
recording_28.mp3	4
recording_29.mp3	0
recording_30.mp3	3
recording_31.mp3	1
recording_32.mp3	2
recording_33.mp3	0
recording_34.mp3	4
recording_35.mp3	0
recording_36.mp3	0
recording_37.mp3	3
recording_38.mp3	1
recording_39.mp3	14
recording_40.mp3	1
recording_41.mp3	0
recording_42.mp3	0
recording_43.mp3	5
recording_44.mp3	1
recording_45.mp3	3
recording_46.mp3	8
recording_47.mp3	7
recording_48.mp3	4
recording_49.mp3	0
recording_50.mp3	0
recording_51.mp3	3



recording_57.mp3	4
recording_58.mp3	0
recording_59.mp3	5
recording_60.mp3	5
recording_61.mp3	14
recording_62.mp3	0
recording_63.mp3	10
recording_64.mp3	2
recording_65.mp3	3
recording_66.mp3	0
recording_67.mp3	0
recording_68.mp3	1
recording_69.mp3	1
recording_70.mp3	0
recording_71.mp3	11
recording_72.mp3	4
recording_73.mp3	0
recording_74.mp3	0
recording_75.mp3	1
recording_76.mp3	0
recording_77.mp3	3
recording_78.mp3	14
recording_79.mp3	0
recording_80.mp3	1
recording_81.mp3	2
recording_82.mp3	0
recording_83.mp3	0