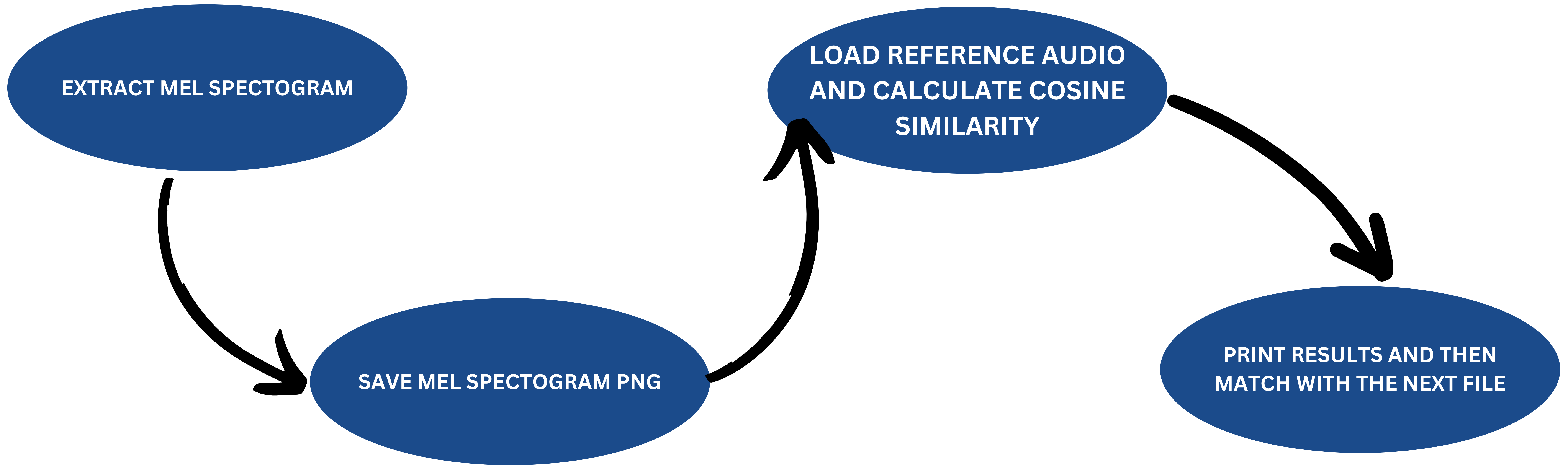
CLASSIFICATION METHODS

1. Pattern matching
2. SVM
3. CNN

FLOW OF PATTERN MATCHING


EXTRACT MEL SPECTOGRAM
SAVE MEL SPECTOGRAM PNG
LOAD REFERENCE AUDIO AND CALCULATE COSINE SIMILARITY
PRINT RESULTS AND THEN MATCH WITH THE NEXT FILE

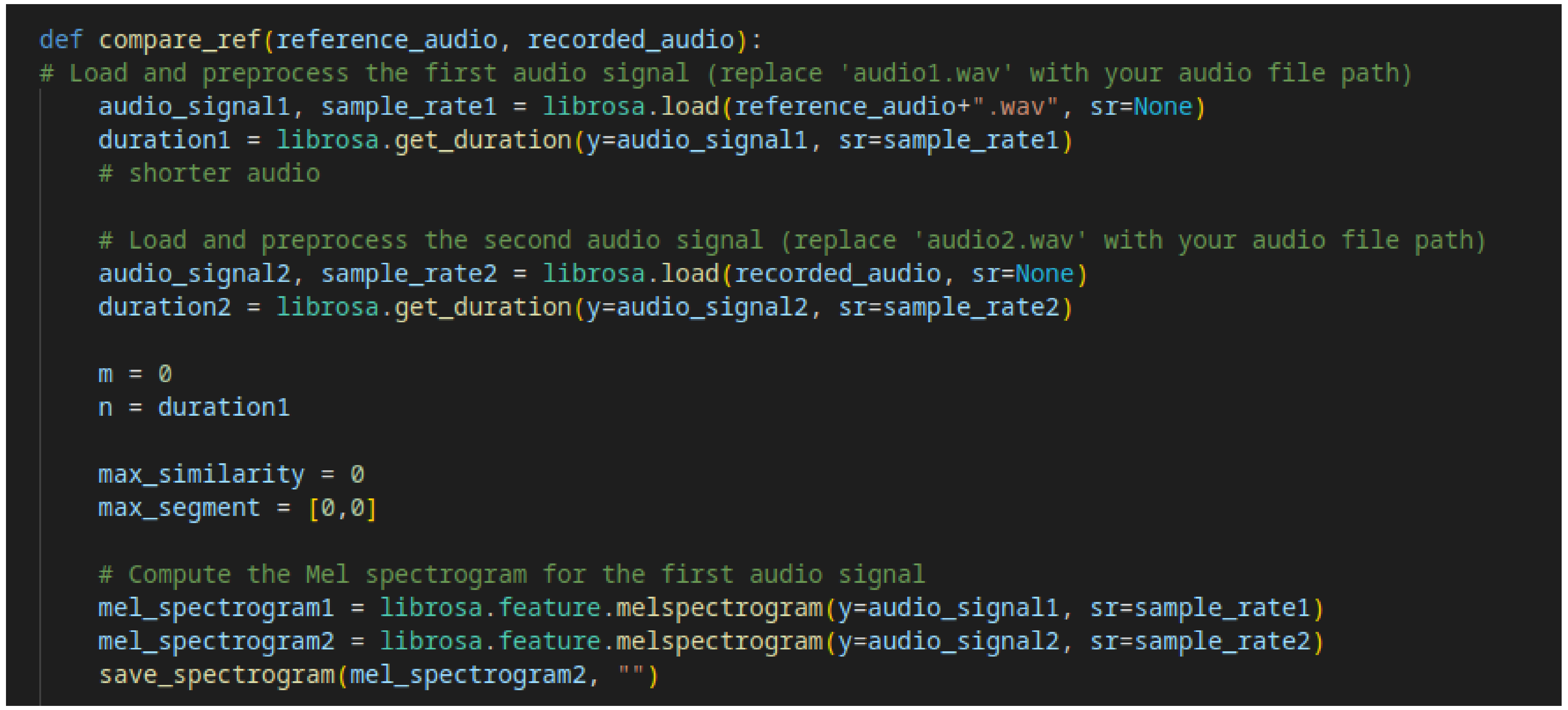
PATTERN MATCHING

A classifier that uses pattern matching that makes decision based on identifying patterns or similarities between the input data and predefined dataset

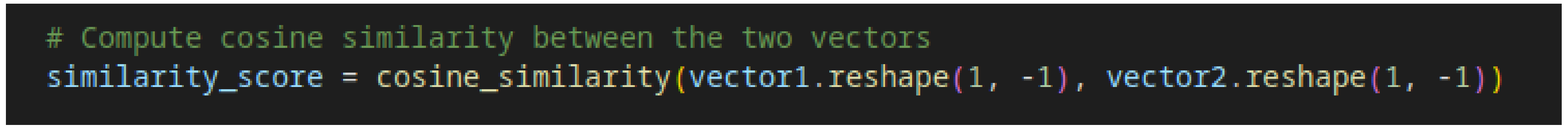
Calculates the cosine similarity for the mel spectogram and prints the similarity with each file.

Not an effiecient method because, two audios having the same characteristic sound do not have a greater similarity percentage in most of the cases.

Also time consuming method because, every file in the dataset must be compared with the given audio. Instead its better to train a machine and use it.

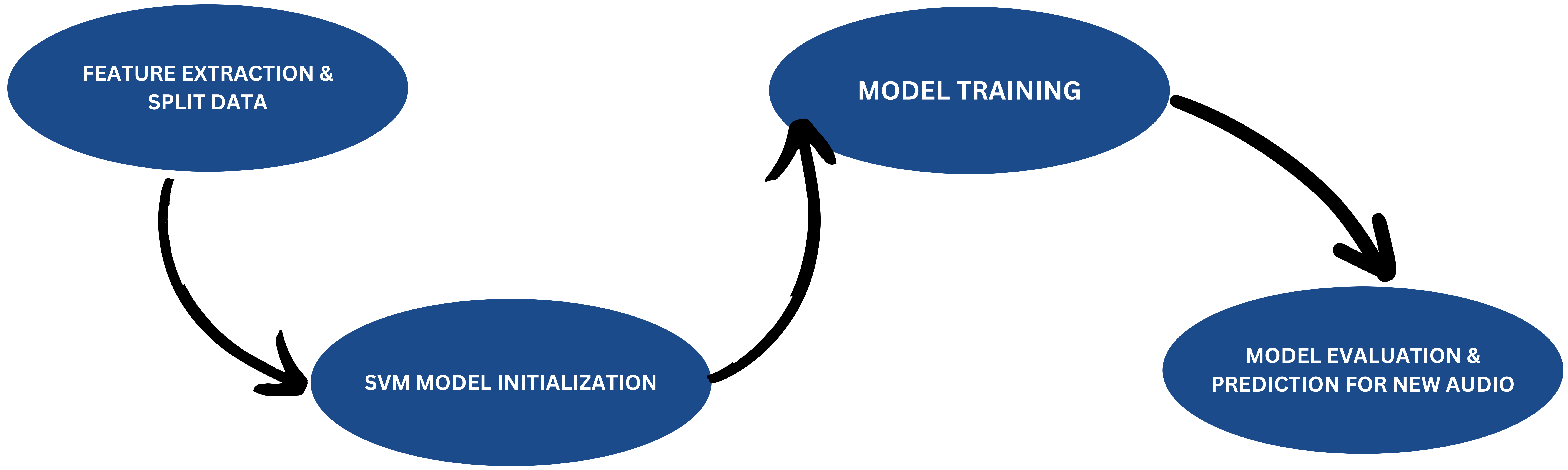


```python
def compare_ref(reference_audio, recorded_audio):
# Load and preprocess the first audio signal (replace 'audio1.wav' with your audio file path)
    audio_signal1, sample_rate1 = librosa.load(reference_audio+".wav", sr=None)
    duration1 = librosa.get_duration(y=audio_signal1, sr=sample_rate1)
    # shorter audio

    # Load and preprocess the second audio signal (replace 'audio2.wav' with your audio file path)
    audio_signal2, sample_rate2 = librosa.load(recorded_audio, sr=None)
    duration2 = librosa.get_duration(y=audio_signal2, sr=sample_rate2)

    m = 0
    n = duration1

    max_similarity = 0
    max_segment = [0,0]

    # Compute the Mel spectrogram for the first audio signal
    mel_spectrogram1 = librosa.feature.melspectrogram(y=audio_signal1, sr=sample_rate1)
    mel_spectrogram2 = librosa.feature.melspectrogram(y=audio_signal2, sr=sample_rate2)
    save_spectrogram(mel_spectrogram2, "")
```

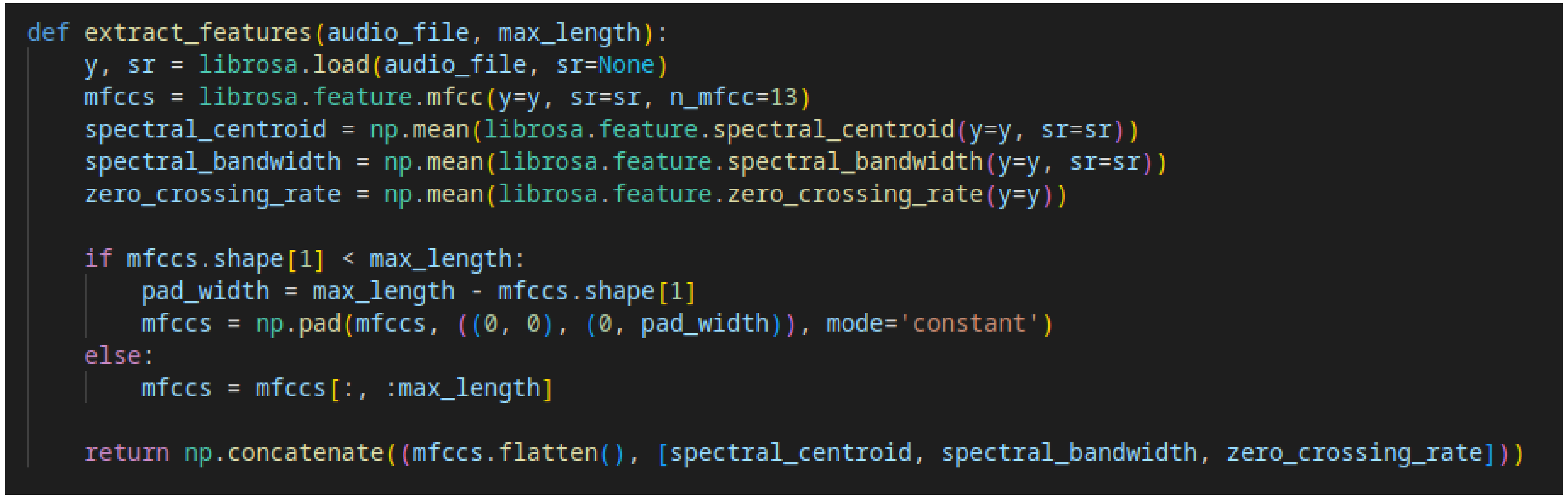


```python
# Compute cosine similarity between the two vectors
similarity_score = cosine_similarity(vector1.reshape(1, -1), vector2.reshape(1, -1))
```

Support Vector Machines


FEATURE EXTRACTION & SPLIT DATA
SVM MODEL INITIALIZATION
MODEL TRAINING
MODEL EVALUATION & PREDICTION FOR NEW AUDIO

An SVM classifier is initialized with a linear kernel using the SVC (Support Vector Classification) class from sklearn.svm.
The SVM classifier is trained on the training data using the fit method.
The fit method is a common method used to train a model on a given dataset. The term "fitting" refers to adjusting the parameters of the model to make it perform well on the training data.



```python
def extract_features(audio_file, max_length):
    y, sr = librosa.load(audio_file, sr=None)
    mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
    spectral_centroid = np.mean(librosa.feature.spectral_centroid(y=y, sr=sr))
    spectral_bandwidth = np.mean(librosa.feature.spectral_bandwidth(y=y, sr=sr))
    zero_crossing_rate = np.mean(librosa.feature.zero_crossing_rate(y=y))

    if mfccs.shape[1] < max_length:
        pad_width = max_length - mfccs.shape[1]
        mfccs = np.pad(mfccs, ((0, 0), (0, pad_width)), mode='constant')
    else:
        mfccs = mfccs[:, :max_length]

    return np.concatenate((mfccs.flatten(), [spectral_centroid, spectral_bandwidth, zero_crossing_rate]))
```

Zero-crossing rate

Wheezing sounds often have a more continuous and sinusoidal nature,the waveform may have a smoother transition between positive and negative values. This can result in a relatively low zero-crossing rate.

Crackle sounds,the waveform may have more rapid changes in amplitude, leading to a higher zero-crossing rate compared to wheeze sounds