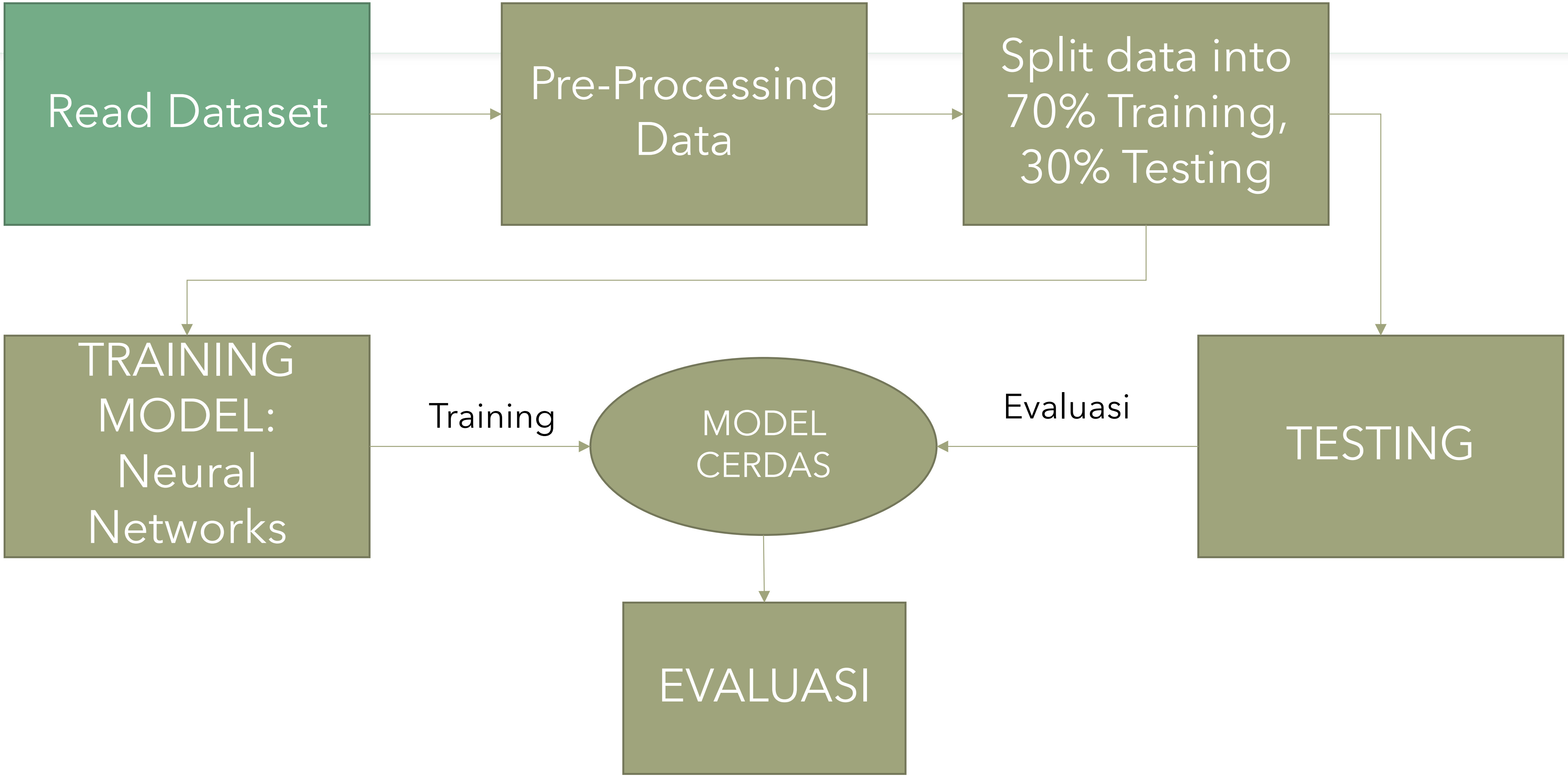
Music Processing dengan Python

M Octaviano Pratama, S.Kom., M.Kom

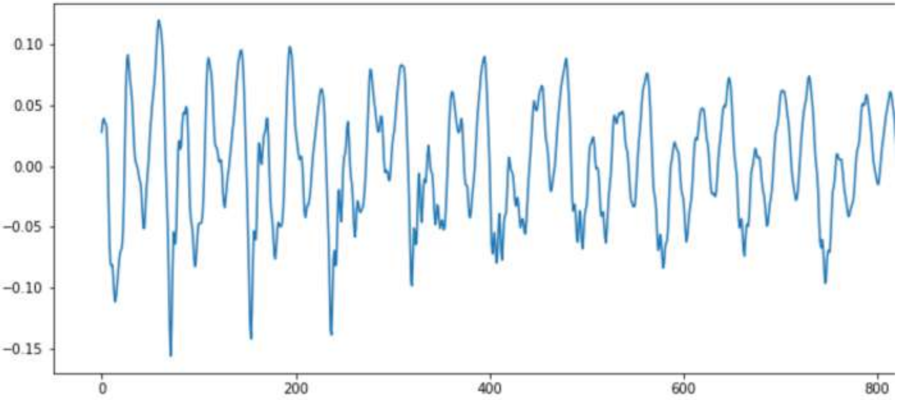
Director BISA AI Academy

Flow Classification: Voice Gender



Flow Classification: Voice Gender Recognition

emosi	sentiment	ZCR	SC	RMSE	SB	SROLL	SFLAT	SCON
4	3	0.430664	5395.540679	0.000003	2970.705638	8914.746094	0.305180	24.323693
4	3	0.040527	1180.375774	0.026604	1557.050021	2713.183594	0.000447	29.543887
4	3	0.068848	1617.700879	0.000417	1895.989101	3186.914062	0.007749	10.379656
4	3	0.074707	2067.990375	0.000701	1784.612375	3552.978516	0.011723	22.355055
4	3	0.065918	2118.206491	0.000601	2251.859553	4618.872070	0.010714	10.943335

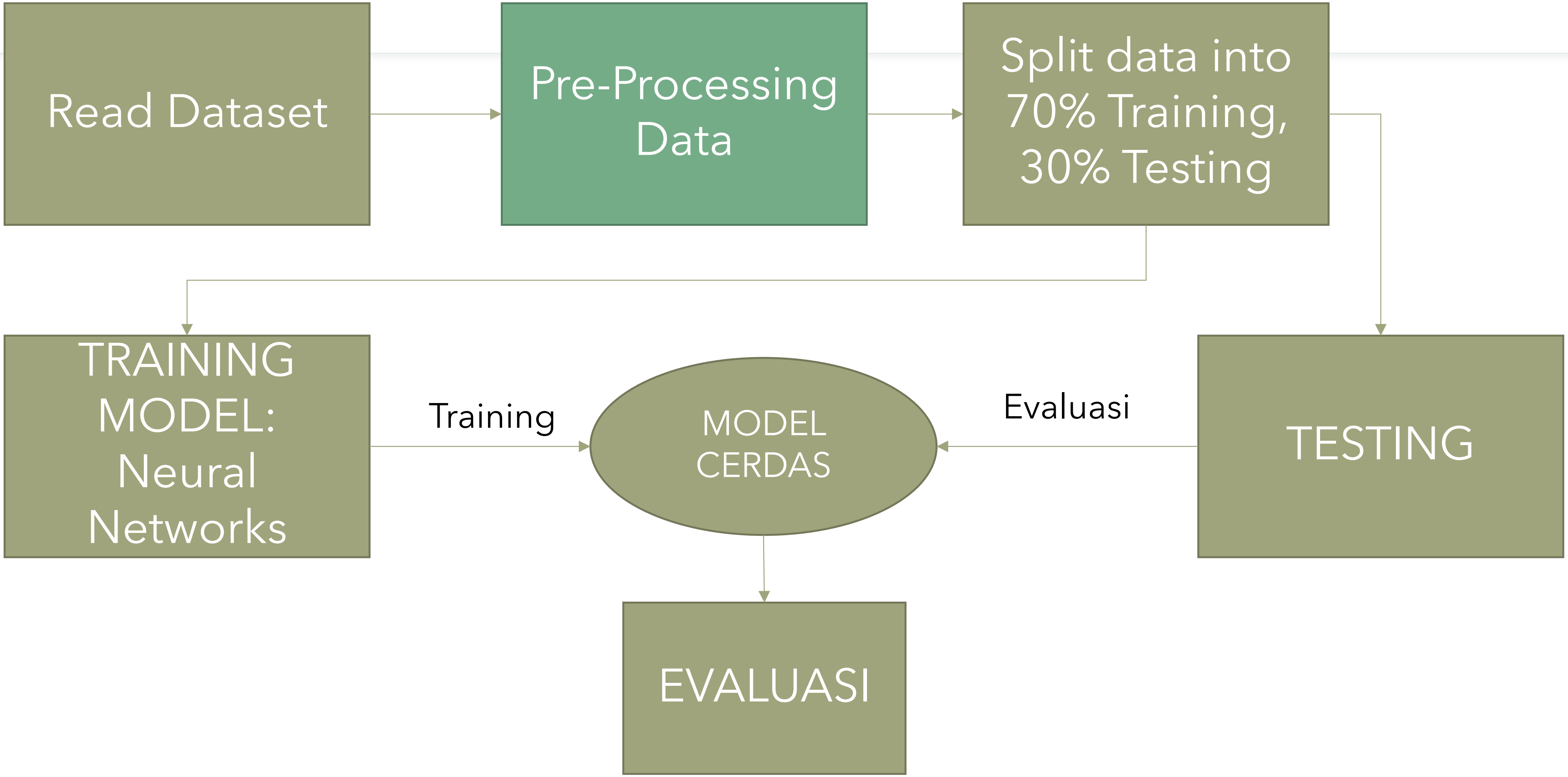


meanfreq	sd	median	Q25	Q75	IQR	skew
0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462
0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285
0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155

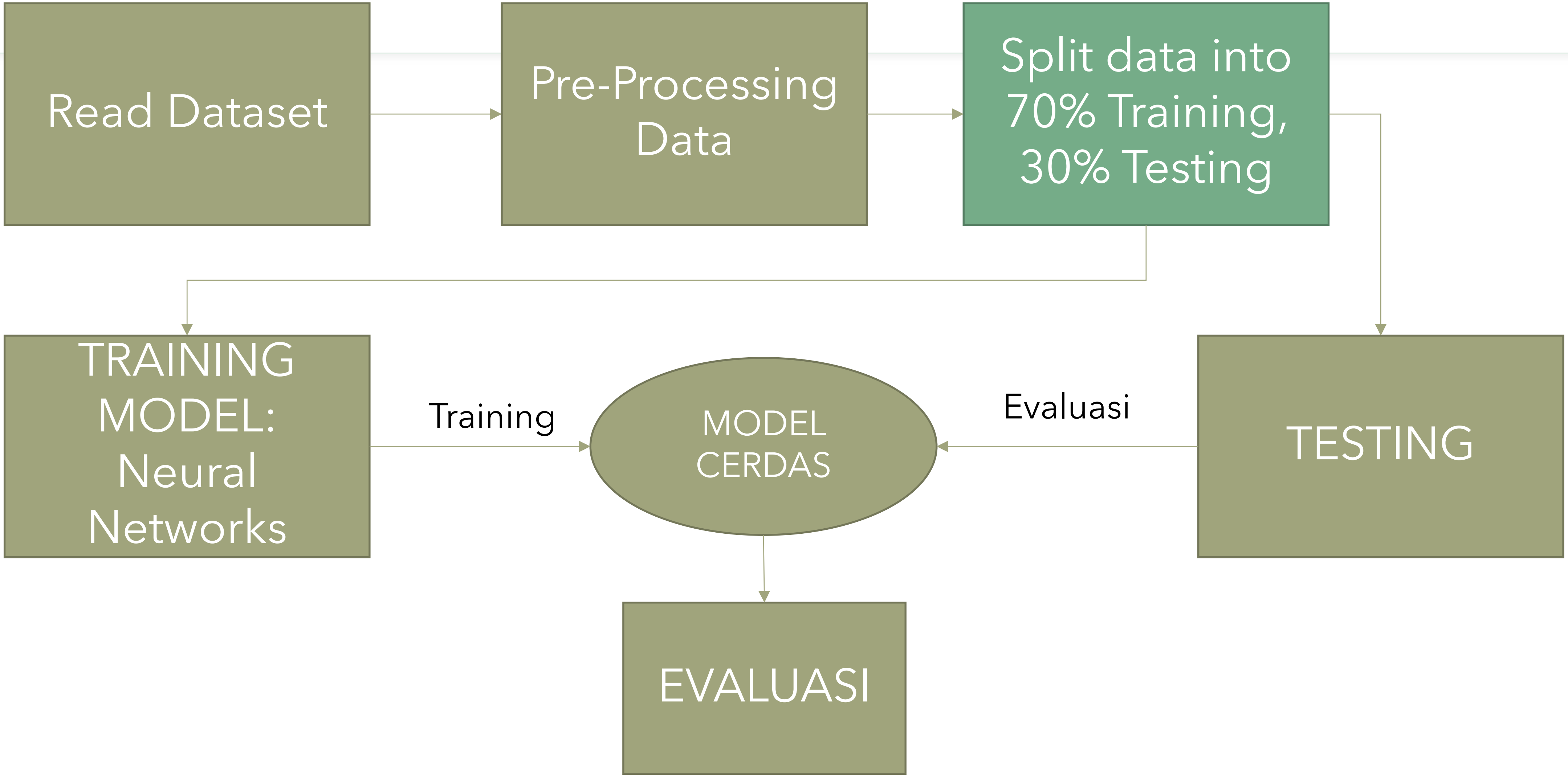
Flow Classification: Voice Gender Recognition

```
data_low_level = []
def extract_low_features(signal):
  zcr = librosa.feature.zero_crossing_rate(signal[0][0])[0, 0]
  sc  = librosa.feature.spectral_centroid(signal[0][0])[0, 0] #average freq
  sb  = librosa.feature.spectral_bandwidth(signal[0][0])[0, 0] #varian
  sroll  =  librosa.feature.spectral_rolloff(signal[0][0])[0, 0] #max freq
  sflat  =  librosa.feature.spectral_flatness(signal[0][0])[0, 0] #flat
  scon  = librosa.feature.spectral_contrast(signal[0][0])[0, 0] #contrast
  rmse = librosa.feature.rmse(signal[0][0])[0, 0]
  mfcc = librosa.feature.mfcc(y=signal[0][0], sr=signal[0][1], n_mfcc=40)

  return zcr, sc, rmse, mfcc, sb, sroll, sflat, scon

for x in audio_spec:
  try:
    data_low_level.append(extract_low_features(x))
  except:
    print("Error Baca File")
```

Flow Classification: Voice Gender



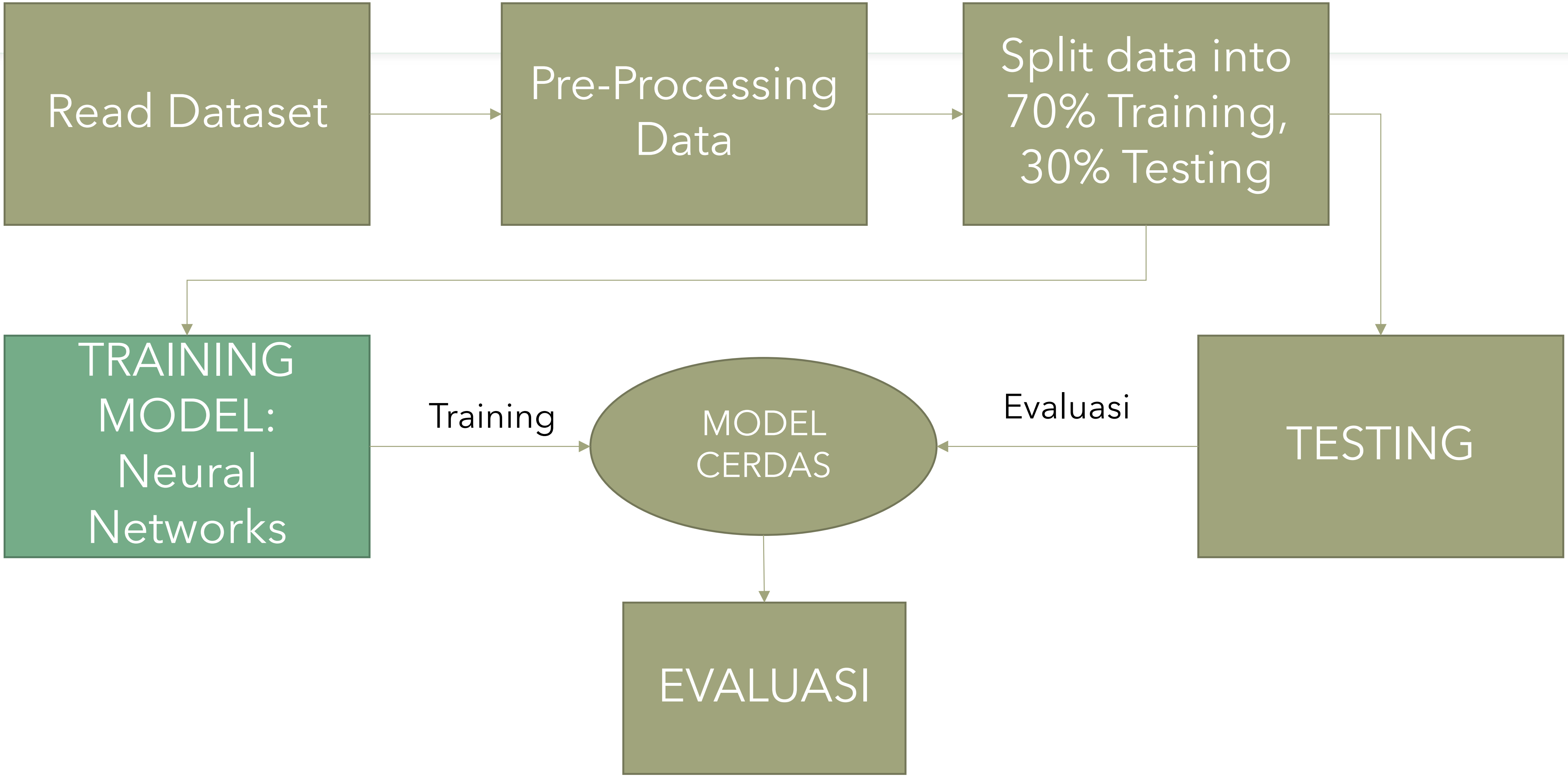
Flow Classification: Voice Gender



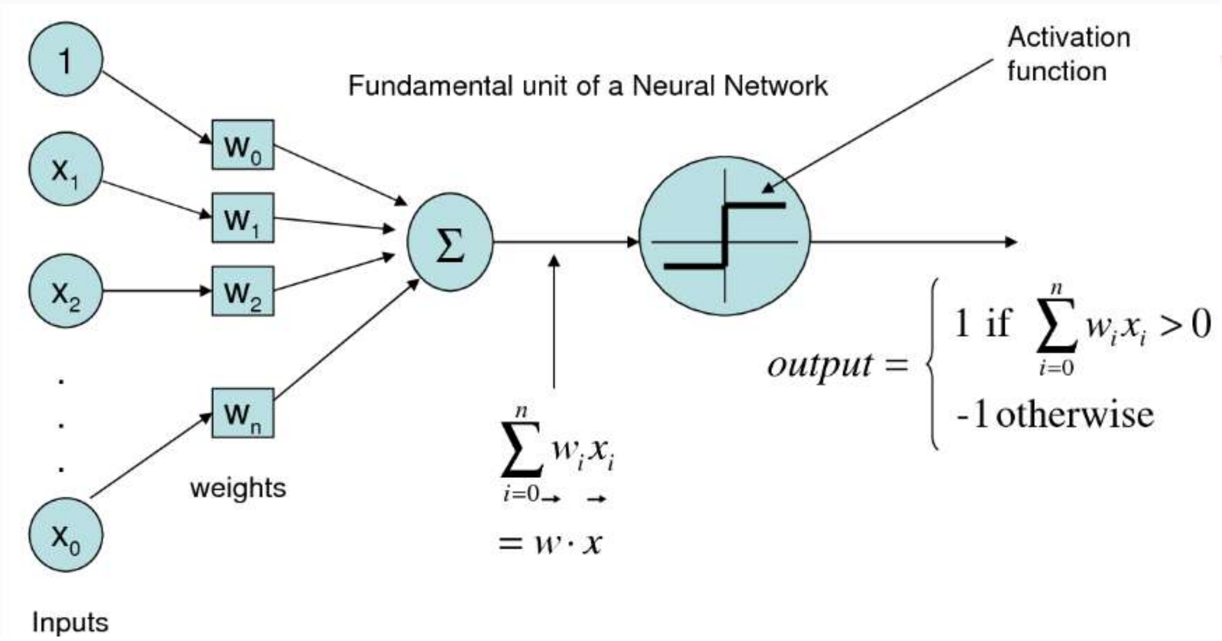
Flow Classification: Voice Gender

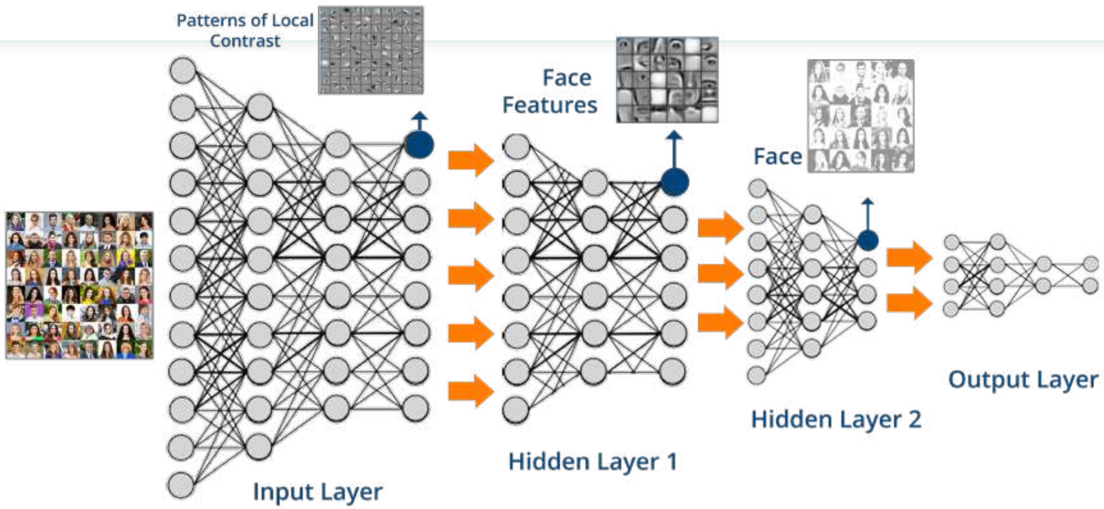
```
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from sklearn import preprocessing #label encoder: categorical --> numeric
from keras.utils import np_utils

X = df.iloc[:, 0:df.shape[1]-1] #dataset_fix yang isinya low level feature kit
y = df.iloc[:, df.shape[1]-1] #dataset_fix untuk class label kita jadikan y

le = preprocessing.LabelEncoder() #panggil LE
le.fit(y)
y = le.transform(y) #ubah class yang masih text ke numeric

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)

y_train_ = to_categorical(y_train, 2) #change label to binary / categorical: [
y_test_ = to_categorical(y_test, 2) #change label to binary / categorical
```

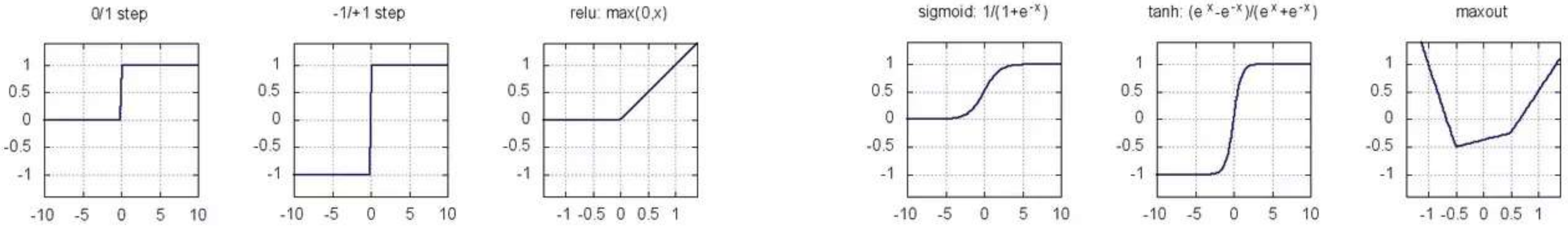
Flow Classification: Contoh Klasifikasi



Neural Networks





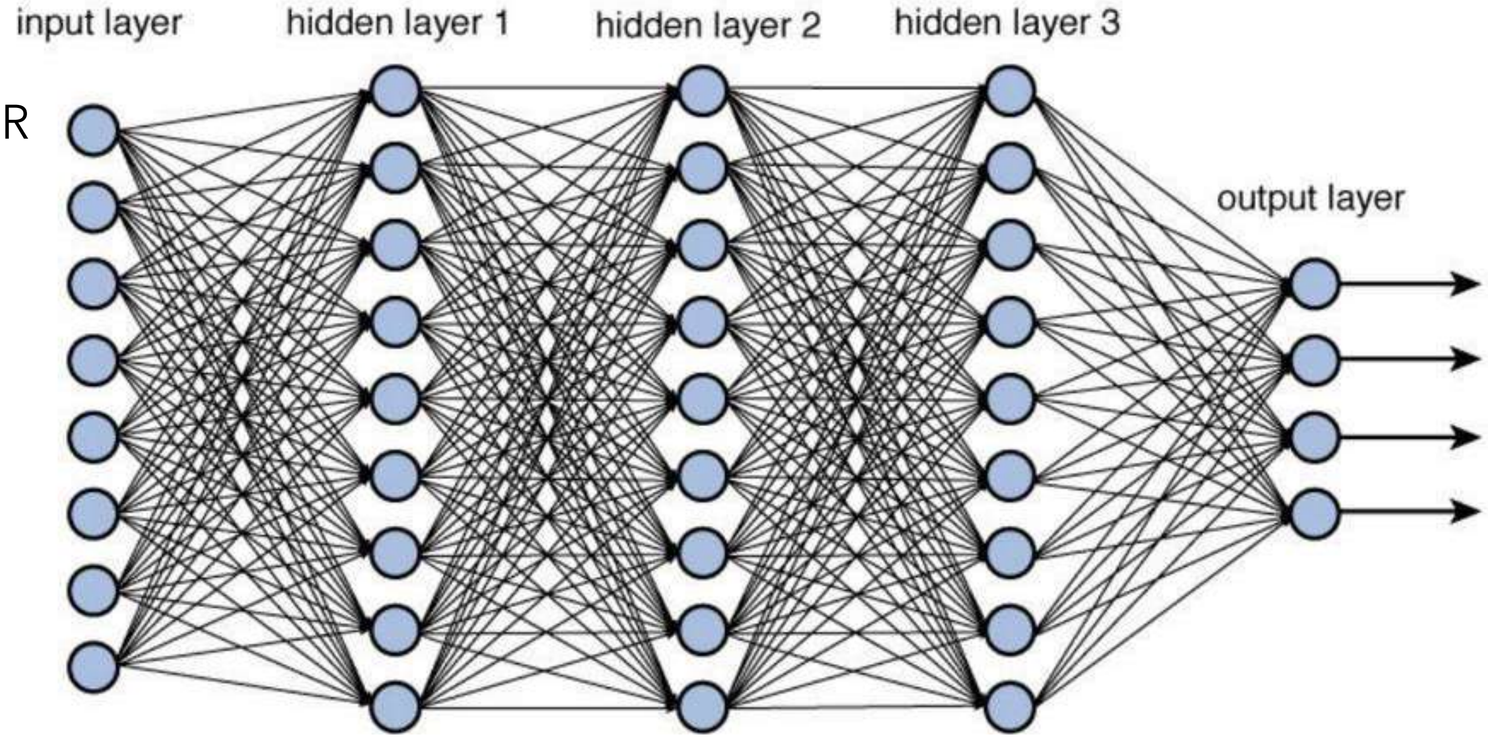


Flow Classification: Machine Learning Model

ZCR

SC

AE

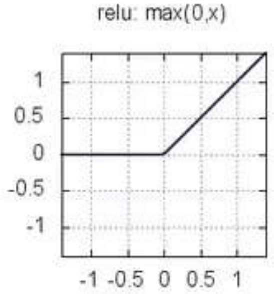


Xi h1i h2i

1 = male
0 = female

$$S(x, W) = \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} W_{(i-m, j-n)}$$

$$Z(S, U) = \sum_{i=1}^{n} \sum_{j=1}^{m} S_{ij} U_{(i-m, j-n)}$$



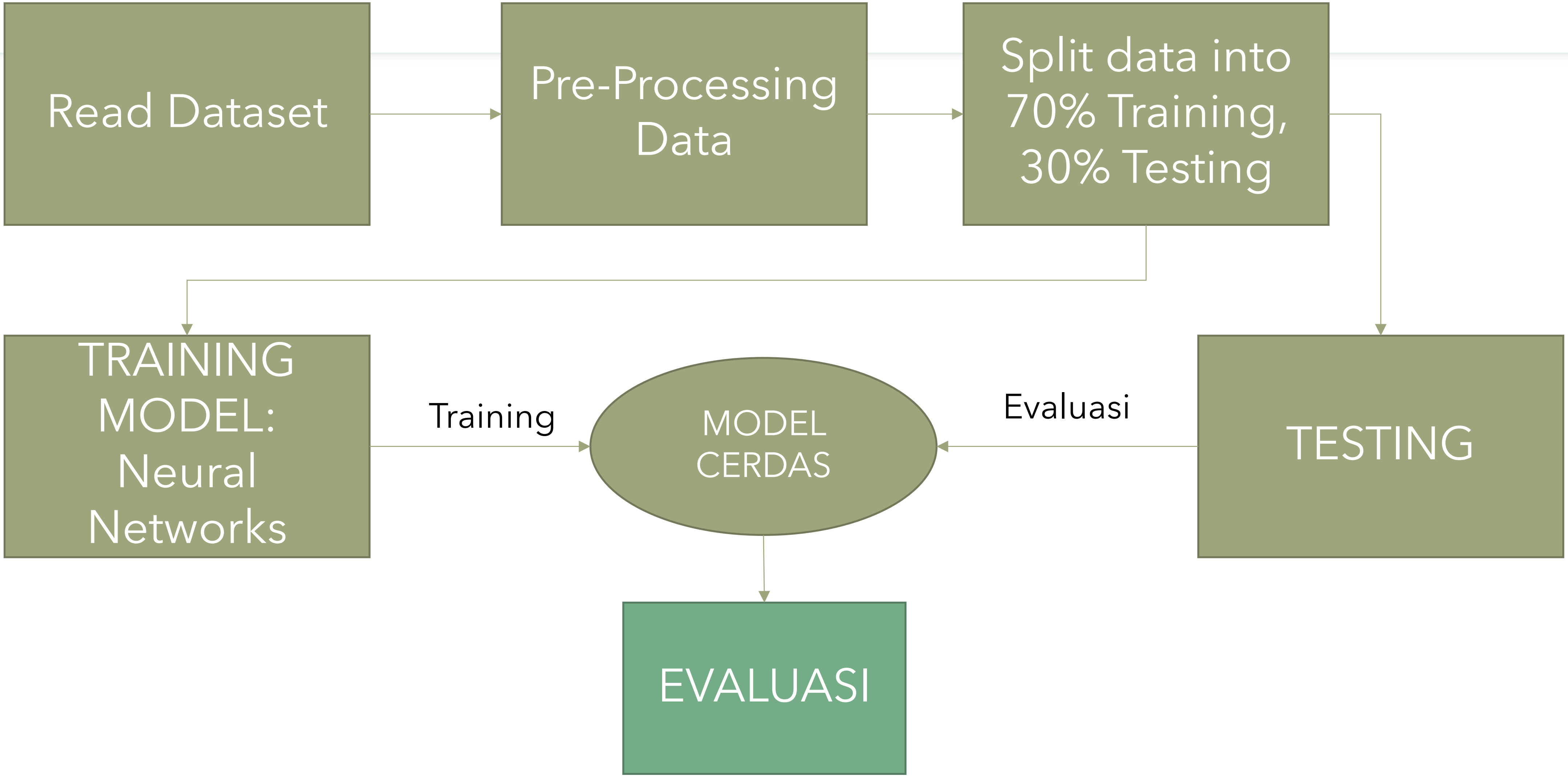
$$ReLU = \begin{cases} x, if\ x > 0 \\ 0, otherwise \end{cases}$$

$$Softmax(z_i) = \frac{\exp(Dl(B_{ij} + h_i, W))}{\sum_{i=1}^{n} \exp(Dl(B_{ij} + h_i, W))}$$

Training Process

```
loss: 0.4712 - acc: 0.8066 - val_loss: 0.4341 - val_acc: 0.8494

loss: 0.4568 - acc: 0.8184 - val_loss: 0.4301 - val_acc: 0.8564

loss: 0.4561 - acc: 0.8189 - val_loss: 0.4374 - val_acc: 0.8546

loss: 0.4509 - acc: 0.8202 - val_loss: 0.4273 - val_acc: 0.8476
```

Flow Classification: Contoh Klasifikasi

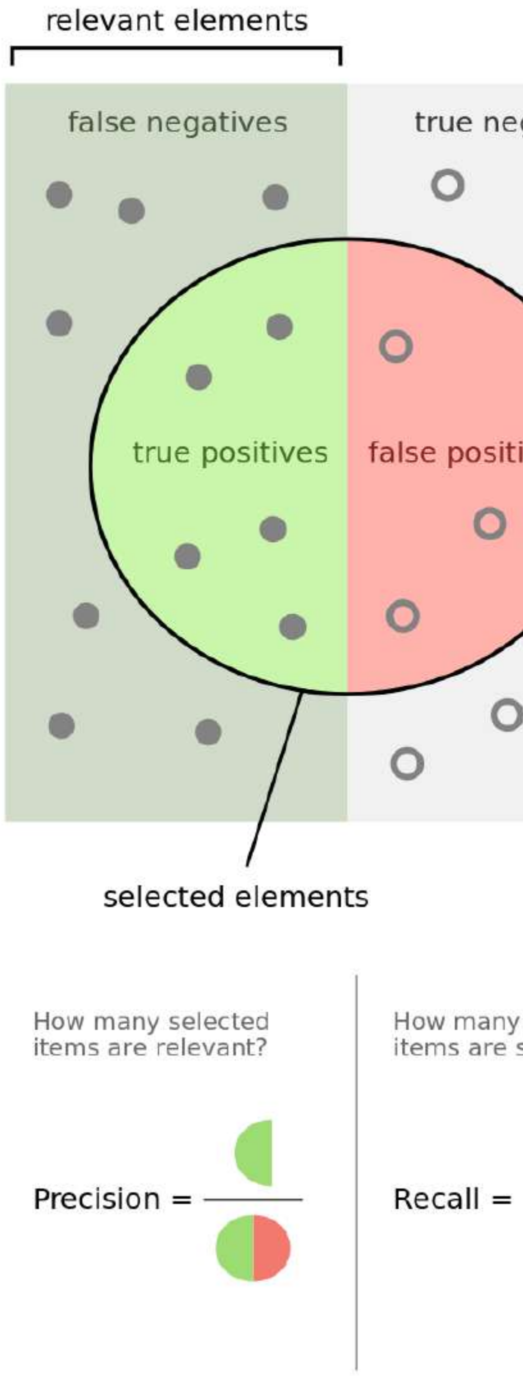


Precision Recall + Confusion Matrix

$$\text{Precision} = \frac{tp}{tp + fp} \qquad \text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

$$\text{Recall} = \frac{tp}{tp + fn} \qquad F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

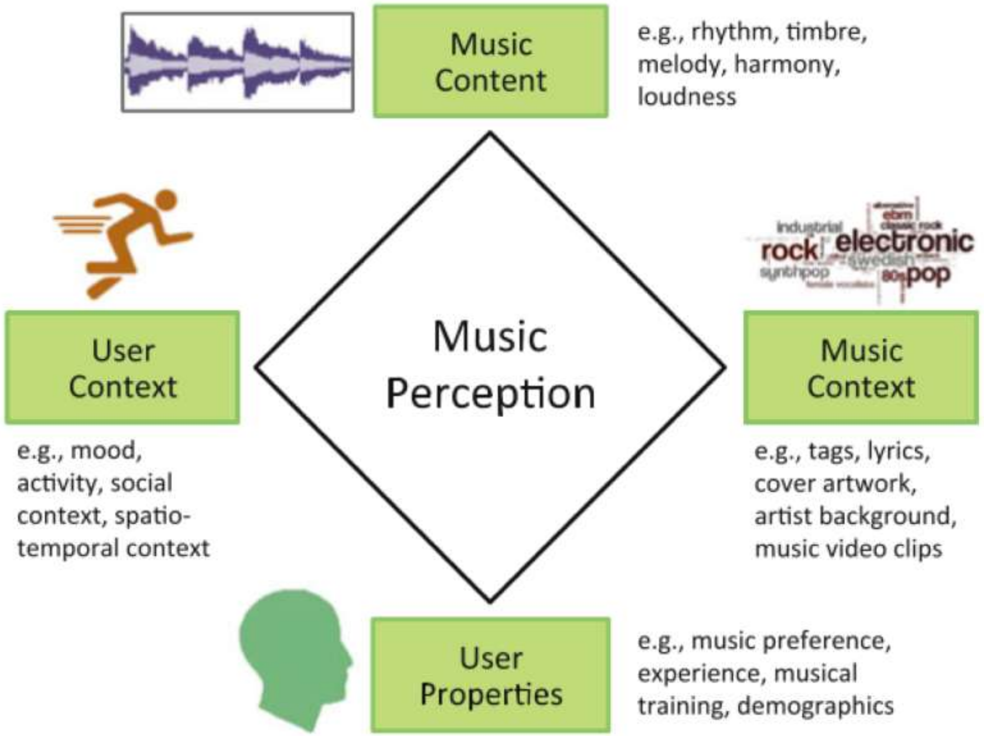
```
[[249,   0,   8,   0,  10,   0,   7,   4,   0],
 [  0, 261,   4,   0,   0,   0,   0,   1,   4],
 [ 15,   3, 232,   0,   1,   0,   0,   2,   0],
 [  0,   0,   0, 363,   0,   7,   1,   0,   0],
 [ 63,   1,   7,  16,  14,   5,  13,  12,   0],
 [  1,   0,   0,  35,   1,  15,  11,   0,   0],
 [  0,   0,   0,   0,   0,   0, 393,   1,   0],
 [  2,   0,   0,   0,   0,   0,   2, 514,   0],
 [  0,  55,   2,   0,   0,   0,   0,   0,  50]])
```

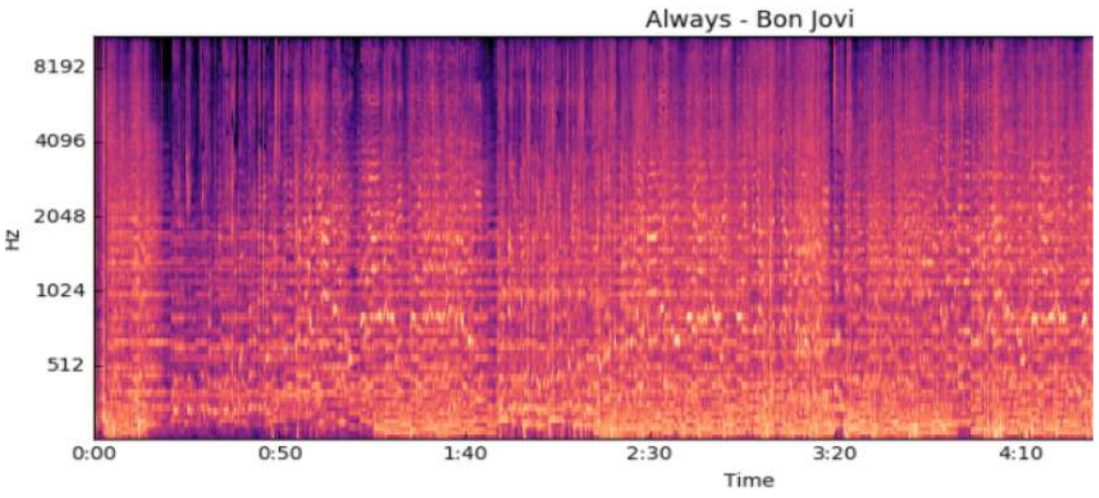


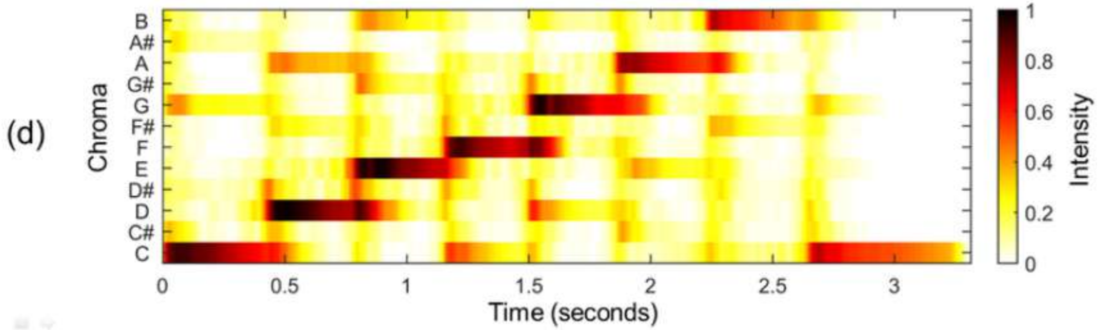
Flow Classification: Evaluasi

precision	recall	f1-score	support
0.77	0.80	0.79	41
0.83	0.80	0.82	50
0.80	0.80	0.80	91

Music Retrieval





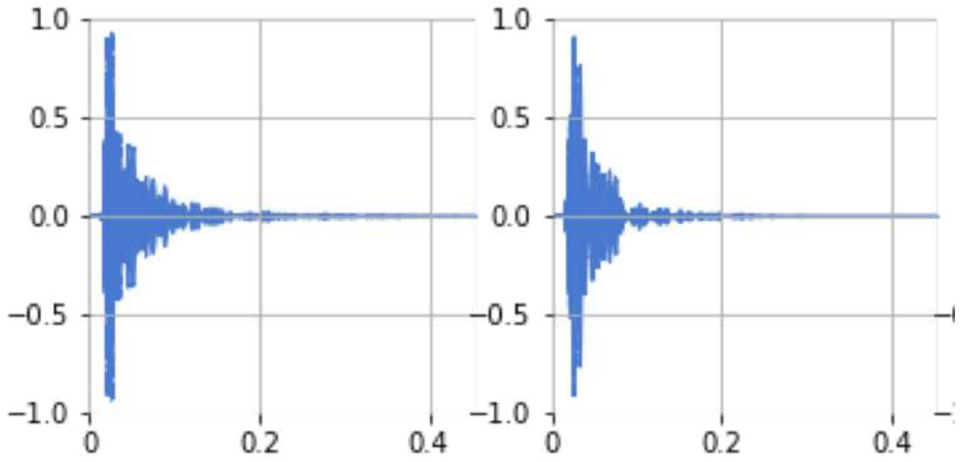


Acoustic Feature

- **Low Level Feature:** Zero Crossing Rate, Bandwidth, Spectral Coeficient, Energy, RMSE
- **High Level Feature:** MFCC, Spectogram, Chroma Feature

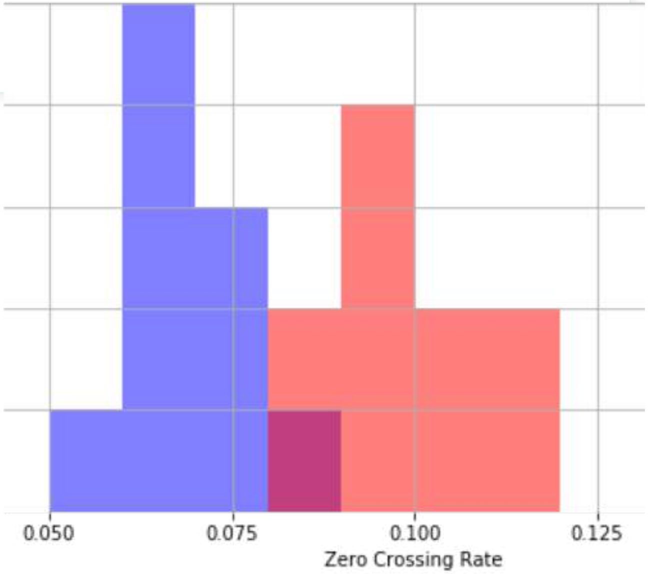
Low Level: Basic Feature Extraction

```
kick_signals = [
        librosa.load(p)[0] for p in Path().glob('kick*.mp3’)
]
snare_signals = [
        librosa.load(p)[0] for p in Path().glob('snare_*.mp3’)
]
len(kick_signals)
plt.figure(figsize=(15, 6))
for i, x in enumerate(kick_signals):
        plt.subplot(2, 5, i+1)
        librosa.display.waveplot(x[:10000])
        plt.ylim(-1, 1)
```

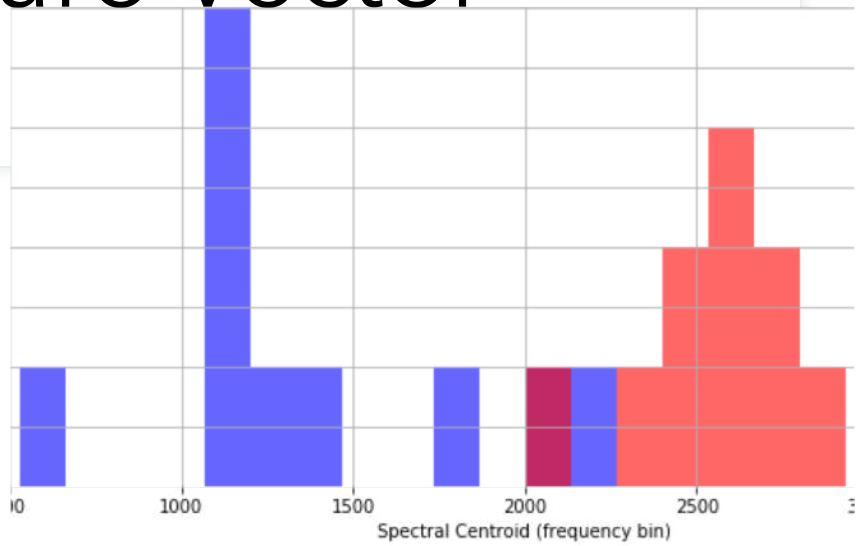


Low Level: Constructing Feature Vector

```
def extract_features(signal):
        return [
                librosa.feature.zero_crossing_rate(signal)[0, 0],
                librosa.feature.spectral_centroid(signal)[0, 0]
        ]
kick_f = numpy.array([extract_features(x) for x in kick_signals])
snare_f = numpy.array([extract_features(x) for x in snare_signals])
plt.figure(figsize=(14, 5))
plt.hist(kick_features[:,0], color='b', range=(0, 0.2), alpha=0.5, bins=20)
plt.hist(snare_features[:,0], color='r', range=(0, 0.2), alpha=0.5, bins=20)
plt.legend(('kicks', 'snares’))
plt.xlabel('Zero Crossing Rate') plt.ylabel('Count')
```



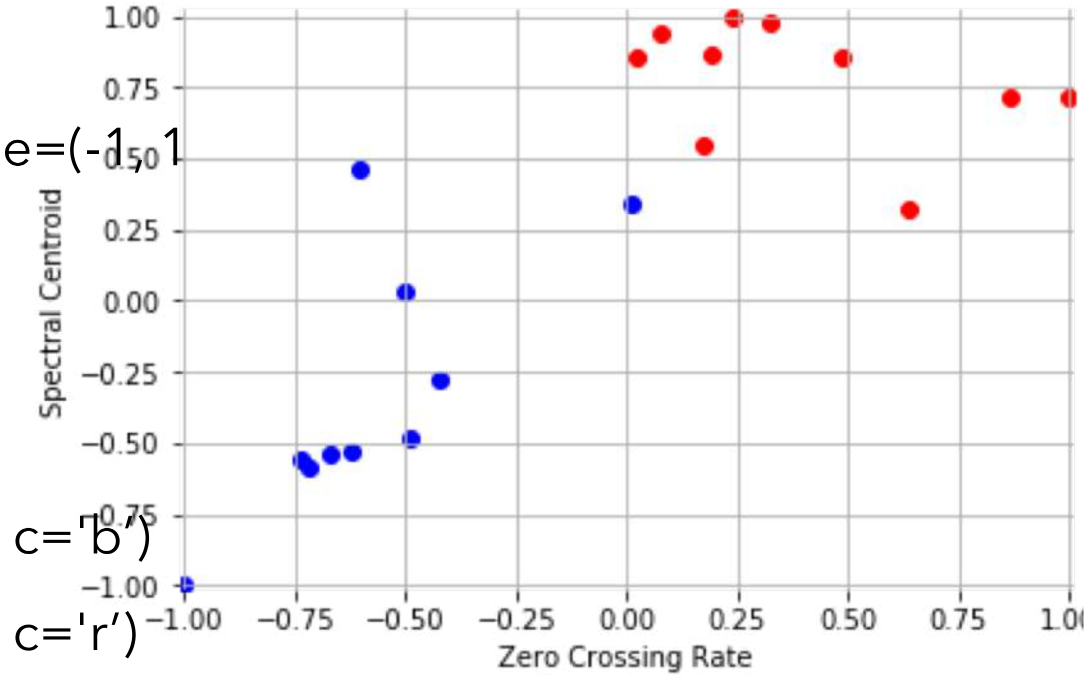
Low Level: Constructing Feature Vector



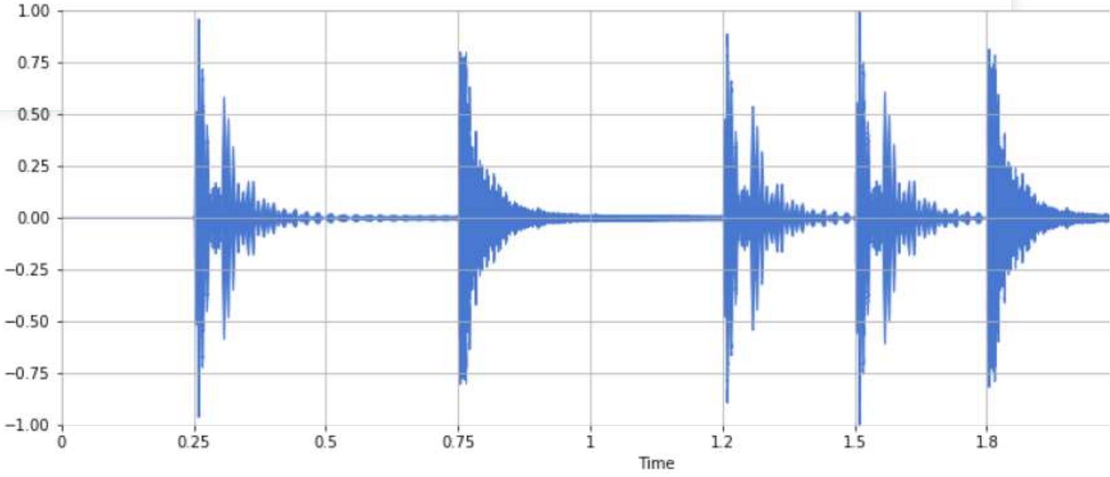

```
def extract_features(signal):
        return [
                librosa.feature.zero_crossing_rate(signal)[0, 0],
                librosa.feature.spectral_centroid(signal)[0, 0]
        ]
kick_f = numpy.array([extract_features(x) for x in kick_signals])
snare_f = numpy.array([extract_features(x) for x in snare_signals])
plt.figure(figsize=(14, 5))
plt.hist(kick_features[:,1], color='b', range=(0, 4000), bins=30, alpha=0.6)
plt.hist(snare_features[:,1], color='r', range=(0, 4000), bins=30, alpha=0.6)
plt.legend(('kicks', 'snares'))
plt.xlabel('Spectral Centroid (frequency bin)') plt.ylabel('Count')
```

Low Level: Feature Scaling

```
feature_table = numpy.vstack((kick_features, snare_features))
print(feature_table.shape)
scaler = sklearn.preprocessing.MinMaxScaler(feature_range=(-1, 1
training_features = scaler.fit_transform(feature_table)
print(training_features.min(axis=0))
print(training_features.max(axis=0))
plt.scatter(training_features[:10,0], training_features[:10,1], c='b’)
plt.scatter(training_features[10:,0], training_features[10:,1], c='r’)
plt.xlabel('Zero Crossing Rate') plt.ylabel('Spectral Centroid')
```



Low Level: Energy and RMSE

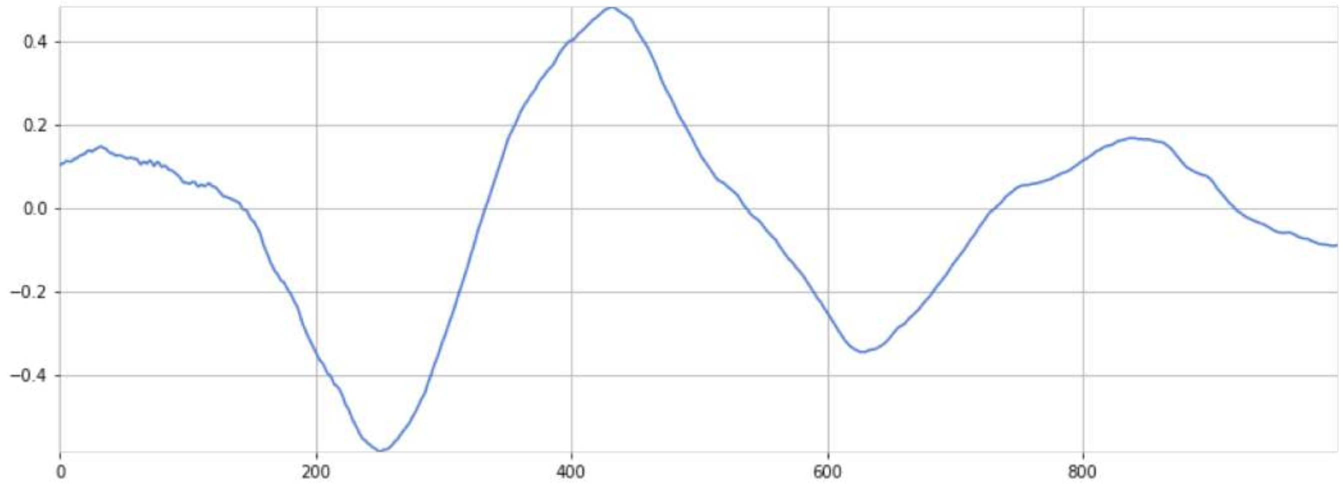


$$Energy = \sum |x(n)|^2 \qquad RMSE = \frac{1}{n}\sqrt{\sum |x(n)|^2}$$

```
x, sr = librosa.load(simple_loop.wav')

librosa.get_duration(x, sr)

hop_length = 256

frame_length = 512

energy = numpy.array(

        [ sum(abs(x[i:i+frame_length]**2))

        for i in range(0, len(x), hop_length) ]

)

rmse = librosa.feature.rmse(x, frame_length=frame_length, hop_length=hop_length, center=True)

frames = range(len(energy)) t = librosa.frames_to_time(frames, sr=sr, hop_length=hop_length)
```

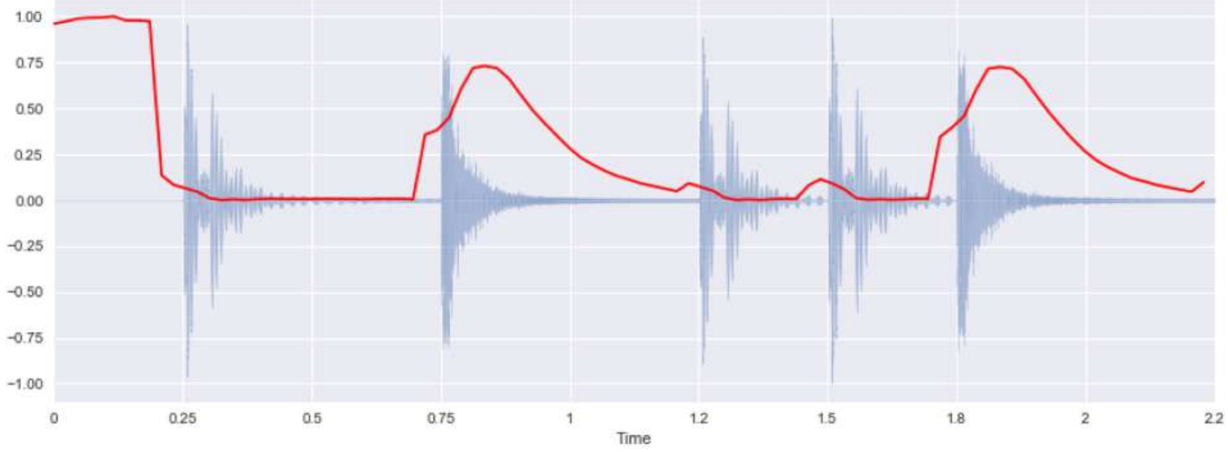
Low Level: Zero Crossing Rate

```
x, sr = librosa.load('audio/simple_loop.wav')
plt.figure(figsize=(14, 5))
librosa.display.waveplot(x, sr=sr)


n0 = 6500
n1 = 7500
plt.figure(figsize=(14, 5))
plt.plot(x[n0:n1])
```



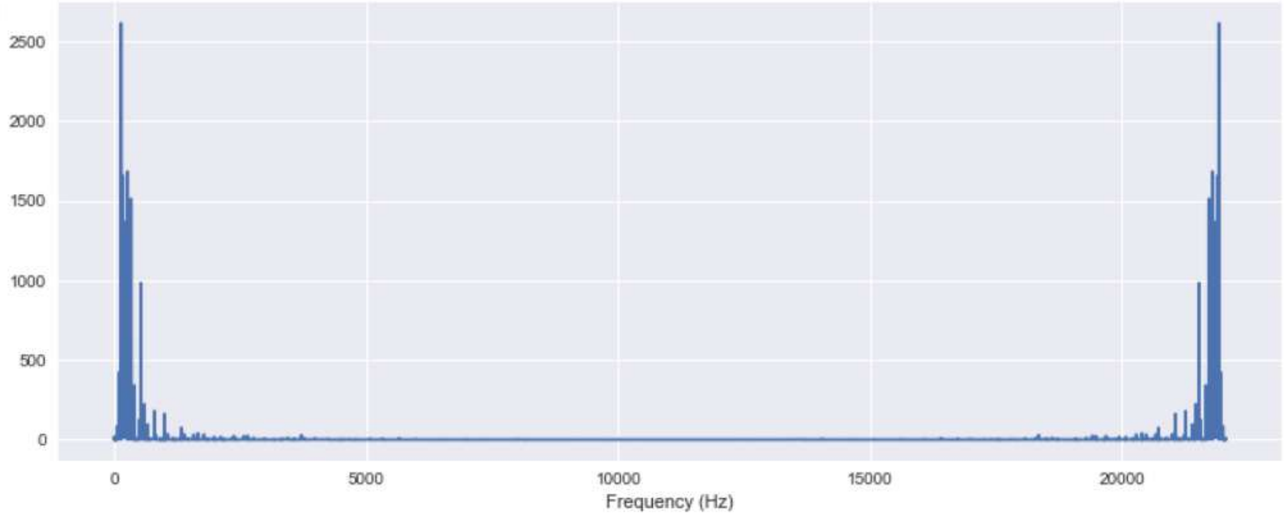
Low Level: Spectral Centroid

```
def normalize(x, axis=0):
        return sklearn.preprocessing.minmax_scale(x, axis=axis)


x, sr = librosa.load('audio/simple_loop.wav’)
ipd.Audio(x, rate=sr)
spectral_centroids = librosa.feature.spectral_centroid(x, sr=sr)[0]
spectral_centroids.shape
frames = range(len(spectral_centroids))
 t = librosa.frames_to_time(frames)
librosa.display.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_centroids), color='r
```

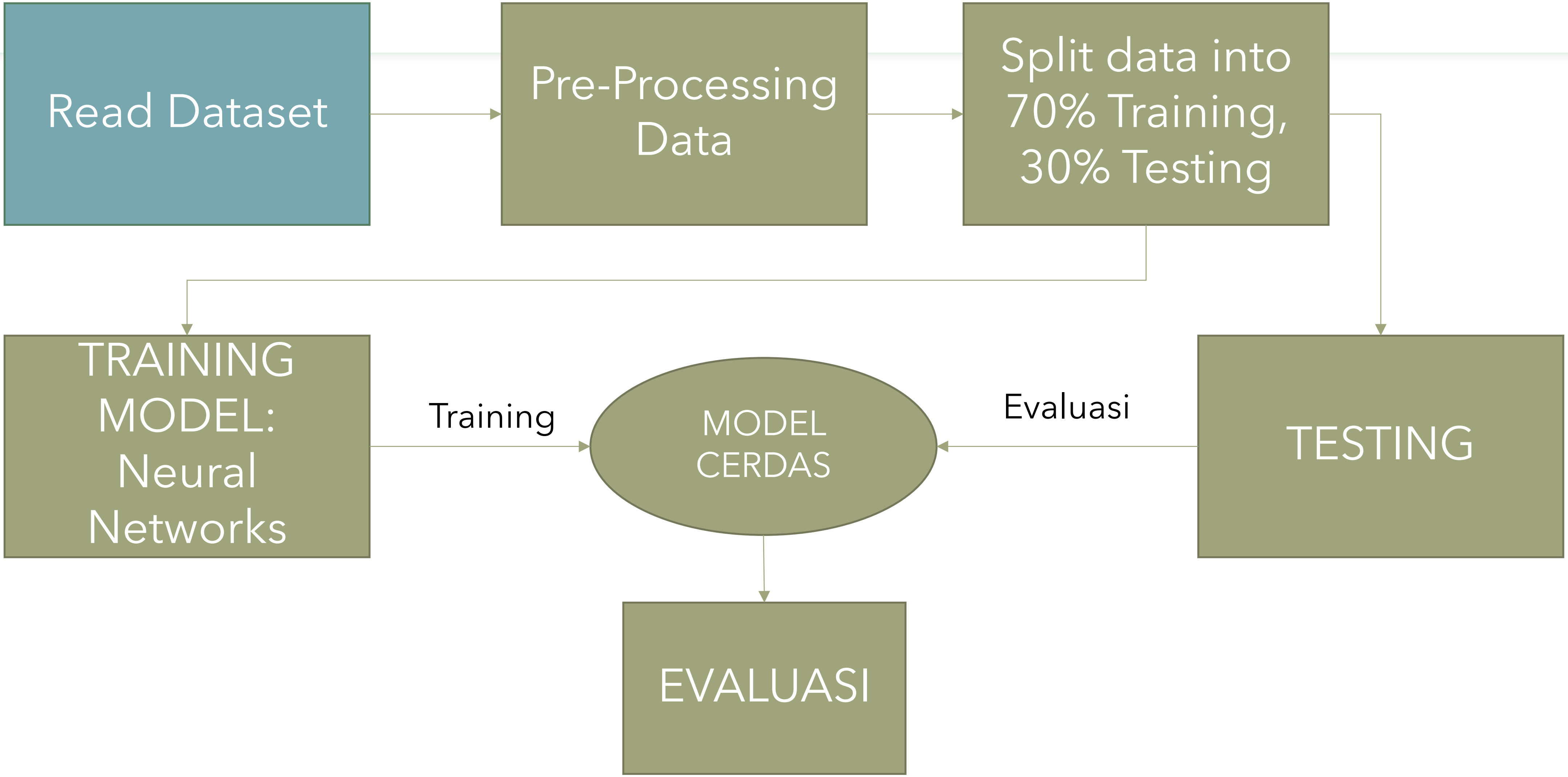


Low Level: Fourier Transform

```
X = scipy.fft(x)

X_mag = numpy.absolute(X)

f = numpy.linspace(0, sr, len(X_mag))


plt.figure(figsize=(13, 5))

plt.plot(f, X_mag)

plt.xlabel('Frequency (Hz)')


#ZOOM IN

plt.figure(figsize=(13, 5))

plt.plot(f[:5000], X_mag[:5000])

plt.xlabel('Frequency (Hz)')
```



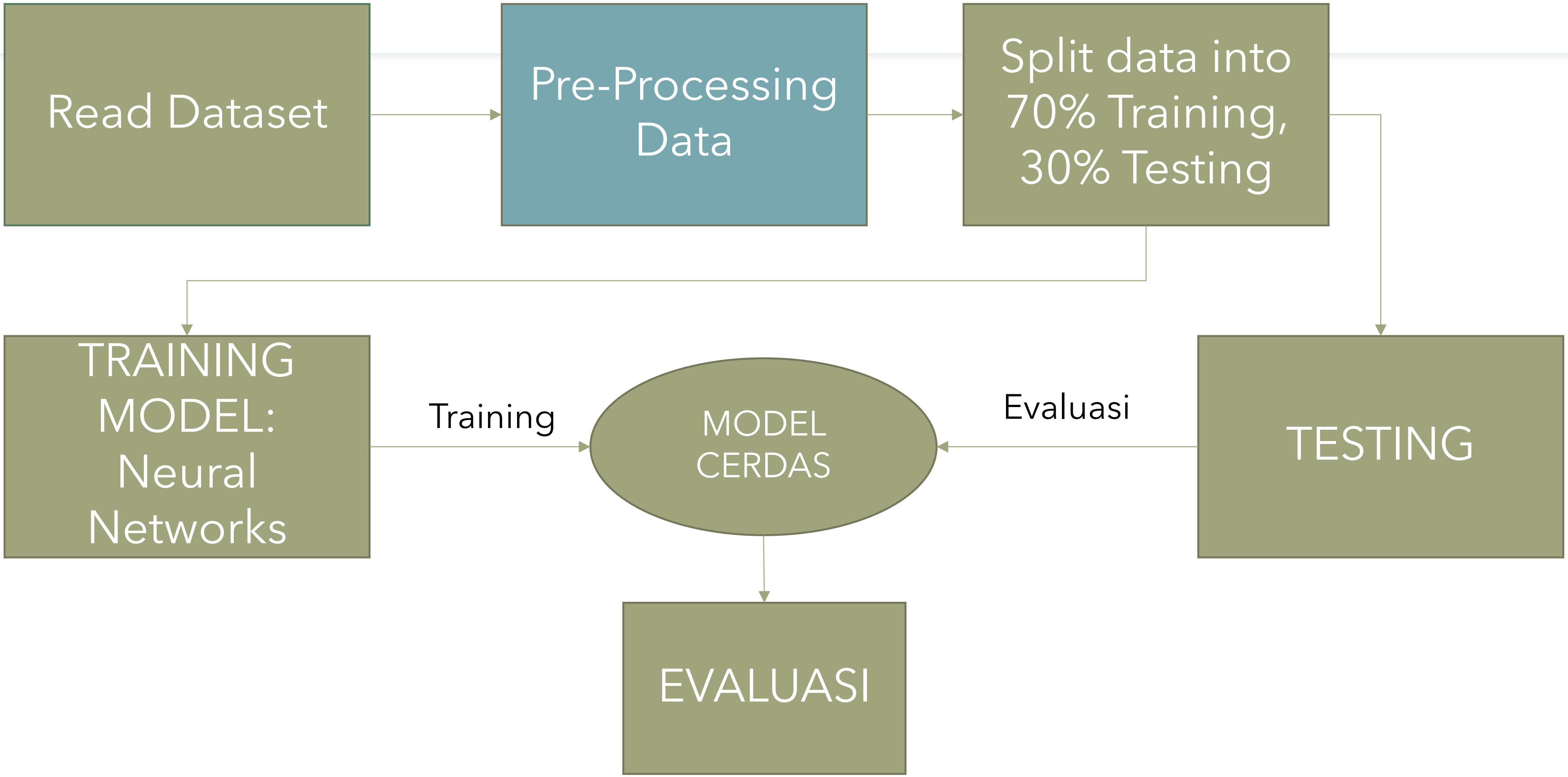
Flow Classification: Voice Gender



Method: Dataset Collection

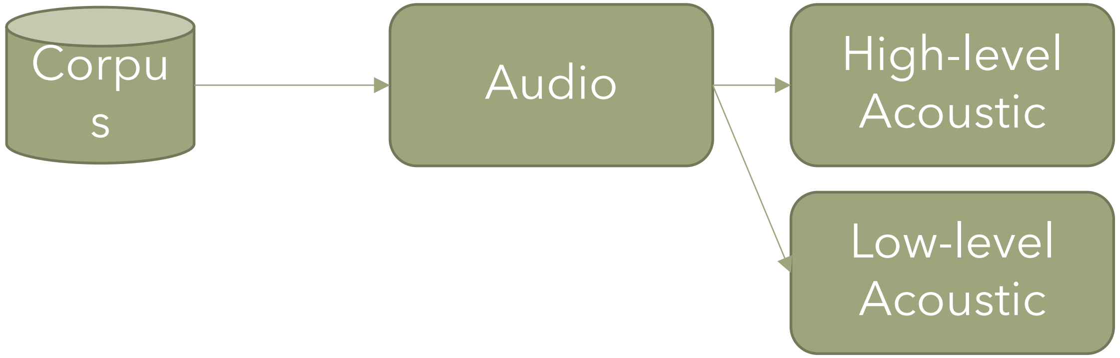
- Dataset diambil dari Talkshow Indonesia Lawyer Club.
- Segmentasi oleh 2 orang annotators. Segmentasi dilakukan per 1 kalimat di speech. Setelah dipisahkan, dicoba diberikan label emosi sesuai dengan konsep Emosi Valence-Arousal. Persetujuan diukur oleh Kappa Score untuk level agreement.
- Setiap segment akan dipisahkan audio dan text.

Flow Classification: Voice Gender



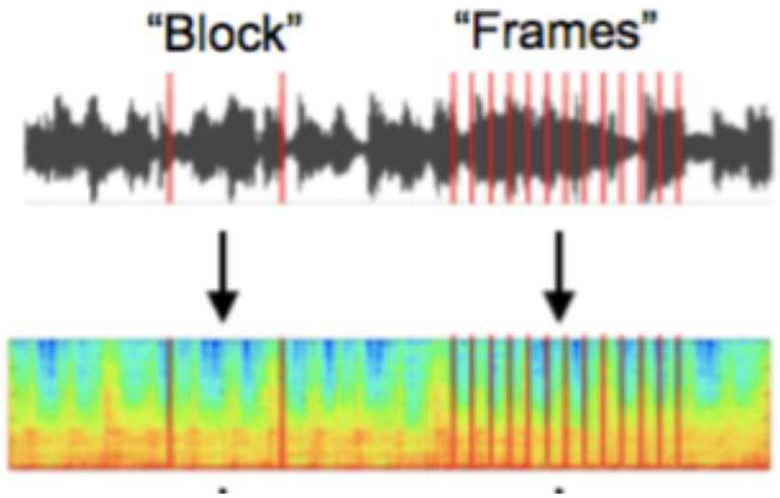
Method: Features

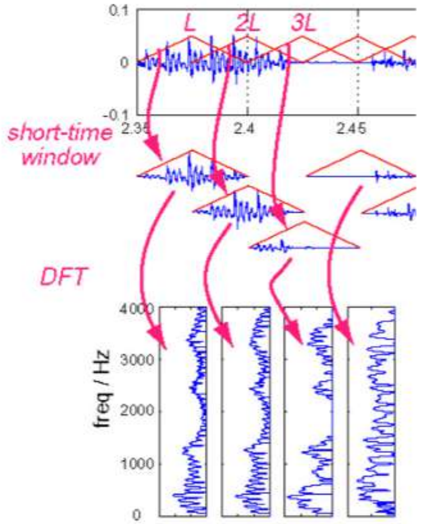
- Extract from High-level Acoustic and Low-level Acoustic Features



Ekstraksi Mel-Spectrogram

1. Bagi setiap 3 detik *track* lagu kedalam *overlapping frame,* setiap durasi 25ms. Umumnya, dari satu frame ke frame lainnya digunakan pergeseran 5ms.
2. Berikan *Fourier transform* pada setiap *frame* dan tumpuk dalam sumbu frekuensi dan waktu
3. Berikan *Triangular Filter Bank* untuk mendapatkan respon frekuensi setiap *frame* pada *mel-scale.*
4. Untuk mendapatkan *mel-spectrogram*, berikan logaritma pada intensitas spectral.
5. Setiap 3 detik lagu direpresentasikan sebagai 600 x 128 tensor
6. Fitur disediakan oleh library Librosa



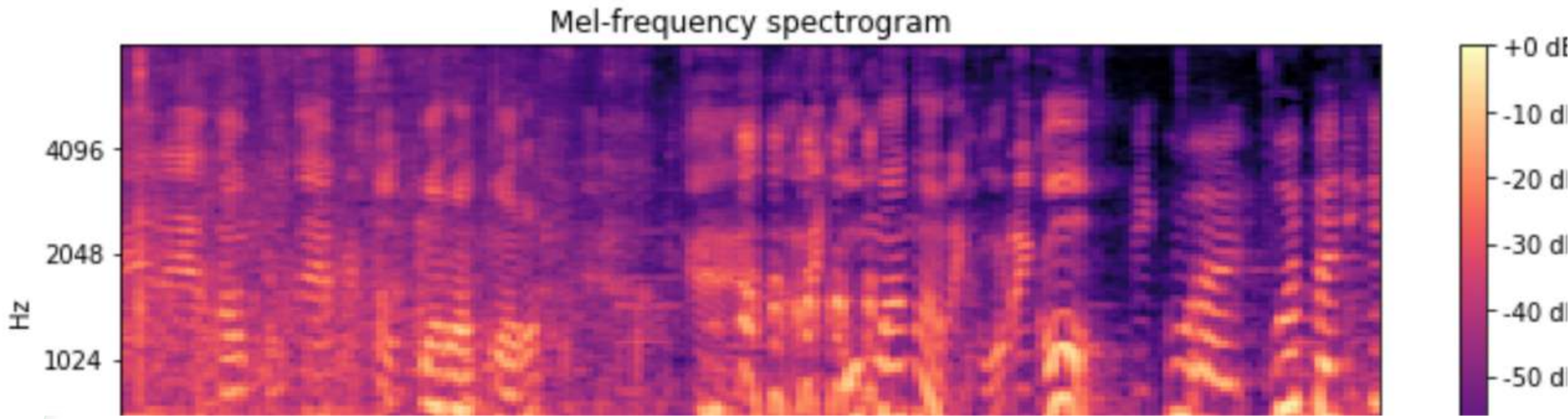


Read, preprocessing

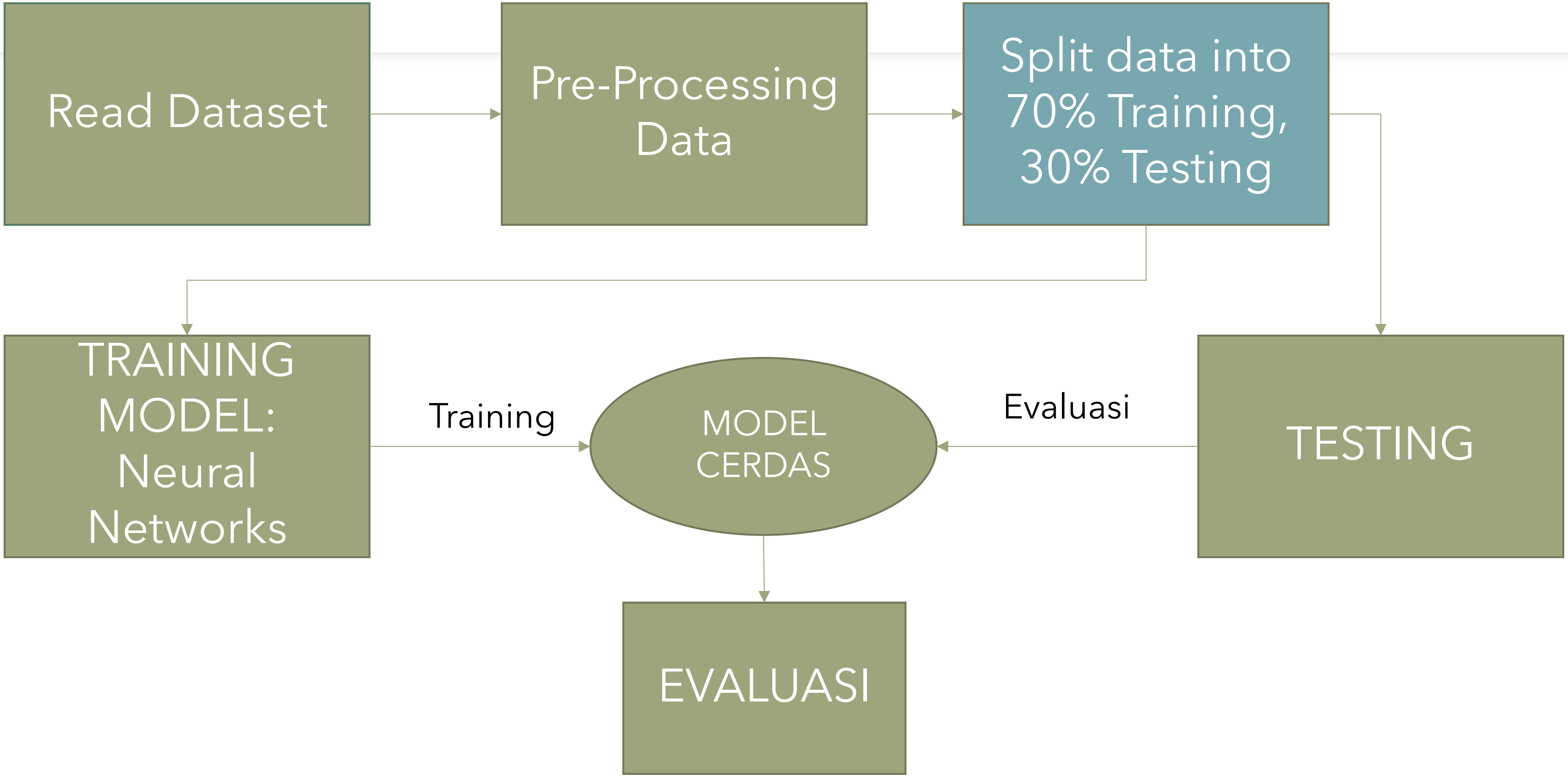
```
dirs = os.listdir('/content/drive/My Drive/DATASET/spectro
label = 0
im_arr = []
lb_arr = []
X = []
y = []
for i in dirs: #loop all directory
    count = 0
    for pic in glob.glob('/content/drive/My Drive/DATASET/
        im = cv2.imread(pic) #open image
        im = cv2.resize(im,(70,70))
        im = np.array(im) #change into array
        count = count + 1
        X.append(im)
        y.append(label)
        if(count == 3): #SAmple
            im_arr.append({str(i):im})
    print("Jumlah "+str(i)+" : "+str(count))
    label = label + 1
    lb_arr.append(i)
X = np.array(X)
y = np.array(y);
```

Read, preprocessing

```
import matplotlib.pyplot as plt
import librosa

y, sr = librosa.load("/content/drive/My Drive/DATASET/chunk2.wav")
S = librosa.feature.melspectrogram(y=y, sr=sr, n_mels=128, fmax=8000)
plt.figure(figsize=(10, 4))
S_dB = librosa.power_to_db(S, ref=np.max)
librosa.display.specshow(S_dB, x_axis='time', y_axis='mel', sr=sr,fmax=8000)
plt.colorbar(format='%+2.0f dB')
plt.title('Mel-frequency spectrogram')
plt.tight_layout()
plt.show()
```



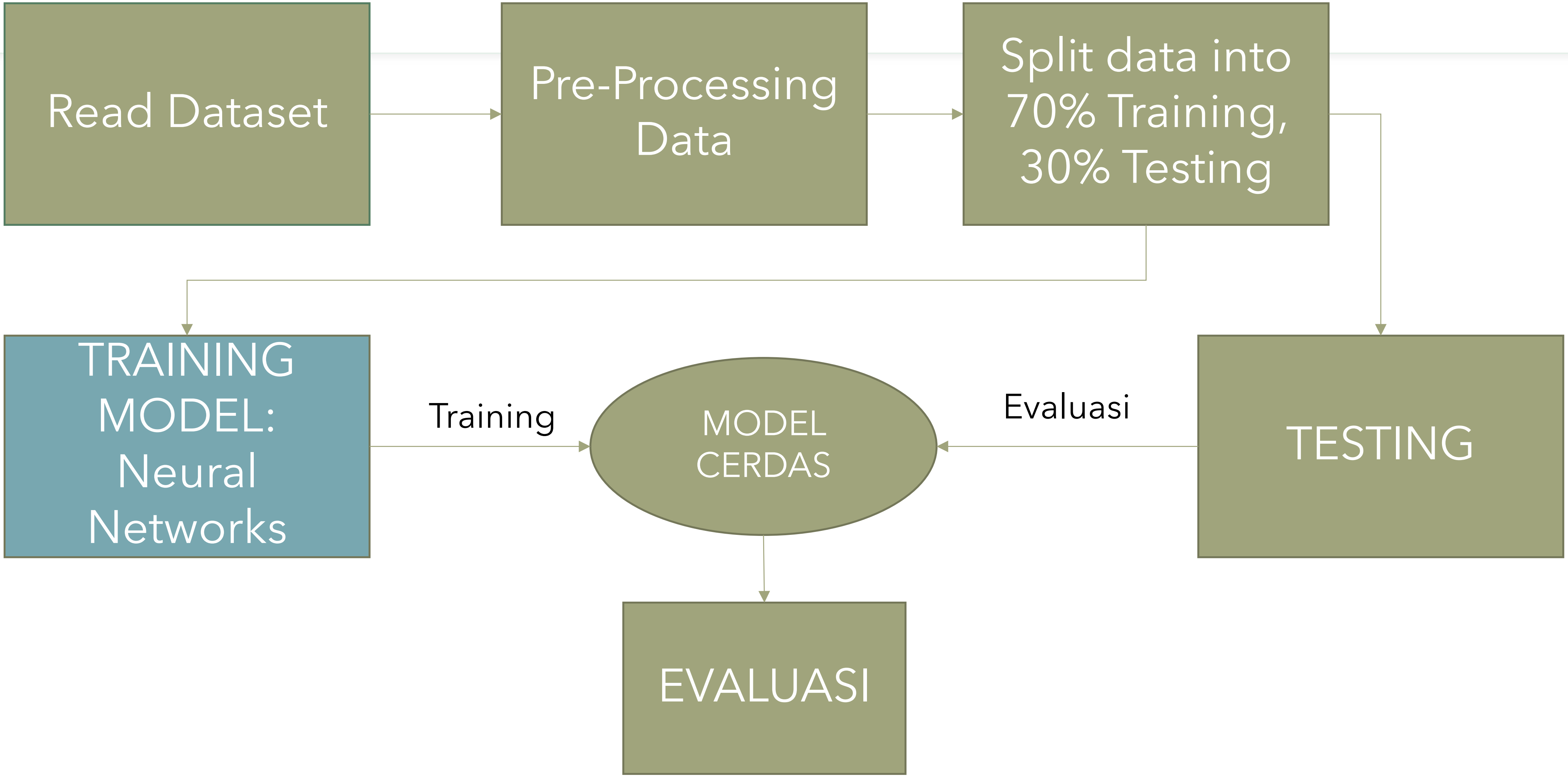
Flow Classification: Voice Gender



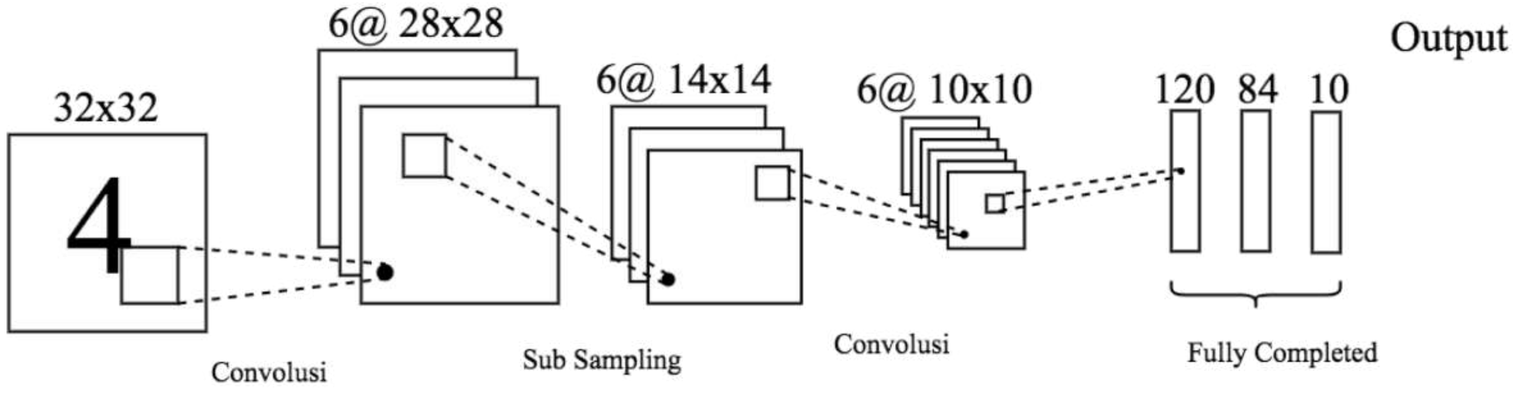
Split Data

```
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

X_train, X_test, y_train, y_test = train_test_split(X, y,

X_train = X_train.astype('float32') #set x_train data typ
X_test = X_test.astype('float32') #set x_test data type a
X_train /= 255 #change x_train value between 0 - 1
X_test /= 255 #change x_test value between 0 - 1
y_train = to_categorical(y_train, 5) #change label to bin
y_test = to_categorical(y_test, 5) #change label to binar
```

Flow Classification: Voice Gender



Convolutional Neural Networks



$$S(x, W) = \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} W_{(i-m, j-n)}$$

$$Softmax(z_i) = \frac{\exp(z_i)}{\sum_{i=1}^{n} \exp(z_j)}$$

Training Neural Networks

```
# ARSITEKTUR
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropo
model = Sequential() #model = sequential
model.add(Conv2D(32, kernel_size=(3, 3),activation='
model.add(MaxPooling2D(pool_size=(2,2))) #max poolin
model.add(Conv2D(32, (3, 3), activation='relu')) #la
model.add(MaxPooling2D(pool_size=(2,2))) #max poolin
model.add(Dropout(0.25)) #delete neuron randomly whi
model.add(Flatten()) #make layer flatten
model.add(Dense(128, activation='relu')) #fully conn
model.add(Dropout(0.5)) #delete neuron randomly and
model.add(Dense(5, activation='softmax')) #softmax w
```