EEG

1. Overview

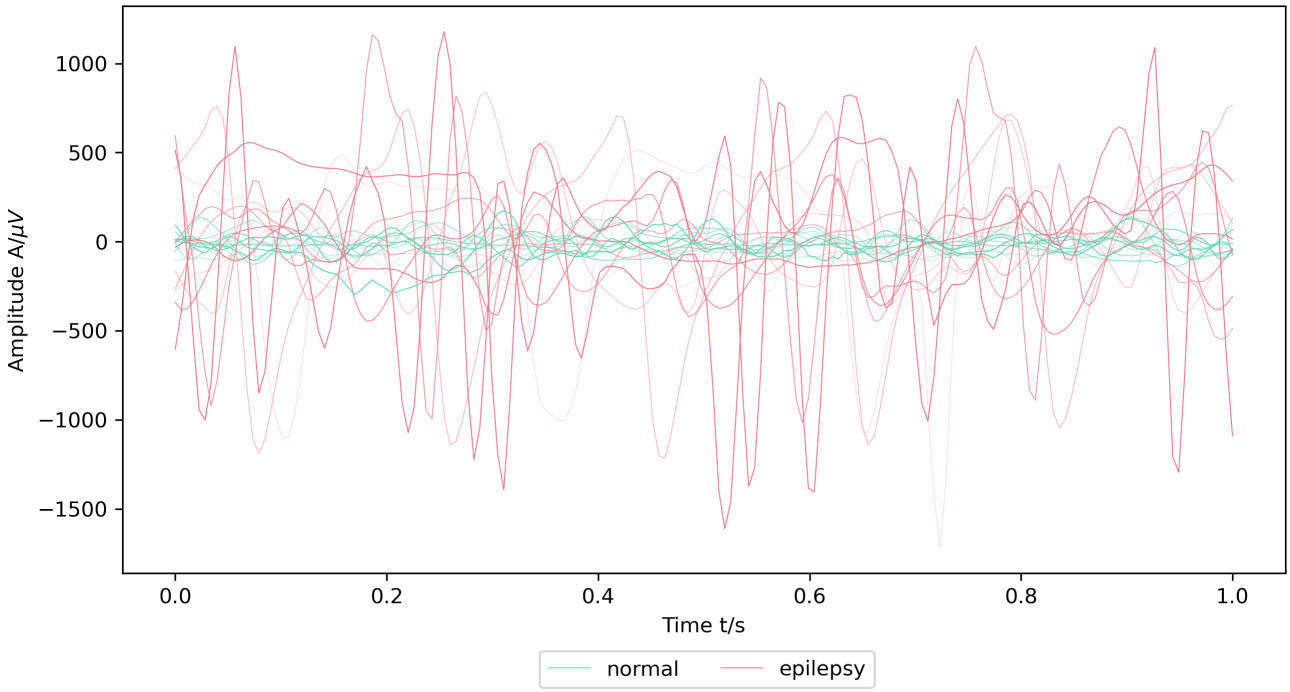


Fig. 1 Overview of EEG samples.

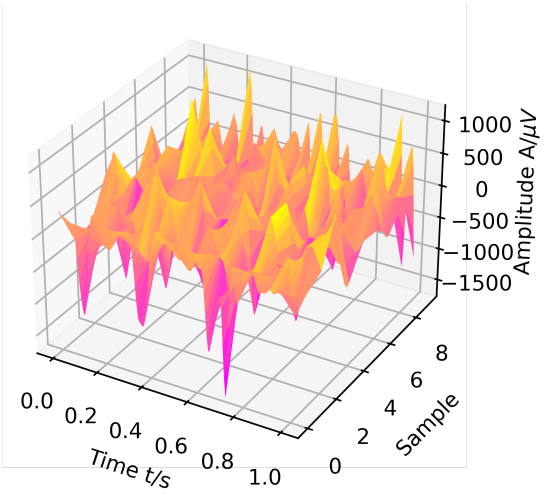


Fig. 2.1 3D plot of Epilepsy EEG samples.

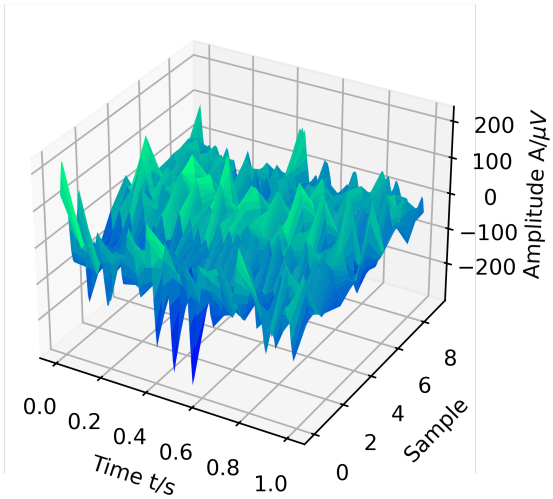


Fig. 2.2 3D plot of Normal EEG samples.

2. Feature extraction

2.1 ApEn

```
def ApEn(U, m=30):
    r = np.std(U)
    def _maxdist(x_i, x_j):
        return max([abs(ua - va) for ua, va in zip(x_i, x_j)])

    def _phi(m):
        x = [[U[j] for j in range(i, i + m - 1 + 1)] for i in range(N - m + 1)]
        C = [len([1 for x_j in x if _maxdist(x_i, x_j) <= r]) / (N - m + 1.0) for x_i in x]
        return (N - m + 1.0)**(-1) * sum(np.log(C))

    N = len(U)

    return abs(_phi(m+1) - _phi(m))
```

2.2 peek

```
data_epi_N = data_epi.sub(data_epi.mean(axis=1), axis=0)
data_epi_pos = data_epi_N[data_epi_N > 0]
data_epi_neg = data_epi_N[data_epi_N < 0]
data_epi_pospeek = data_epi_N.sub(1.5*data_epi_pos.mean(axis=1), axis=0)
data_epi_negpeek = data_epi_N.sub(1.5*data_epi_neg.mean(axis=1), axis=0)

def epi_SUM_peek(i):
    pospeek = lib.zero_crossings(np.array(data_epi_pospeek.iloc[i]))
    negpeek = lib.zero_crossings(np.array(data_epi_negpeek.iloc[i]))
    return max(np.sum(pospeek==1), np.sum(negpeek==1))
```

2.3 fft

```
from scipy.fftpack import fft, fftfreq
def mean_freq(y):
    fs = 178
    # signal
    t = np.linspace(0, 1, 178)
    N = len(t)
    # fft
    yf = fft(y)
    xf = fftfreq(N, 1/fs)[:N//2]
    yf_nor = 2.0/N * np.abs(yf[0:N//2]) # normalization
    xf_half = xf[:len(yf_nor)]
    return np.mean(xf_half[yf_nor.argsort()[-2:]])
```

2.4 RMS

```
if y is not None:
        y = to_mono(y)
        if center:
            y = np.pad(y, int(frame_length // 2), mode=pad_mode)

        x = util.frame(y, frame_length=frame_length, hop_length=hop_length)

        # Calculate power
        power = np.mean(np.abs(x) ** 2, axis=0, keepdims=True)
    elif S is not None:
        # Check the frame length
        if S.shape[0] != frame_length // 2 + 1:
```

```
            raise ParameterError(
                "Since S.shape[0] is {}, "
                "frame_length is expected to be {} or {}; "
                "found {}".format(
                    S.shape[0], S.shape[0] * 2 - 2, S.shape[0] * 2 - 1, frame_length
                )
            )

        # power spectrogram
        x = np.abs(S) ** 2

        # Adjust the DC and sr/2 component
        x[0] *= 0.5
        if frame_length % 2 == 0:
            x[-1] *= 0.5

        # Calculate power
        power = 2 * np.sum(x, axis=0, keepdims=True) / frame_length ** 2
    else:
        raise ParameterError("Either `y` or `S` must be input.")

    return np.sqrt(power)
```

3. Feature distribution

Fig. 3.1 Distribution of ApEn values

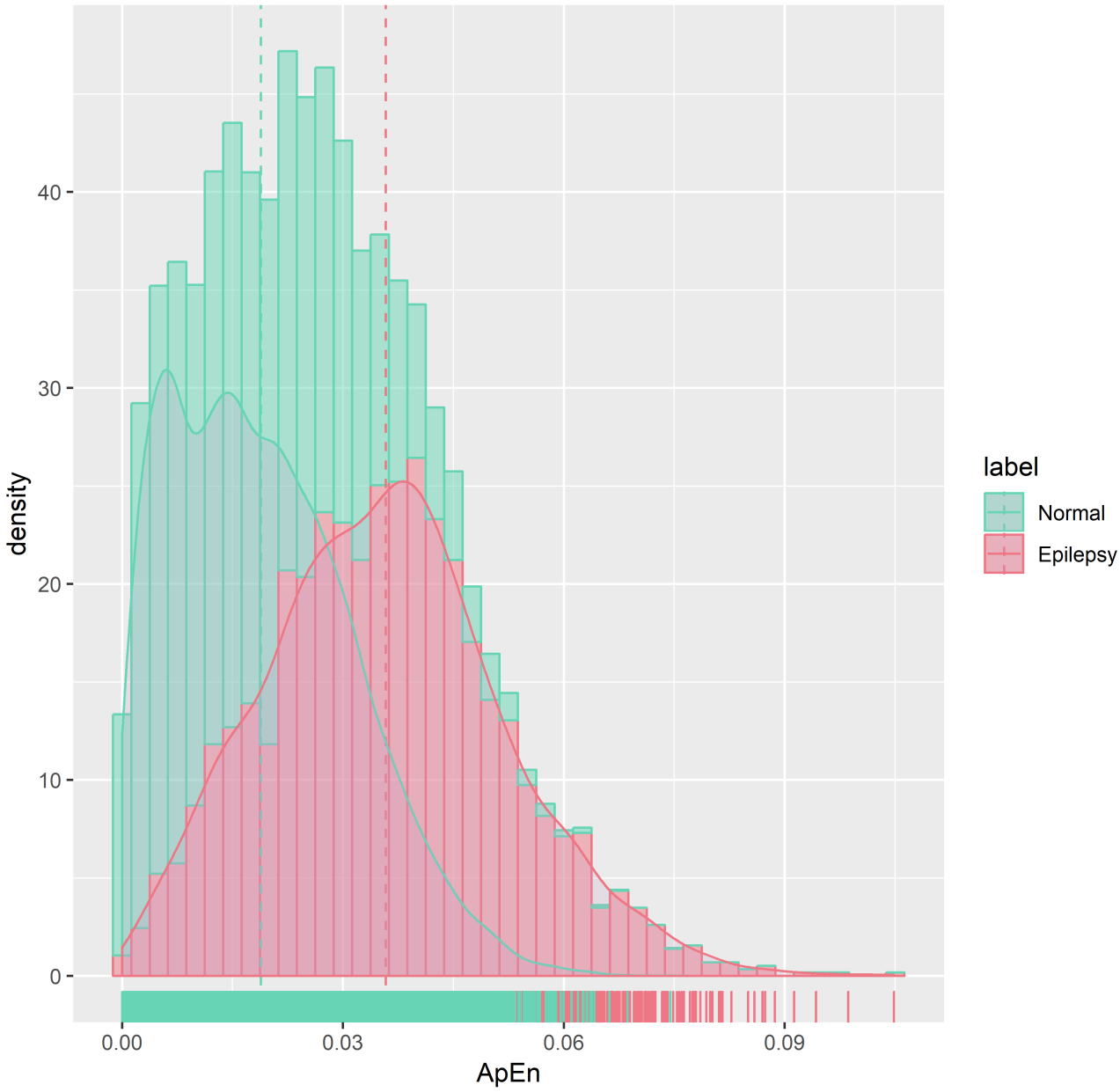


Fig. 3.2 Distribution of mean frequency values

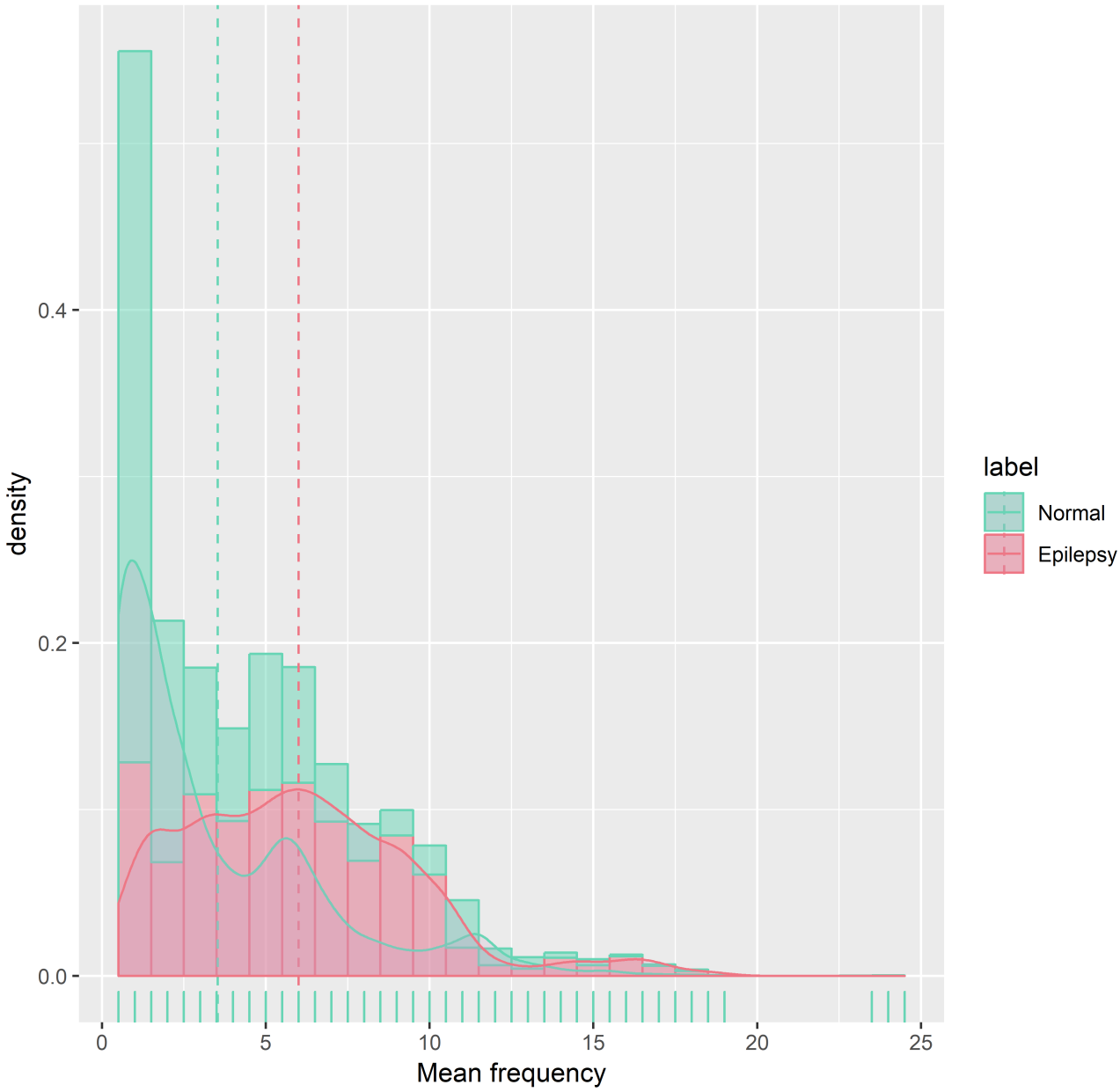


Fig. 3.3 Distribution of the number of peeks

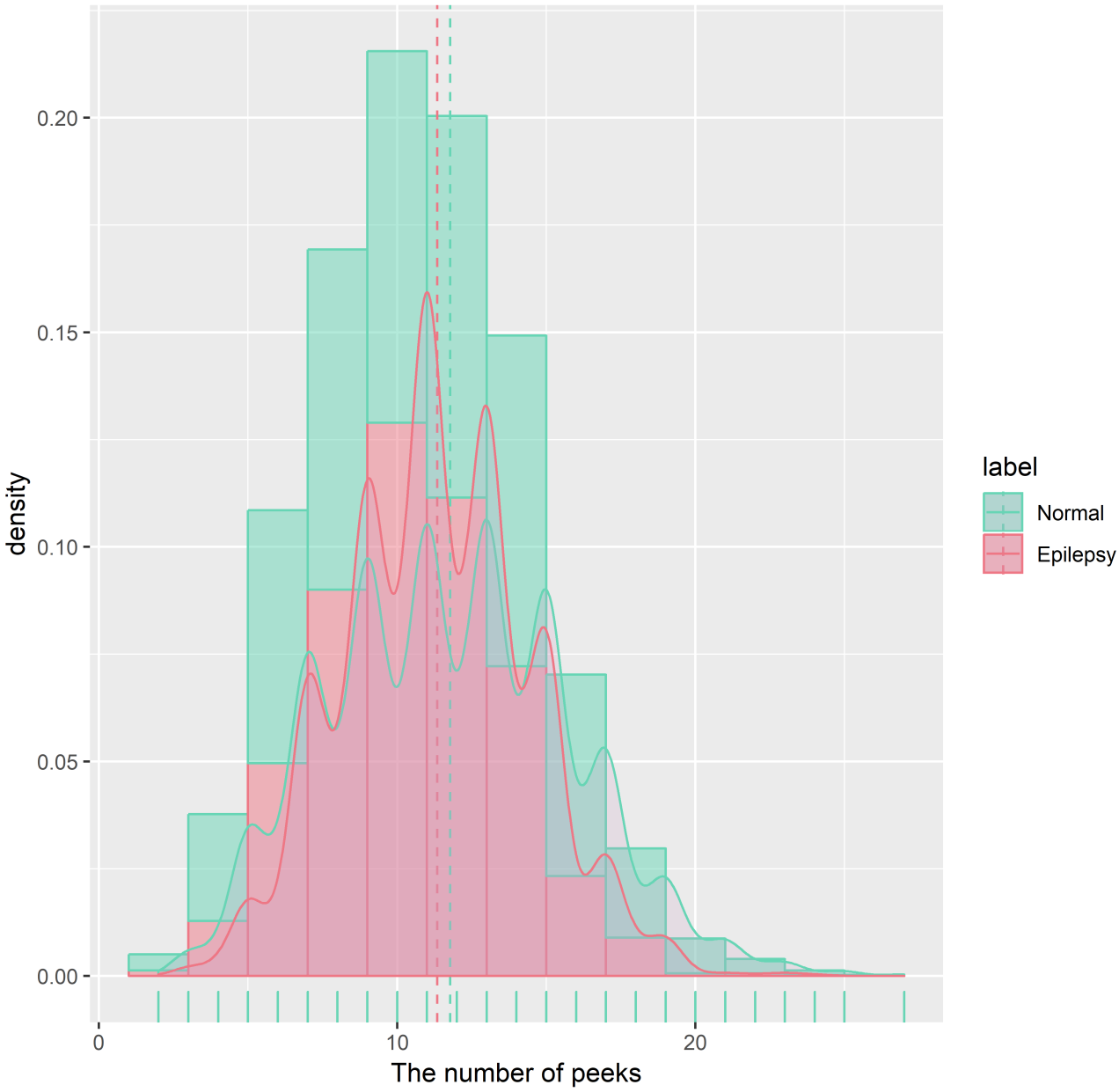
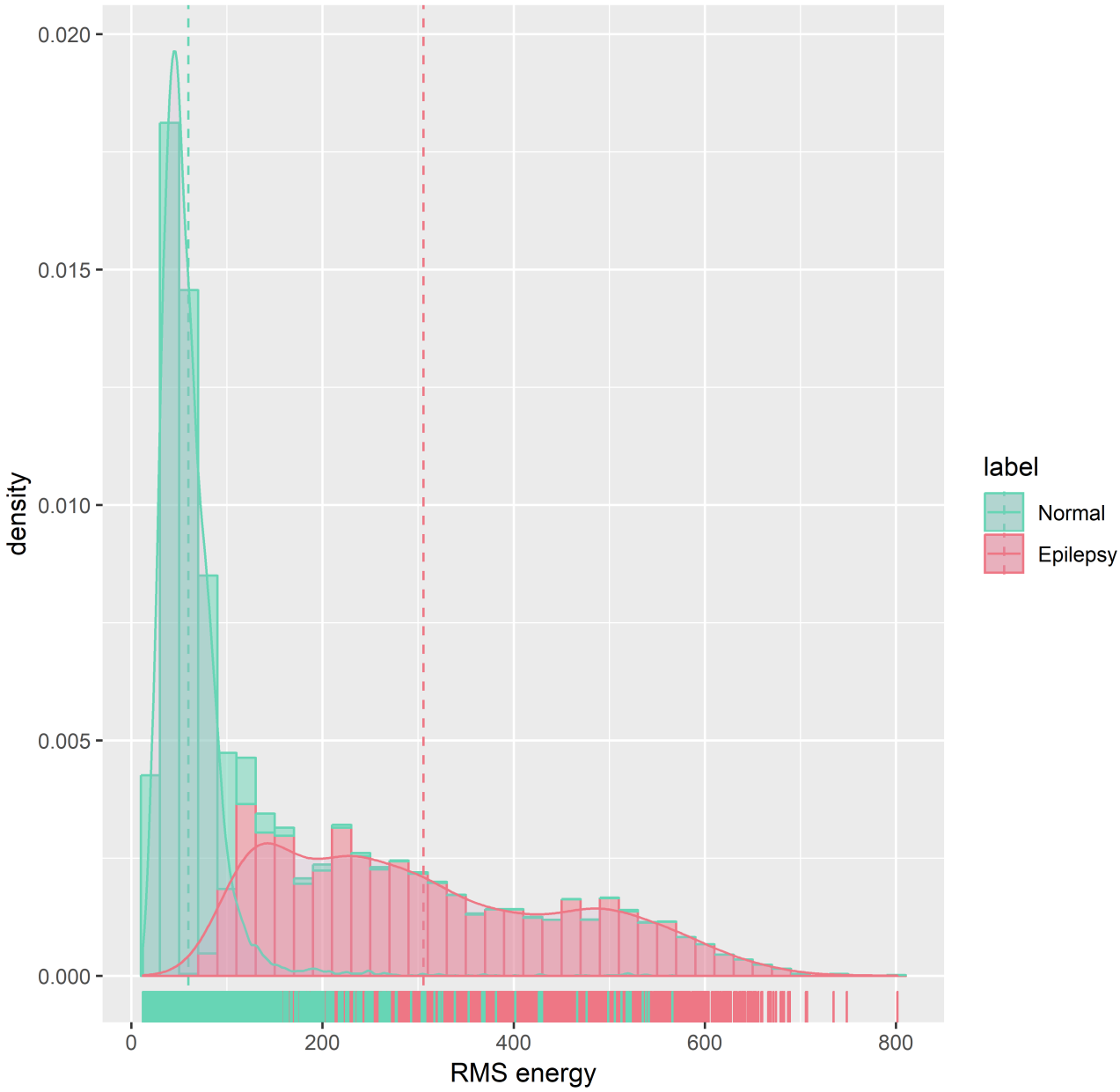


Fig. 3.4 Distribution of root-mean-squre energy values



4. Classification

3.1 SVM

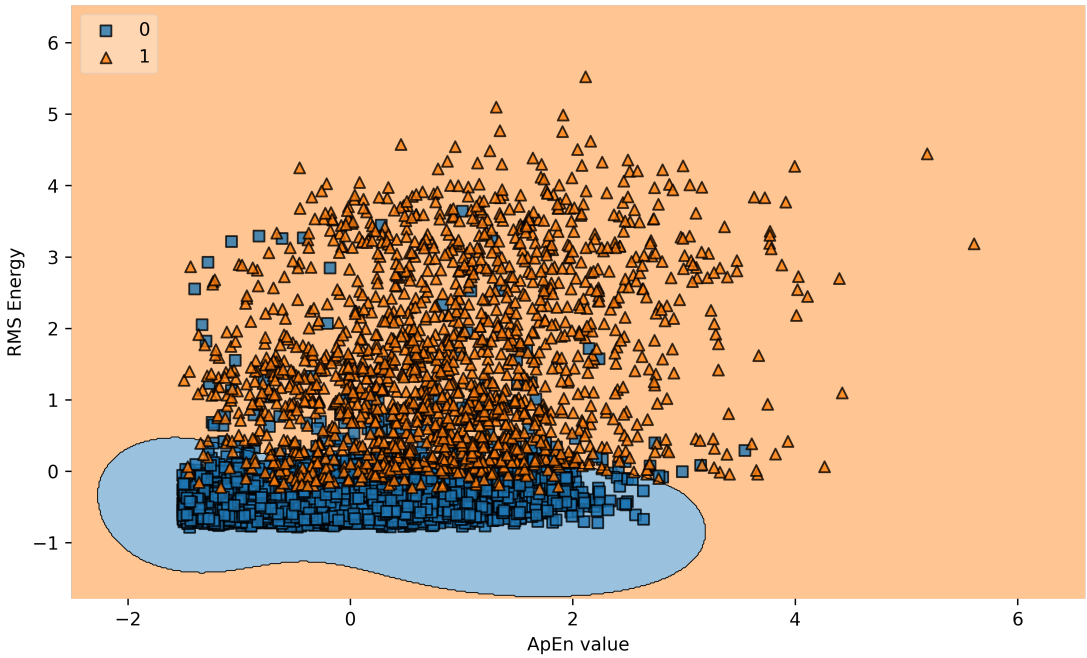


Fig. 4 Classification with SVM method.

Accuracy: 0.9607452774658556
SVC(C=1, gamma=1)

	precision	recall	f1-score	support
0	0.98	0.97	0.98	2785
1	0.90	0.92	0.91	665
accuracy			0.96	3450
macro avg	0.94	0.95	0.94	3450
weighted avg	0.96	0.96	0.96	3450

3.2 Decision tree

Include peek

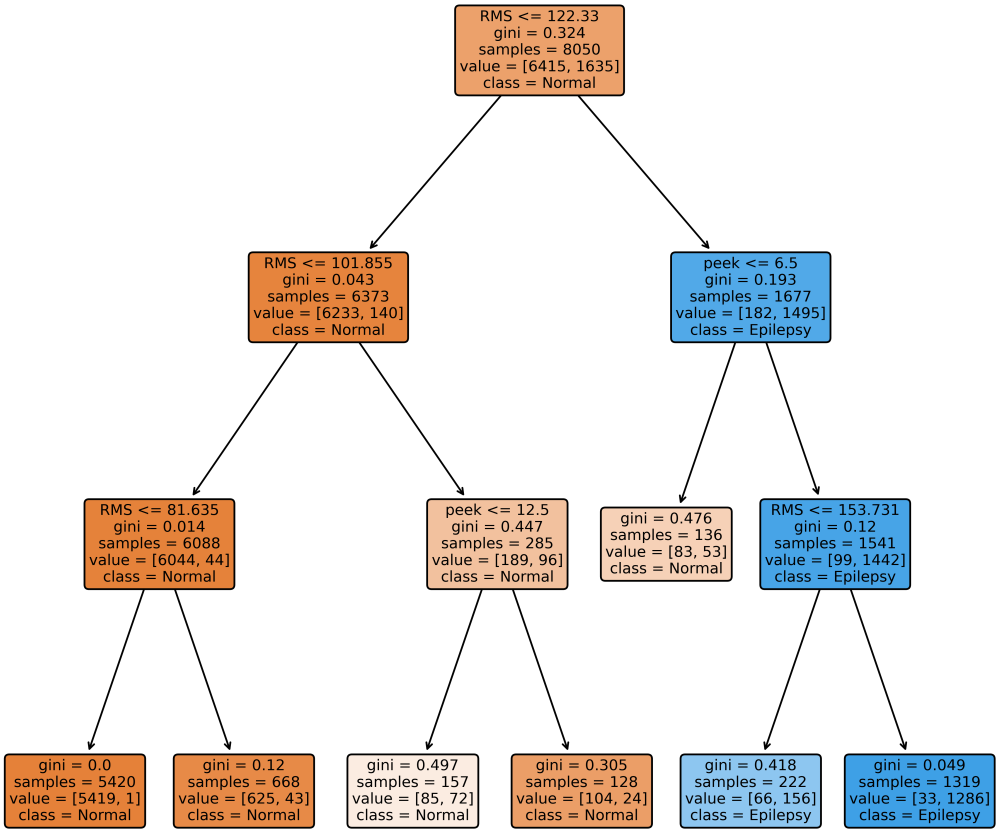


Fig. 5.1 Classification with decision tree.

Accuracy: 0.9637681159420289

	precision	recall	f1-score	support
0	0.97	0.98	0.98	2785
1	0.93	0.88	0.90	665
accuracy			0.96	3450
macro avg	0.95	0.93	0.94	3450
weighted avg	0.96	0.96	0.96	3450

Feature	Importance
ApEn	0.0

Feature	Importance
peek	0.03885787561867085
RMS	0.9611421243813291
mean frequency	0.0

Exclude peek

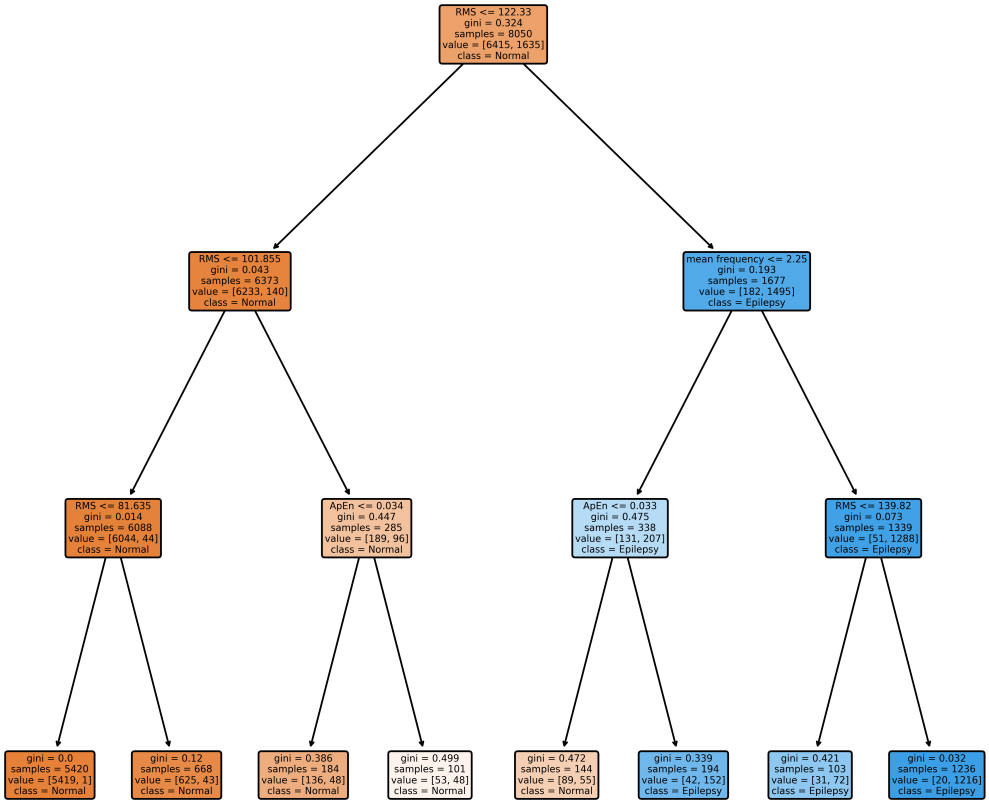


Fig. 5.2 Classification with decision tree
Feature "number of peeks" is excluded.

Accuracy: 0.9585507246376812

	precision	recall	f1-score	support
0	0.97	0.98	0.97	2785
1	0.92	0.86	0.89	665

	precision	recall	f1-score	support
accuracy			0.96	3450
macro avg	0.94	0.92	0.93	3450
weighted avg	0.96	0.96	0.96	3450

Feature	Importance
ApEn	0.014938370928239797
RMS	0.9548974434188772
mean frequency	0.030164185652883146