# DT2470 Lab 02: Teh Machine Learnings

by Bob L. T. Sturm

In this lab you will practice some fundamental concepts of machine learning. In the first part you will work with an artificial dataset. The second part involves a music dataset. There's a lot more to explore, so this lab only scratches the surface!

The lab report you submit should be a testament to your intelligence, as well as a reflection of your willingness to be a part of this module. You are free to use whatever software you want, e.g., python, MATLAB, Processing, C++, etc. But I give tips below in python.

- Numpy API
- Pandas API
- Scikit-learn API
- MatPlotlib API
- Numpy Cheat Sheet
- Pydub API

I also include some images so you can confirm whether you are on the right track, or just to have a brief pause to laugh at how far your answer is from being correct.

Names: Sergi Andreu and Carsten van de Kamp

#### Part 1: Basics

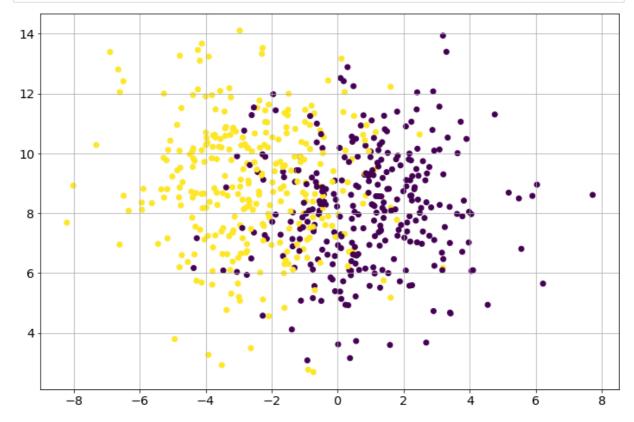
In this part, you will explore some basic principles of machine learning. First we are going to synthesize a "toy" dataset, where we know everything about the data. This is the "best case scenario", and can help us understand ML methods and their parameters, as well as debug implementations.

For more information:

sklearn.datasets

```
In [38]:
```

```
# make a scatter plot of the dataset
fig, ax = plt.subplots()
plt.scatter(X[:,0],X[:,1],c=y)
plt.grid(); plt.show()
```



Create a 80/20 train/test split of your data. See <a href="mailto:sklearn.model\_selection">sklearn.model\_selection</a> for the function train\_test\_split.

```
In [39]:
          from sklearn.model_selection import train_test_split
          # add your code below
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

# 1.2

Create an instance of a nearest neighbor classifier and train it with your training data. See sklearn.neighbors for the appropriate function.

```
In [40]:
             # add your code below
             \textbf{from} \  \, \textbf{sklearn.neighbors} \  \, \textbf{import} \  \, \textbf{KNeighborsClassifier}
             neigh1 = KNeighborsClassifier(n_neighbors=1)
             neigh1.fit(X_train, y_train)
            KNeighborsClassifier(n_neighbors=1)
```

1.3

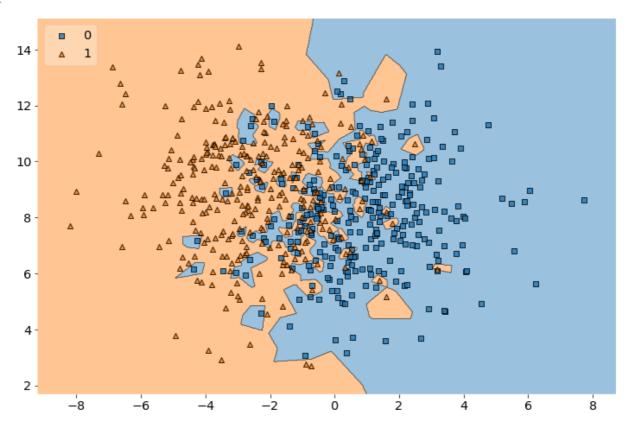
Out[40]:

Plot the decision boundary of your trained classifier. Use the *plot\_decision\_regions* function in mlxtend.plotting.

```
from mlxtend.plotting import plot_decision_regions
    # add your code below

plot_decision_regions(X, y, clf=neigh1, legend=2)
```

#### Out[41]: <AxesSubplot:>



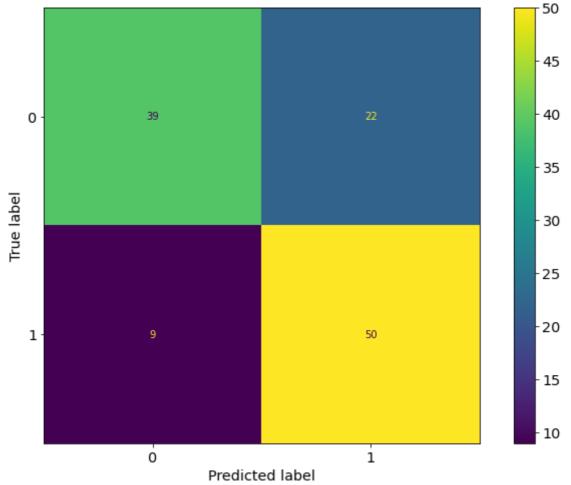
## 1.4

Use your classifier to label the hold out set, and print out the predicted labels and true labels.

Print the confusion matrix of the results. See sklearn.metrics for a relevant function.

```
In [43]: # add your code below
    from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
        conf_matrix = confusion_matrix(y_test, y_pred)
        print(conf_matrix)
        disp = ConfusionMatrixDisplay(conf_matrix, display_labels=['0', '1'])
        disp.plot()

[[39 22]
        [9 50]]
        <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1de829a3f10>
Out[43]:
```



## 1.6

Use the *classification\_report* function of sklearn.metrics to print out more metrics for your classifier.

class 1	0.69	0.85	0.76	59
accuracy			0.74	120
macro avg	0.75	0.74	0.74	120
weighted avg	0.75	0.74	0.74	120

Now you will explore k nearest neighbor classification with the toy dataset. Create a 50/30/20 train/validation/test split of your data.

```
In [47]: # add your code below

X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size=0.2)

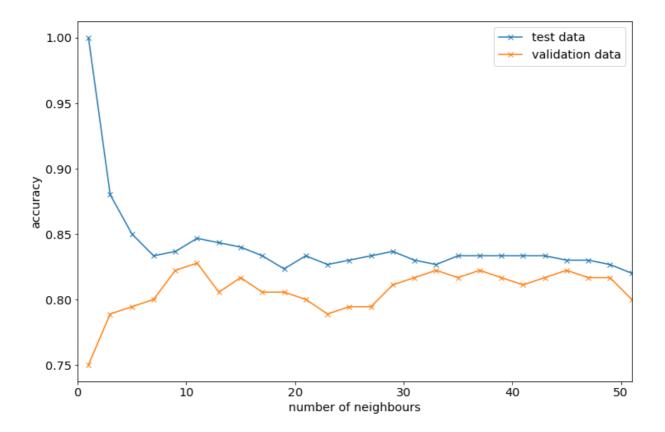
X_train2, X_val2, y_train2, y_val2 = train_test_split(X_train2, y_train2, test_size=
```

### 1.8

For k in 1 to 51 by odd integers, train a kNN classifier and compute its accuracy on the training dataset and the validation dataset. Plot these accuracies as a function of k.

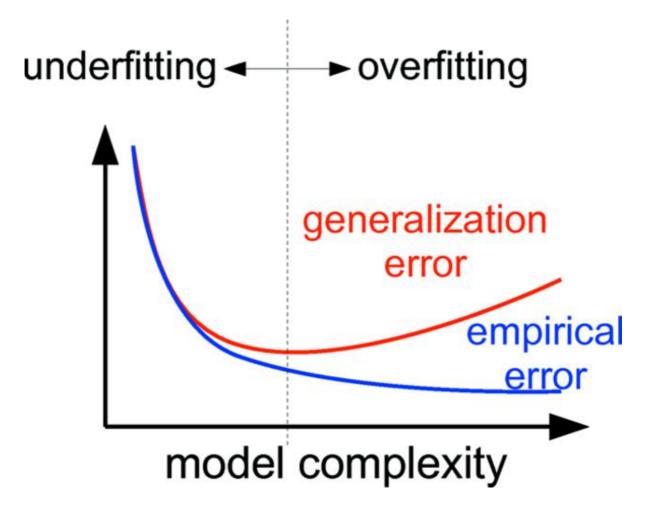
```
In [48]:
           # add your code below
           ks = [(2*i + 1) \text{ for } i \text{ in } range(26)]
           test_acc = []
           val_acc = []
           for k in ks:
                neigh2 = KNeighborsClassifier(n_neighbors=k)
                neigh2.fit(X_train2, y_train2)
                test_acc.append(neigh2.score(X_train2, y_train2))
                val_acc.append(neigh2.score(X_val2, y_val2))
           # Plotting
           fig, ax = plt.subplots()
           ax.plot(ks, test_acc, 'x-', label='test data')
ax.plot(ks, val_acc, 'x-', label='validation data')
           ax.set_xlim((0,51))
           # ax.set_ylim((0.7,1))
           ax.set_xlabel('number of neighbours')
           ax.set_ylabel('accuracy')
           ax.legend()
```

Out[48]: <matplotlib.legend.Legend at 0x1de82d74fd0>



# Generalization, Overfitting, and Underfitting

- If a model is able to make accurate predictions on unseen data:
  - it is able to **generalize** from the training set to the test set.
- Building a model that is too complex for the amount of information in the data is called overfitting.
- This refers to the complexity of the decision boundary, **NOT** computational complexity.



The complexity of KNN models decreases with K. Find the K with the highest validation accuracy, and train a kNN classifier with that K on the training data. Test it on the testing data and print its classification\_report. If there are several possible K, choose one by some strategy.

```
# Determine the best k, which is found where the validation accuracy is maximal.
# If there are several possible k, we choose the largest value because we favour a s
# Small k yields a complex decision boundary, using a small k is prone to overfittin
best_k = np.max(ks[np.argmax(val_acc)])

neigh_best_k = KNeighborsClassifier(n_neighbors=best_k)
neigh_best_k.fit(X_train2, y_train2)

y_pred2 = neigh_best_k.predict(X_test2)
print(f'The best k is {best_k}. \n')
print(classification_report(y_test2, y_pred2, target_names=target_names))
```

The best k is 11.

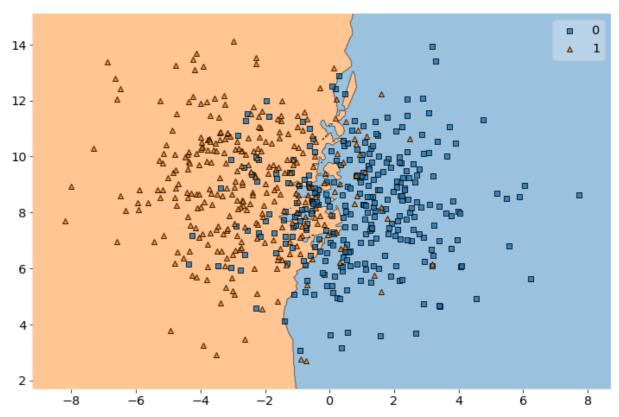
	precision	recall	f1-score	support
class 0 class 1	0.87 0.67	0.61 0.89	0.72 0.76	64 56
accuracy macro avg weighted avg	0.77 0.77	0.75 0.74	0.74 0.74 0.74	120 120 120

For this classifier, plot its decision boundary, and compare with the boundary you see in #1.3.

```
In [50]: plot_decision_regions(X, y, clf=neigh_best_k)
```

Out[50]:

<AxesSubplot:>

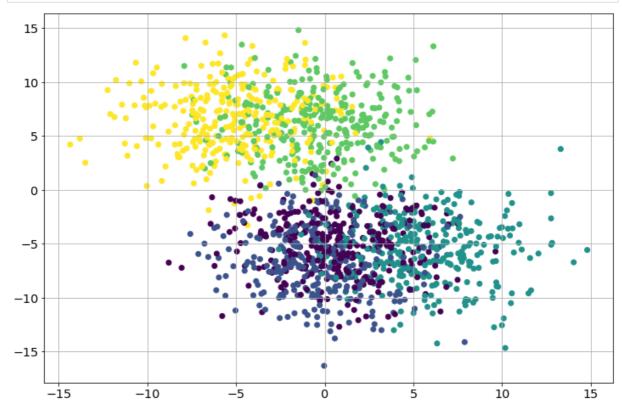


Compared to the decision boundary in assignment 1.3 we see that this one is much smoother. Therefore we can assume that it generalizes better on unseen data. The non-smoothness in the decision boundary is caused by specific properties in the training data that are unlikely to be found in new unseen data.

## 1.11

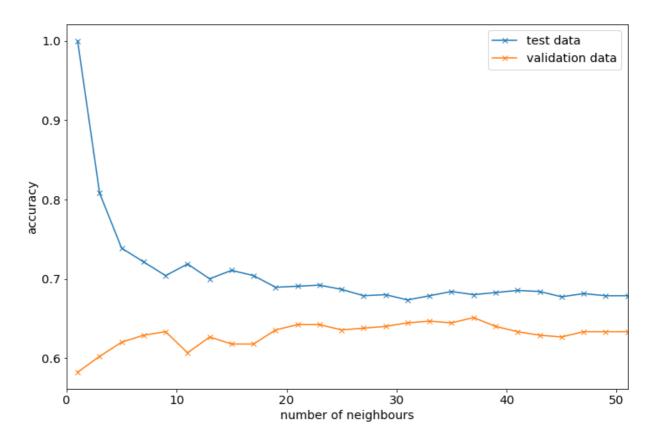
Change the toy dataset (number of classes, number of observations, cluster spread) and repeat the above. What happens as the cluster spread increase? What happens if there are more classes?

```
# make a scatter plot of the dataset
fig, ax = plt.subplots()
plt.scatter(X3[:,0],X3[:,1],c=y3)
plt.grid(); plt.show()
```



```
In [13]: # Split the data in a 50/30/20 train/validation/test set
    X_train3, X_test3, y_train3, y_test3 = train_test_split(X3, y3, test_size=0.2)
    X_train3, X_val3, y_train3, y_val3 = train_test_split(X_train3, y_train3, test_size=
```

```
In [14]:
          # add your code below
          ks = [(2*i + 1) \text{ for } i \text{ in } range(26)]
          test_acc3 = []
          val_acc3 = []
          for k in ks:
               neigh3 = KNeighborsClassifier(n_neighbors=k)
               neigh3.fit(X_train3, y_train3)
               test_acc3.append(neigh3.score(X_train3, y_train3))
               val_acc3.append(neigh3.score(X_val3, y_val3))
          fig, ax = plt.subplots()
          ax.plot(ks, test_acc3, 'x-', label='test data')
          ax.plot(ks, val_acc3, 'x-', label='validation data')
          ax.set_xlim((0,51))
          # ax.set_ylim((0.9,1))
          ax.set_xlabel('number of neighbours')
          ax.set_ylabel('accuracy')
          ax.legend()
```



```
In [15]:
# Determine the best k, which is found where the validation accuracy is maximal.
# If there are several possible k, we choose the largest value because we favour a s
# Small k yields a complex decision boundary, using a small k is prone to overfittin
best_k3 = np.max(ks[np.argmax(val_acc3)])

neigh_best_k3 = KNeighborsClassifier(n_neighbors=best_k3)
neigh_best_k3.fit(X_train3, y_train3)

y_pred3 = neigh_best_k3.predict(X_test3)

target_names3 = [f'class {c}' for c in np.arange(0,numClasses3)]
print(f'The best k is {best_k3}. \n')
print(classification_report(y_test3, y_pred3, target_names=target_names3))
```

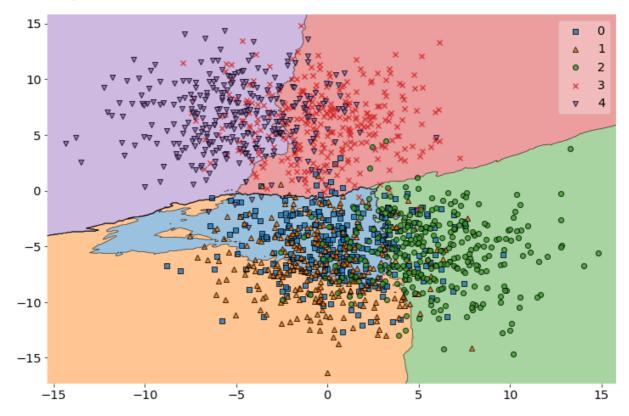
The best k is 37.

	precision	recall	f1-score	support
class 0	0.45	0.57	0.50	58
class 1	0.67	0.49	0.57	63
class 2	0.77	0.78	0.78	69
class 3	0.69	0.75	0.72	53
class 4	0.77	0.72	0.75	57
accuracy			0.66	300
macro avg	0.67	0.66	0.66	300
weighted avg	0.68	0.66	0.66	300

```
In [16]: plot_decision_regions(X3, y3, clf=neigh_best_k3)
```

C:\Users\Carsten\anaconda3\lib\site-packages\mlxtend\plotting\decision\_regions.py:27 4: UserWarning: You passed a edgecolor/edgecolors ('black') for an unfilled marker ('x'). Matplotlib is ignoring the edgecolor in favor of the facecolor. This behavi or may change in the future.

ax.scatter(x=x\_data,



When the cluster spread increases, then the data sets are more likely to overlap and hence the confusion increases. It is harder for the classifier do determine a good decision boundary (because a hard decision boundary might not exist) and the accuracy is lower. We notice that the optimal k is in general higher compared to the case where there is little overlap.

When there are more classes, then again the probability of overlap between the data cloud from different classes increases, again resulting in less accuracy and a higher optimal k.

```
In [ ]:
```

# **Part 2: Application**

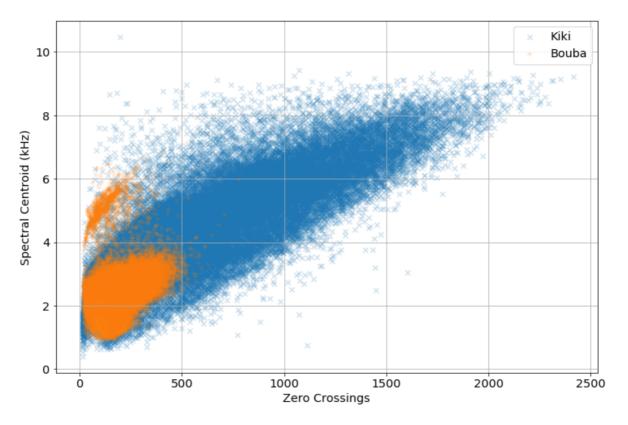
In this portion of the lab, you will work with audio recordings of two music genres: *Kiki* and *Bouba*. These genres are popular amongst the beings that inhabit the digital world. Download the zipfile from the Canvas lab 02 page, unzip, and put them somewhere you can find them.

Below are two functions we will use to extract features from the audio data.

```
def extractZC( data, windowsizeinsamples, windowhopinsamples ):
    totalnumberofwindows = np.floor((len(data)-windowsizeinsamples)/windowhopinsampl
    numbersignchanges = []
    for ii in np.arange(totalnumberofwindows):
        windowstart = np.int(ii*windowhopinsamples)
        windowend = np.int(windowstart + windowhopinsamples)
        datasign = np.sign(data[windowstart:windowend])
        datasignderivative = datasign - np.roll(datasign,1)
        numbersignchanges.append(np.count_nonzero(datasignderivative[1:-1]))
    return np.array(numbersignchanges)

def extractSC( data, windowsizeinsamples, windowhopinsamples, samplerate ):
    totalnumberofwindows = np.floor((len(data)-windowsizeinsamples)/windowhopinsampl
```

Using the functions above, compute zero crossing and spectral centroid features for all kiki and bouba training data files using contiguous 100 ms frames. (Ignore any frames at the end of audio files that are less than that length.) Create a two-dimensional scatter plot of the features extracted from the training data. Below is what my scatter plot looks like, just so you know.



```
def extract_features(path, frame_size_seconds, window_hop_seconds):
    ZC_array = np.array([])
    SC_array = np.array([])

for file in os.scandir(path):
    print("analyzing ", file.name, file.path)

    sound = pydub.AudioSegment.from_file(path + "/" + file.name, format = "m4a")
    sample_rate = sound.frame_rate

    windowsizeinsamples = frame_size_seconds * sample_rate
    windowhopinsamples = window_hop_seconds * sample_rate

    samples = sound.get_array_of_samples()

    ZC = extractZC( samples, windowsizeinsamples, windowhopinsamples, sample_rate
```

```
ZC_array = np.append(ZC_array, ZC)
SC_array = np.append(SC_array, SC)
return ZC_array, SC_array
```

In [19]:

```
# add your code below
import pydub
import os

kiki_train_path = 'KikiBoubadata/kikibouba_train/kiki'
bouba_train_path = 'KikiBoubadata/kikibouba_train/bouba'

frame_size_seconds = 0.100
window_hop_seconds = 0.100

# Kiki
kiki_train_ZC, kiki_train_SC = extract_features(kiki_train_path, frame_size_seconds,
# Bouba
bouba_train_ZC, bouba_train_SC = extract_features(bouba_train_path, frame_size_seconds)

# Converting Hz to kHz
kiki_train_SC = kiki_train_SC/1000
bouba_train_SC = bouba_train_SC/1000
```

analyzing kiki\_0001.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0001.m4a

C:\Users\Carsten\AppData\Local\Temp/ipykernel\_18876/3852559072.py:5: DeprecationWarn ing: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When repla cing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the prec ision. If you wish to review your current use, check the release note link for addit ional information.

Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

windowstart = np.int(ii\*windowhopinsamples)

C:\Users\Carsten\AppData\Local\Temp/ipykernel\_18876/3852559072.py:6: DeprecationWarn ing: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When repla cing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.

Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

windowend = np.int(windowstart + windowhopinsamples)

C:\Users\Carsten\AppData\Local\Temp/ipykernel\_18876/3852559072.py:17: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.

Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

windowstart = np.int(ii\*windowhopinsamples)

C:\Users\Carsten\AppData\Local\Temp/ipykernel\_18876/3852559072.py:18: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.

Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/r

elease/1.20.0-notes.html#deprecations windowend = np.int(windowstart + windowsizeinsamples) kiki 0002.m4a KikiBoubadata/kikibouba train/kiki\kiki 0002.m4a analyzing kiki\_0003.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0003.m4a analyzing analyzing

kiki\_0004.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0004.m4a kiki 0005.m4a KikiBoubadata/kikibouba train/kiki\kiki 0005.m4a kiki\_0006.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0006.m4a kiki\_0007.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0007.m4a kiki\_0008.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0008.m4a kiki 0009.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0009.m4a kiki\_0010.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0010.m4a kiki\_0011.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0011.m4a kiki\_0012.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0012.m4a kiki\_0013.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0013.m4a analyzing kiki\_0014.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0014.m4a analyzing kiki\_0015.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0015.m4a kiki 0016.m4a KikiBoubadata/kikibouba train/kiki\kiki 0016.m4a analyzing kiki\_0017.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0017.m4a kiki\_0018.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0018.m4a analyzing analyzing kiki\_0019.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0019.m4a analyzing kiki\_0020.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0020.m4a analyzing kiki\_0021.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0021.m4a analyzing kiki\_0022.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0022.m4a analyzing kiki\_0023.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0023.m4a analyzing kiki\_0024.m4a KikiBoubadata/kikibouba\_train/kiki\kiki 0024.m4a kiki\_0025.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0025.m4a analyzing kiki\_0026.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0026.m4a analyzing kiki 0027.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0027.m4a analyzing kiki\_0028.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0028.m4a analyzing analyzing kiki\_0029.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0029.m4a analyzing kiki\_0030.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0030.m4a analyzing kiki\_0031.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0031.m4a analyzing kiki\_0032.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0032.m4a kiki\_0033.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0033.m4a analyzing kiki\_0034.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0034.m4a analyzing analyzing kiki\_0035.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0035.m4a analyzing kiki\_0036.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0036.m4a analyzing kiki\_0037.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0037.m4a kiki 0038.m4a KikiBoubadata/kikibouba train/kiki\kiki 0038.m4a analyzing kiki\_0039.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0039.m4a analyzing kiki\_0040.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0040.m4a kiki 0041.m4a KikiBoubadata/kikibouba train/kiki\kiki 0041.m4a kiki\_0042.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0042.m4a kiki\_0043.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0043.m4a kiki\_0044.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0044.m4a kiki 0045.m4a KikiBoubadata/kikibouba train/kiki\kiki 0045.m4a kiki\_0046.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0046.m4a kiki\_0047.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0047.m4a kiki 0048.m4a KikiBoubadata/kikibouba train/kiki\kiki 0048.m4a kiki\_0049.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0049.m4a kiki\_0050.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0050.m4a kiki\_0051.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0051.m4a kiki\_0052.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0052.m4a kiki\_0053.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0053.m4a kiki\_0054.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0054.m4a kiki 0055.m4a KikiBoubadata/kikibouba train/kiki\kiki 0055.m4a kiki\_0056.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0056.m4a kiki\_0057.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0057.m4a kiki\_0058.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0058.m4a kiki 0059.m4a KikiBoubadata/kikibouba train/kiki\kiki 0059.m4a kiki\_0060.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0060.m4a kiki\_0061.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0061.m4a kiki\_0062.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0062.m4a

analyzing kiki\_0063.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0063.m4a kiki\_0064.m4a KikiBoubadata/kikibouba\_train/kiki\kiki\_0064.m4a analyzing

```
kiki_0065.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0065.m4a
analyzing
analyzing
          kiki_0066.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0066.m4a
analyzing
          kiki_0067.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0067.m4a
analyzing
          kiki_0068.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0068.m4a
          kiki_0069.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0069.m4a
analyzing
          kiki 0070.m4a KikiBoubadata/kikibouba train/kiki\kiki 0070.m4a
analyzing
          kiki_0071.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0071.m4a
analyzing
          kiki_0072.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0072.m4a
analyzing
analyzing
          kiki_0073.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0073.m4a
analyzing
          kiki_0074.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0074.m4a
          kiki_0075.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0075.m4a
analyzing
          kiki_0076.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0076.m4a
analyzing
analyzing
          kiki_0077.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0077.m4a
analyzing
          kiki_0078.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0078.m4a
          kiki_0079.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0079.m4a
analyzing
analyzing
          kiki_0080.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0080.m4a
          kiki 0081.m4a KikiBoubadata/kikibouba train/kiki\kiki 0081.m4a
analyzing
analyzing
          kiki_0082.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0082.m4a
          kiki_0083.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0083.m4a
analyzing
analyzing
          kiki_0084.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0084.m4a
analyzing
          kiki_0085.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0085.m4a
analyzing
          kiki_0086.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0086.m4a
analyzing
          kiki_0087.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0087.m4a
          kiki_0088.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0088.m4a
analyzing
          kiki_0089.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0089.m4a
analyzing
          kiki_0090.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0090.m4a
analyzing
analyzing
          kiki_0091.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0091.m4a
analyzing
          kiki_0092.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0092.m4a
          kiki_0093.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0093.m4a
analyzing
analyzing
          kiki_0094.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0094.m4a
analyzing
          kiki_0095.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0095.m4a
analyzing
          kiki_0096.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0096.m4a
analyzing
          kiki_0097.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0097.m4a
          kiki_0098.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0098.m4a
analyzing
          kiki_0099.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0099.m4a
analyzing
          kiki_0100.m4a KikiBoubadata/kikibouba_train/kiki\kiki_0100.m4a
analyzing
analyzing
          bouba_0001.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0001.m4a
analyzing
          bouba_0002.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0002.m4a
analyzing
          bouba_0003.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0003.m4a
          bouba_0004.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0004.m4a
analyzing
          bouba_0005.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0005.m4a
analyzing
analyzing
          bouba 0006.m4a KikiBoubadata/kikibouba train/bouba\bouba 0006.m4a
analyzing
          bouba_0007.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0007.m4a
          bouba_0008.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0008.m4a
analyzing
analyzing
          bouba_0009.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0009.m4a
analyzing
          bouba 0010.m4a KikiBoubadata/kikibouba train/bouba\bouba 0010.m4a
analyzing
          bouba_0011.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0011.m4a
          bouba_0012.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0012.m4a
analyzing
          bouba 0013.m4a KikiBoubadata/kikibouba train/bouba\bouba 0013.m4a
analyzing
          bouba_0014.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0014.m4a
analyzing
analyzing
          bouba_0015.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0015.m4a
analyzing
          bouba_0016.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0016.m4a
analyzing
          bouba_0017.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0017.m4a
          bouba_0018.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0018.m4a
analyzing
          bouba_0019.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0019.m4a
analyzing
analyzing
          bouba 0020.m4a KikiBoubadata/kikibouba train/bouba\bouba 0020.m4a
analyzing
          bouba 0021.m4a KikiBoubadata/kikibouba train/bouba\bouba 0021.m4a
          bouba_0022.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0022.m4a
analyzing
analyzing
          bouba_0023.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0023.m4a
          bouba 0024.m4a KikiBoubadata/kikibouba train/bouba\bouba 0024.m4a
analyzing
          bouba_0025.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0025.m4a
analyzing
          bouba_0026.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0026.m4a
analyzing
analyzing
          bouba_0027.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0027.m4a
analyzing
          bouba_0028.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0028.m4a
          bouba_0029.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0029.m4a
analyzing
```

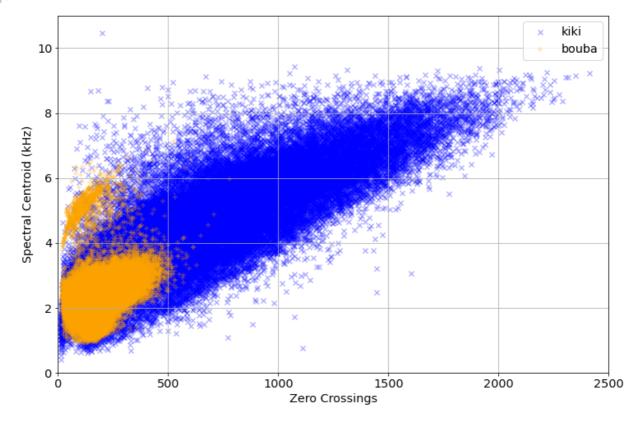
```
bouba_0030.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0030.m4a
analyzing
analyzing
           bouba_0031.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0031.m4a
analyzing
           bouba_0032.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0032.m4a
analyzing
          bouba_0033.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0033.m4a
          bouba_0034.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0034.m4a
analyzing
          bouba 0035.m4a KikiBoubadata/kikibouba train/bouba\bouba 0035.m4a
analyzing
          bouba_0036.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0036.m4a
analyzing
          bouba_0037.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0037.m4a
analyzing
analyzing
          bouba_0038.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0038.m4a
analyzing
          bouba_0039.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0039.m4a
          bouba_0040.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0040.m4a
analyzing
analyzing
          bouba_0041.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0041.m4a
analyzing
          bouba_0042.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0042.m4a
analyzing
          bouba_0043.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0043.m4a
          bouba_0044.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0044.m4a
analyzing
analyzing
          bouba_0045.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0045.m4a
          bouba 0046.m4a KikiBoubadata/kikibouba train/bouba\bouba 0046.m4a
analyzing
analyzing
          bouba_0047.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0047.m4a
          bouba_0048.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0048.m4a
analyzing
analyzing
          bouba_0049.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0049.m4a
analyzing
          bouba_0050.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0050.m4a
analyzing
          bouba_0051.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0051.m4a
analyzing
          bouba_0052.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0052.m4a
          bouba_0053.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0053.m4a
analyzing
analyzing
          bouba_0054.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0054.m4a
          bouba_0055.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0055.m4a
analyzing
analyzing
          bouba_0056.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0056.m4a
          bouba 0057.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0057.m4a
analyzing
          bouba_0058.m4a KikiBoubadata/kikibouba_train/bouba_0058.m4a
analyzing
analyzing
          bouba_0059.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0059.m4a
analyzing
          bouba_0060.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0060.m4a
analyzing
          bouba_0061.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0061.m4a
analyzing
          bouba_0062.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0062.m4a
analyzing
          bouba_0063.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0063.m4a
          bouba_0064.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0064.m4a
analyzing
          bouba_0065.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0065.m4a
analyzing
analyzing
          bouba_0066.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0066.m4a
analyzing
          bouba_0067.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0067.m4a
analyzing
          bouba_0068.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0068.m4a
          bouba_0069.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0069.m4a
analyzing
          bouba_0070.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0070.m4a
analyzing
analyzing
          bouba 0071.m4a KikiBoubadata/kikibouba train/bouba\bouba 0071.m4a
analyzing
          bouba_0072.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0072.m4a
          bouba_0073.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0073.m4a
analyzing
analyzing
          bouba_0074.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0074.m4a
analyzing
          bouba 0075.m4a KikiBoubadata/kikibouba train/bouba\bouba 0075.m4a
analyzing
          bouba_0076.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0076.m4a
analyzing
          bouba_0077.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0077.m4a
          bouba 0078.m4a KikiBoubadata/kikibouba train/bouba\bouba 0078.m4a
analyzing
          bouba_0079.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0079.m4a
analyzing
analyzing
          bouba_0080.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0080.m4a
analyzing
          bouba_0081.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0081.m4a
analyzing
          bouba_0082.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0082.m4a
          bouba_0083.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0083.m4a
analyzing
          bouba_0084.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0084.m4a
analyzing
analyzing
          bouba 0085.m4a KikiBoubadata/kikibouba train/bouba\bouba 0085.m4a
analyzing
          bouba 0086.m4a KikiBoubadata/kikibouba train/bouba\bouba 0086.m4a
          bouba_0087.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0087.m4a
analyzing
analyzing
          bouba_0088.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0088.m4a
          bouba 0089.m4a KikiBoubadata/kikibouba train/bouba\bouba 0089.m4a
analyzing
          bouba_0090.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0090.m4a
analyzing
          bouba_0091.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0091.m4a
analyzing
analyzing
           bouba_0092.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0092.m4a
analyzing
          bouba_0093.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0093.m4a
analyzing
          bouba_0094.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0094.m4a
```

```
analyzing bouba_0095.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0095.m4a bouba_0096.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0096.m4a bouba_0097.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0097.m4a bouba_0098.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0098.m4a bouba_0099.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0099.m4a bouba_0100.m4a KikiBoubadata/kikibouba_train/bouba\bouba_0100.m4a
```

```
In [20]:
    """
    Plotting
    """

    fig, ax = plt.subplots()
    ax.scatter(kiki_train_ZC, kiki_train_SC, c='blue', marker='x', alpha=0.3, label='ki
    ax.scatter(bouba_train_ZC, bouba_train_SC, c='orange', marker ='+', alpha=0.3, label
    ax.legend()
    ax.set_xlabel('Zero Crossings')
    ax.set_ylabel('Spectral Centroid (kHz)')
    ax.grid()
    ax.set_xlim((0,2500))
    ax.set_ylim((0,11))
```

Out[20]: (0.0, 11.0)



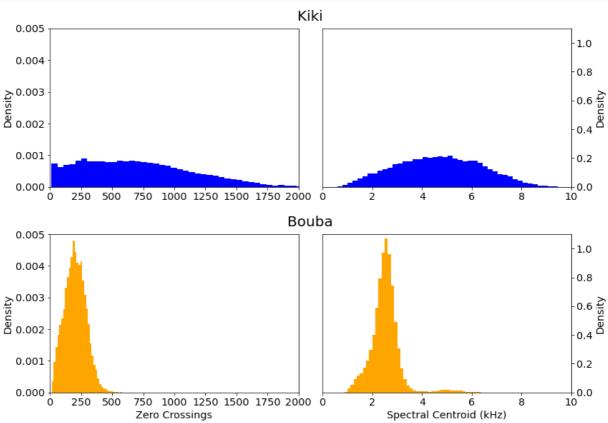
## 2.2

Create and plot histograms of the zero crossings feature and the spectral centroid feature for the class kiki. Use 50 bins. Do the same for bouba. Normalize each histogram by setting the parameter "density=True" in either pandas.DataFrame.hist or matplotlib.pyplot.hist.

```
In [21]: # add your code below

fig, [[ax1, ax2], [ax3, ax4]] = plt.subplots(2,2)
ax1.hist(kiki_train_ZC, bins=50, density=True, color='blue')
ax2.hist(kiki_train_SC, bins=50, density=True, color='blue')
ax3.hist(bouba_train_ZC, bins=50, density=True, color='orange')
```

```
ax4.hist(bouba_train_SC, bins=50, density=True, color='orange')
ax1.set_xlim((0,2000))
ax3.set_xlim((0,2000))
ax1.set_ylim((0,.005))
ax3.set_ylim((0,.005))
ax2.set_xlim((0,10))
ax4.set_xlim((0,10))
ax2.set_ylim((0,1.1))
ax4.set_ylim((0,1.1))
ax2.yaxis.tick_right()
ax4.yaxis.tick_right()
ax3.set_xlabel('Zero Crossings')
ax4.set_xlabel('Spectral Centroid (kHz)')
ax1.set_ylabel('Density')
ax2.set_ylabel('Density')
ax2.yaxis.set_label_position("right")
ax3.set_ylabel('Density')
ax4.set_ylabel('Density')
ax4.yaxis.set_label_position("right")
plt.figtext(0.51, 1, "Kiki", ha='center', va='center', fontsize=20)
plt.figtext(0.51, 0.5, "Bouba", ha='center', va='center', fontsize=20)
fig.tight_layout()
plt.subplots_adjust(hspace=.3) # some extra space for the subtitles
```



Let's model *each* feature of the *each* class by a Gausssian distribution. This means we want to approximate each histogram of each feature observed in class g by a Gaussian function with mean  $\mu_{*,g}$  and variance  $\sigma_{*,g}^2$ :

$$f_{X|G}(x|g) = rac{1}{\sqrt{2\pi\sigma_{*,g}^2}} {
m exp}igl[ -(x-\mu_{*,g})^2/2\sigma_{*,g}^2igr]$$

Use unbiased maximum likelihood estimation (MLE) to compute the parameters of the two distributions using your training observations. Remember, the MLE mean of N numbers  $\{x_1, x_2, \ldots, x_N\}$  is given by

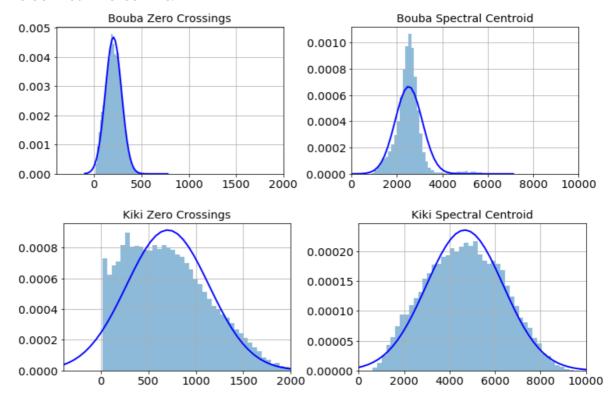
$$\hat{\mu} = rac{1}{N} \sum_{n=1}^N x_n$$

and the MLE variance is given by

$$\sigma^2 = rac{1}{N-1} \sum_{n=1}^N (x_n - \hat{\mu})^2.$$

Plot these four distributions, overlayed on the normalized histograms of the features for the two classes.

Here's what mine look like:



```
from scipy.stats import norm

# add your code below
kiki_train_ZC_mu = np.sum(kiki_train_ZC) / len(kiki_train_ZC_mu)**2 ) / (len(kiki_train_XC_mu)**2 )
```

```
bouba_train_ZC_mu = np.sum(bouba_train_ZC) / len(bouba_train_ZC)
bouba_train_ZC_var = np.sum( (bouba_train_ZC - bouba_train_ZC_mu)**2 ) / (len(bouba_bouba_train_ZC_gaussfit = lambda x: norm.pdf(x, bouba_train_ZC_mu, bouba_train_ZC_va
bouba_train_SC_mu = np.sum(bouba_train_SC) / len(bouba_train_SC)
bouba_train_SC_var = np.sum( (bouba_train_SC - bouba_train_SC_mu)**2 ) / (len(bouba_bouba_train_SC_gaussfit = lambda x: norm.pdf(x, bouba_train_SC_mu, bouba_train_SC_va
```

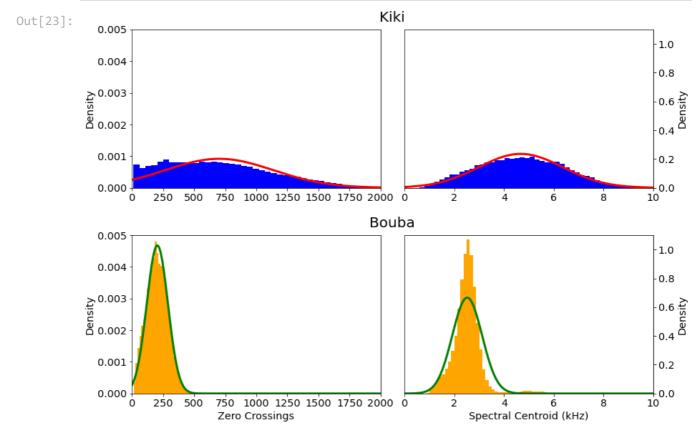
```
In [23]: # Plotting

x_ZC = np.linspace(0,2500,num=1000)
x_SC = np.linspace(0,10, num=1000)

ax1.plot(x_ZC, kiki_train_ZC_gaussfit(x_ZC), 'r-', linewidth=3)
ax2.plot(x_SC, kiki_train_SC_gaussfit(x_SC), 'r-', linewidth=3)

ax3.plot(x_ZC, bouba_train_ZC_gaussfit(x_ZC), 'g-', linewidth=3)
ax4.plot(x_SC, bouba_train_SC_gaussfit(x_SC), 'g-', linewidth=3)

fig
```



Given a particular two-dimensional observation  $\mathbf x$  extracted from an audio file, we want to estimate the probability of  $\mathbf x$  coming from class  $g \in \{\text{Kiki}, \text{Bouba}\}$ , i.e., the *posterior*  $P[g|\mathbf x]$ . By Bayes' Theorem, we can express this probability as:

$$P[G = g | \mathbf{x}] = rac{f_{\mathbf{X}|G}(\mathbf{x}|G = g)P[G = g]}{f_{\mathbf{X}}(\mathbf{x})}$$

where the *likelihood*  $f_{\mathbf{X}|G}(\mathbf{x}|G=g)$  is the probability density of observing  $\mathbf{x}$  from class g, P[G=g] is the *prior* probability of encountering g regardless of what we observe, and  $f_{\mathbf{X}}(\mathbf{x})$  is the probability density of observing  $\mathbf{x}$  no matter what class.

We define the *posterior odds* of Kiki as the ratio:

$$O(G = \text{Kiki}|\mathbf{x}) = \frac{P[G = \text{Kiki}|\mathbf{x}]}{P[G = \text{Bouba}|\mathbf{x}]}$$

Similarly, the posterior odds of Bouba is the ratio

$$O(G = \mathrm{Bouba}|\mathbf{x}) = rac{P[G = \mathrm{Bouba}|\mathbf{x}]}{P[G = \mathrm{Kiki}|\mathbf{x}]} = rac{1}{O(G = \mathrm{Kiki}|\mathbf{x})}.$$

These are easy to understand. For instance, if  $O(G = \mathrm{Kiki}|\mathbf{x}) = 2$ , then according to our model Kiki is twice as likely given  $\mathbf{x}$  than Bouba, or Bouba is half as likely as Kiki. By plugging in the equation for  $P[G = g|\mathbf{x}]$ , the odds of Kiki can be expressed

$$O(G = \text{Kiki}|\mathbf{x}) = rac{f_{\mathbf{X}|G}(\mathbf{x}|G = \text{Kiki})}{f_{\mathbf{X}|G}(\mathbf{x}|G = \text{Bouba})} rac{P[G = \text{Kiki}]}{P[G = \text{Bouba}]}$$

So the posterior odds of Kiki is just the product of the ratio of posterior probability densities and prior probabilities.

If for an observation  $O(G = \mathrm{Kiki}|\mathbf{x}) > 2$ , then we might label it as Kiki. Otherwise we could label it Bouba. We just need to compute the densities  $f_{\mathbf{X}|G}(\mathbf{x}|G = g)$ .

This is where our Gaussian models come into play. We know the first element of  $\mathbf{x}$  is the zero crossing feature, and the second element is spectral centroid. If we assume these two features are independent, then:

$$f_{\mathbf{X}|G}(x_z,x_s|G=g) = f_{X_z|G}(x_z|G=g)f_{X_s|G}(x_s|G=g)$$

where  $x_z$  is the zero crossing feature, and  $x_s$  is the spectral centroid features. If we assume each feature is distributed Gaussian, then

$$f_{X_*|G}(x_*|G=g) = rac{1}{\sqrt{2\pi\sigma_{*,g}^2}} \mathrm{exp}igl[-(x_*-\mu_{*,g})^2/2\sigma_{*,g}^2igr].$$

Given our estimates of these four parameters  $\{\mu_{z,g}, \sigma_{z,g}^2, \mu_{s,g}, \sigma_{s,g}^2\}$  for Kiki and for Bouba, we are *almost* ready to build a classification system.

We need to estimate the prior probabilities, P[G = Kiki] and P[G = Bouba].

To do that, assume the training data reflects how observations are distributed in the real world, and estimate  $P[G=\mathrm{Kiki}]$  as the proportion of observations of Kiki in the training dataset, and  $P[G=\mathrm{Bouba}]$  as the proportion of observations of Bouba in the training dataset.

```
In [24]: # add your code below

kiki_train_prior = len(kiki_train_ZC) / (len(kiki_train_ZC) + len(bouba_train_ZC))
bouba_train_prior = len(bouba_train_ZC) / (len(kiki_train_ZC) + len(bouba_train_ZC))
assert kiki_train_prior + bouba_train_prior == 1
```

Now write a function using all the parameters you have estimated that computes the posterior odds of Kiki given a two-dimensional observation, and returns 1 (Kiki) if the odds are greater than 2, and otherwise 0 (Bouba). Use the MLE parameters you found in the previous part.

```
In [25]:
          # add your code below
          def kiki_posterior_odds_greater2(x,
                                            kiki_train_prior = kiki_train_prior,
                                           bouba_train_prior = bouba_train_prior,
                                           kiki_train_ZC_gaussfit = kiki_train_ZC_gaussfit,
                                           kiki_train_SC_gaussfit = kiki_train_SC_gaussfit,
                                           bouba_train_ZC_gaussfit = bouba_train_ZC_gaussfit,
                                           bouba_train_SC_gaussfit = bouba_train_SC_gaussfit):
              Compute posterior odds of kiki given a 2D observation x = [num_zero_crossings, s
              if odds > 2, returns 1 (=Kiki)
              else returns 0 (=Bouba)
              prior_ratio = kiki_train_prior / bouba_train_prior
              ZC = x[0]
              SC = x[1]
              kiki likelihood = kiki train ZC gaussfit(ZC) * kiki train SC gaussfit(SC)
              bouba_likelihood = bouba_train_ZC_gaussfit(ZC) * bouba_train_SC_gaussfit(SC)
              kiki_posterior_odds = kiki_likelihood / bouba_likelihood * prior_ratio
              return True if (kiki_posterior_odds > 2) else False
```

## 2.6

Use your function to label all observations in the training data and create one scatter plot of the labeled Kiki observations, and one scatter plot of the labeled Bouba observations, both color-coded by the predicted labels. (This should take some time to compute because the classifier you have created is not very optimized.)

```
you have created is not very optimized.)

In [26]: 
# add your code below

kiki_training_data = np.stack((kiki_train_ZC, kiki_train_SC), axis=1)
bouba_training_data = np.stack((bouba_train_ZC, bouba_train_SC), axis=1)
training_data = np.concatenate((kiki_training_data, bouba_training_data))

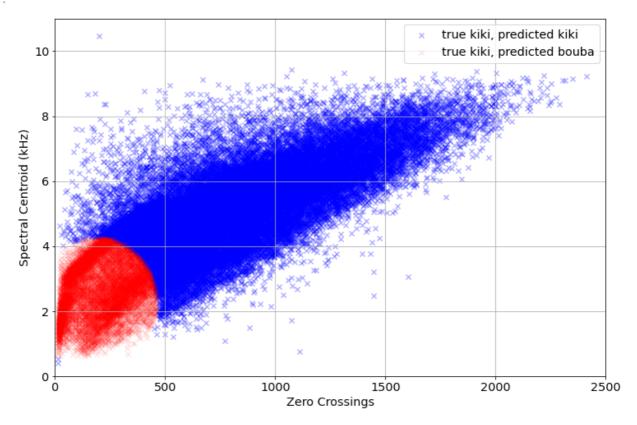
In [27]: 
# Predicting Labels using classifier

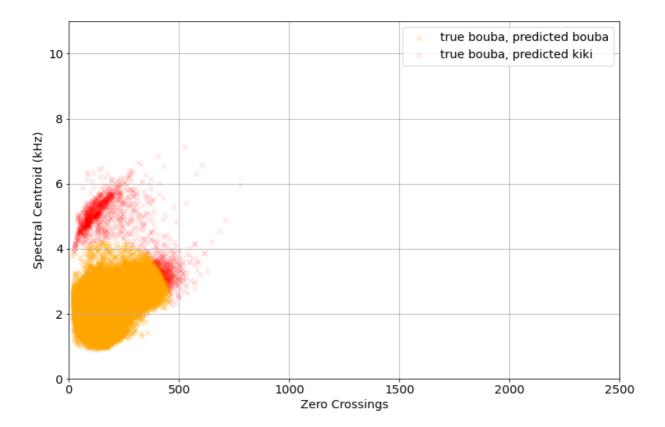
kiki_true_labels = np.ones( len(kiki_train_ZC), dtype=bool)
bouba_true_labels = np.zeros(len(bouba_train_ZC), dtype=bool)
true_labels = np.concatenate((kiki_true_labels, bouba_true_labels))
```

kiki\_predicted\_labels = np.array([kiki\_posterior\_odds\_greater2(sample) for sample

```
In [28]:
          # Plotting
          fig1, ax1 = plt.subplots()
          ax1.scatter(kiki_training_data[kiki_predicted_labels, 0],
                      kiki_training_data[kiki_predicted_labels, 1],
                      c='blue', marker='x', alpha=0.3, label='true kiki, predicted kiki')
          ax1.scatter(kiki_training_data[~kiki_predicted_labels, 0],
                      kiki_training_data[~kiki_predicted_labels, 1],
                      c='red', marker='x', alpha=0.1, label='true kiki, predicted bouba')
          ax1.legend()
          ax1.set_xlabel('Zero Crossings')
          ax1.set_ylabel('Spectral Centroid (kHz)')
          ax1.grid()
          ax1.set_xlim((0,2500))
          ax1.set_ylim((0,11))
          fig2, ax2 = plt.subplots()
          ax2.scatter(bouba_training_data[~bouba_predicted_labels, 0],
                      bouba_training_data[~bouba_predicted_labels, 1],
                      c='orange', marker='x', alpha=0.3, label='true bouba, predicted bouba')
          ax2.scatter(bouba_training_data[bouba_predicted_labels, 0],
                      bouba_training_data[bouba_predicted_labels, 1],
                      c='red', marker='x', alpha=0.1, label='true bouba, predicted kiki')
          ax2.legend()
          ax2.set_xlabel('Zero Crossings')
          ax2.set_ylabel('Spectral Centroid (kHz)')
          ax2.grid()
          ax2.set_xlim((0,2500))
          ax2.set_ylim((0,11))
```

#### Out[28]: (0.0, 11.0)





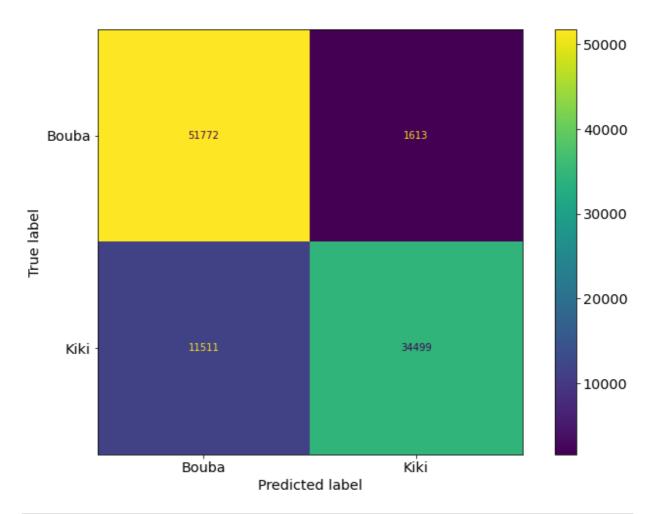
Explain what you see, and how it compares to the scatter plot you created in #2.1.

The classifier came up with a clear and hard decision boundary to predict whether a sample belongs to either the Kiki class or the Bouba class. In the scatter plot in 2.1 (true labels) we see that there are some Kikia samples that have the same properties as typical Kiki samples. Our classifier cannot deal with these.

# 2.8

Compute a confusion table summarizing the results of your classifier on the training dataset.

Compute the accuracy and balanced accuracy. Compute the precision and recall for both classes.



TRAINING DATA Accuracy: 0.87,

Balanced accuracy: 0.86,

Precision: 0.96, Recall: 0.75

### 2.9

Use your function to label all observations in the testing dataset. Compute a confusion table summarizing the results of your classifier on the testing dataset. Compute the accuracy and balanced accuracy. Compute the precision and recall for both classes. Compare with the results on the training dataset.

```
In [31]: # Import test data features

bouba_test_path = 'KikiBoubadata/kikibouba_test/bouba'
kiki_test_path = 'KikiBoubadata/kikibouba_test/kiki'
```

```
frame_size_seconds = 0.100
window_hop_seconds = 0.100

# Kiki
kiki_test_ZC, kiki_test_SC = extract_features(kiki_test_path, frame_size_seconds, wi
# Bouba
bouba_test_ZC, bouba_test_SC = extract_features(bouba_test_path, frame_size_seconds,
# Converting Hz to kHz
kiki_test_SC = kiki_test_SC/1000
bouba_test_SC = bouba_test_SC/1000
```

analyzing kiki\_0101.m4a KikiBoubadata/kikibouba\_test/kiki\kiki\_0101.m4a

C:\Users\Carsten\AppData\Local\Temp/ipykernel\_18876/3852559072.py:5: DeprecationWarn ing: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.

Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

windowstart = np.int(ii\*windowhopinsamples)

C:\Users\Carsten\AppData\Local\Temp/ipykernel\_18876/3852559072.py:6: DeprecationWarn ing: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When repla cing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.

Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

windowend = np.int(windowstart + windowhopinsamples)

C:\Users\Carsten\AppData\Local\Temp/ipykernel\_18876/3852559072.py:17: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.

Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

windowstart = np.int(ii\*windowhopinsamples)

C:\Users\Carsten\AppData\Local\Temp/ipykernel\_18876/3852559072.py:18: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.

Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

windowend = np.int(windowstart + windowsizeinsamples)

```
analyzing kiki_0102.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0102.m4a analyzing kiki_0103.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0103.m4a kiki_0104.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0104.m4a analyzing kiki_0105.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0105.m4a kiki_0106.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0106.m4a analyzing kiki_0107.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0107.m4a analyzing kiki_0108.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0108.m4a analyzing kiki_0109.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0109.m4a analyzing kiki_0110.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0110.m4a analyzing kiki_0111.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0111.m4a analyzing kiki_0112.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0112.m4a kiki_0113.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0113.m4a analyzing kiki_0114.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0113.m4a kikiBoubadata/kikibouba_test/kiki\kiki_0113.m4a kikiBoubadata/kikibouba_test/kiki\kiki_0114.m4a
```

```
kiki_0115.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0115.m4a
analyzing
analyzing
          kiki_0116.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0116.m4a
          kiki_0117.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0117.m4a
analyzing
analyzing
          kiki_0118.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0118.m4a
          kiki_0119.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0119.m4a
analyzing
          kiki_0120.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0120.m4a
analyzing
          kiki_0121.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0121.m4a
analyzing
          kiki_0122.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0122.m4a
analyzing
analyzing
          kiki_0123.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0123.m4a
analyzing
          kiki_0124.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0124.m4a
          kiki_0125.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0125.m4a
analyzing
analyzing
          kiki_0126.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0126.m4a
analyzing
          kiki_0127.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0127.m4a
analyzing
          kiki_0128.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0128.m4a
          kiki_0129.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0129.m4a
analyzing
analyzing
          kiki_0130.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0130.m4a
          kiki 0131.m4a KikiBoubadata/kikibouba test/kiki\kiki 0131.m4a
analyzing
analyzing
          kiki_0132.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0132.m4a
          kiki_0133.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0133.m4a
analyzing
analyzing
          kiki_0134.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0134.m4a
analyzing
          kiki_0135.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0135.m4a
analyzing kiki_0136.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0136.m4a
analyzing kiki_0137.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0137.m4a
          kiki_0138.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0138.m4a
analyzing
          kiki_0139.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0139.m4a
analyzing
          kiki_0140.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0140.m4a
analyzing
analyzing
          kiki_0141.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0141.m4a
          kiki_0142.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0142.m4a
analyzing
          kiki_0143.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0143.m4a
analyzing
analyzing
          kiki_0144.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0144.m4a
analyzing
          kiki_0145.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0145.m4a
analyzing
          kiki_0146.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0146.m4a
analyzing
          kiki_0147.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0147.m4a
          kiki_0148.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0148.m4a
analyzing
          kiki_0149.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0149.m4a
analyzing
          kiki_0150.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0150.m4a
analyzing
          kiki_0151.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0151.m4a
analyzing
analyzing
          kiki_0152.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0152.m4a
analyzing
          kiki_0153.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0153.m4a
          kiki_0154.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0154.m4a
analyzing
          kiki_0155.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0155.m4a
analyzing
          kiki_0156.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0156.m4a
analyzing
analyzing
          kiki_0157.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0157.m4a
          kiki_0158.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0158.m4a
analyzing
analyzing
          kiki_0159.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0159.m4a
analyzing
          kiki 0160.m4a KikiBoubadata/kikibouba test/kiki\kiki 0160.m4a
analyzing
          kiki_0161.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0161.m4a
          kiki_0162.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0162.m4a
analyzing
          kiki_0163.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0163.m4a
analyzing
          kiki_0164.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0164.m4a
analyzing
analyzing
          kiki_0165.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0165.m4a
analyzing
          kiki_0166.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0166.m4a
analyzing
          kiki_0167.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0167.m4a
          kiki_0168.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0168.m4a
analyzing
          kiki_0169.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0169.m4a
analyzing
analyzing
          kiki 0170.m4a KikiBoubadata/kikibouba test/kiki\kiki 0170.m4a
analyzing
          kiki_0171.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0171.m4a
          kiki_0172.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0172.m4a
analyzing
analyzing
          kiki_0173.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0173.m4a
          kiki 0174.m4a KikiBoubadata/kikibouba test/kiki\kiki 0174.m4a
analyzing
          kiki_0175.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0175.m4a
analyzing
          kiki_0176.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0176.m4a
analyzing
analyzing
          kiki_0177.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0177.m4a
analyzing
          kiki_0178.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0178.m4a
analyzing kiki_0179.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0179.m4a
```

```
kiki_0180.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0180.m4a
analyzing
analyzing
          kiki_0181.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0181.m4a
analyzing
          kiki_0182.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0182.m4a
analyzing
          kiki_0183.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0183.m4a
          kiki_0184.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0184.m4a
analyzing
          kiki 0185.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0185.m4a
analyzing
          kiki_0186.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0186.m4a
analyzing
          kiki_0187.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0187.m4a
analyzing
analyzing
          kiki_0188.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0188.m4a
analyzing
          kiki_0189.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0189.m4a
          kiki_0190.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0190.m4a
analyzing
analyzing
          kiki_0191.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0191.m4a
analyzing
          kiki_0192.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0192.m4a
analyzing
          kiki_0193.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0193.m4a
          kiki_0194.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0194.m4a
analyzing
analyzing
          kiki_0195.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0195.m4a
          kiki 0196.m4a KikiBoubadata/kikibouba test/kiki\kiki 0196.m4a
analyzing
analyzing
          kiki_0197.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0197.m4a
          kiki_0198.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0198.m4a
analyzing
analyzing
          kiki_0199.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0199.m4a
analyzing
          kiki_0200.m4a KikiBoubadata/kikibouba_test/kiki\kiki_0200.m4a
analyzing
          bouba_0101.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0101.m4a
analyzing
          bouba_0102.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0102.m4a
          bouba_0103.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0103.m4a
analyzing
analyzing
          bouba_0104.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0104.m4a
          bouba_0105.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0105.m4a
analyzing
analyzing
          bouba_0106.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0106.m4a
          bouba 0107.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0107.m4a
analyzing
          bouba_0108.m4a KikiBoubadata/kikibouba_test/bouba_0108.m4a
analyzing
analyzing
          bouba_0109.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0109.m4a
analyzing
          bouba_0110.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0110.m4a
analyzing
          bouba_0111.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0111.m4a
analyzing
          bouba_0112.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0112.m4a
          bouba_0113.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0113.m4a
analyzing
          bouba_0114.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0114.m4a
analyzing
          bouba_0115.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0115.m4a
analyzing
analyzing
          bouba_0116.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0116.m4a
analyzing
          bouba_0117.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0117.m4a
analyzing
          bouba_0118.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0118.m4a
          bouba_0119.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0119.m4a
analyzing
          bouba_0120.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0120.m4a
analyzing
analyzing
          bouba 0121.m4a KikiBoubadata/kikibouba test/bouba\bouba 0121.m4a
analyzing
          bouba_0122.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0122.m4a
          bouba_0123.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0123.m4a
analyzing
analyzing
          bouba_0124.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0124.m4a
analyzing
          bouba 0125.m4a KikiBoubadata/kikibouba test/bouba\bouba 0125.m4a
analyzing
          bouba_0126.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0126.m4a
          bouba_0127.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0127.m4a
analyzing
          bouba_0128.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0128.m4a
analyzing
          bouba_0129.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0129.m4a
analyzing
analyzing
          bouba_0130.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0130.m4a
analyzing
          bouba_0131.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0131.m4a
analyzing
          bouba_0132.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0132.m4a
          bouba_0133.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0133.m4a
analyzing
          bouba_0134.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0134.m4a
analyzing
analyzing
          bouba 0135.m4a KikiBoubadata/kikibouba test/bouba\bouba 0135.m4a
analyzing
          bouba_0136.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0136.m4a
          bouba_0137.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0137.m4a
analyzing
analyzing
          bouba_0138.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0138.m4a
          bouba_0139.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0139.m4a
analyzing
          bouba_0140.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0140.m4a
analyzing
          bouba_0141.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0141.m4a
analyzing
analyzing
          bouba_0142.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0142.m4a
analyzing
          bouba_0143.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0143.m4a
          bouba_0144.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0144.m4a
analyzing
```

```
bouba_0145.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0145.m4a
analyzing
analyzing
          bouba_0146.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0146.m4a
          bouba_0147.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0147.m4a
analyzing
analyzing
          bouba_0148.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0148.m4a
          bouba_0149.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0149.m4a
analyzing
          bouba 0150.m4a KikiBoubadata/kikibouba test/bouba\bouba 0150.m4a
analyzing
          bouba_0151.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0151.m4a
analyzing
analyzing
          bouba_0152.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0152.m4a
analyzing
          bouba_0153.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0153.m4a
analyzing
          bouba_0154.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0154.m4a
          bouba_0155.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0155.m4a
analyzing
analyzing
          bouba_0156.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0156.m4a
analyzing
          bouba_0157.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0157.m4a
analyzing
          bouba_0158.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0158.m4a
          bouba_0159.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0159.m4a
analyzing
analyzing
          bouba_0160.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0160.m4a
          bouba 0161.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0161.m4a
analyzing
analyzing
          bouba_0162.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0162.m4a
          bouba_0163.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0163.m4a
analyzing
analyzing
          bouba_0164.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0164.m4a
analyzing
          bouba_0165.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0165.m4a
analyzing
          bouba_0166.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0166.m4a
analyzing
          bouba_0167.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0167.m4a
          bouba_0168.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0168.m4a
analyzing
          bouba_0169.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0169.m4a
analyzing
          bouba_0170.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0170.m4a
analyzing
analyzing
          bouba_0171.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0171.m4a
analyzing
          bouba_0172.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0172.m4a
          bouba_0173.m4a KikiBoubadata/kikibouba_test/bouba_0173.m4a
analyzing
analyzing
          bouba_0174.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0174.m4a
analyzing
          bouba_0175.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0175.m4a
analyzing
          bouba_0176.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0176.m4a
analyzing
          bouba_0177.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0177.m4a
          bouba_0178.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0178.m4a
analyzing
          bouba_0179.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0179.m4a
analyzing
          bouba_0180.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0180.m4a
analyzing
analyzing
          bouba_0181.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0181.m4a
analyzing
          bouba_0182.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0182.m4a
          bouba 0183.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0183.m4a
analyzing
          bouba_0184.m4a KikiBoubadata/kikibouba_test/bouba_0184.m4a
analyzing
          bouba_0185.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0185.m4a
analyzing
analyzing
          bouba 0186.m4a KikiBoubadata/kikibouba test/bouba\bouba 0186.m4a
analyzing
          bouba_0187.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0187.m4a
          bouba_0188.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0188.m4a
analyzing
analyzing
          bouba_0189.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0189.m4a
analyzing
          bouba_0190.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0190.m4a
analyzing
          bouba_0191.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0191.m4a
          bouba_0192.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0192.m4a
analyzing
          bouba_0193.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0193.m4a
analyzing
          bouba_0194.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0194.m4a
analyzing
analyzing
          bouba_0195.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0195.m4a
analyzing
          bouba_0196.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0196.m4a
          bouba_0197.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0197.m4a
analyzing
          bouba_0198.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0198.m4a
analyzing
          bouba_0199.m4a KikiBoubadata/kikibouba_test/bouba\bouba_0199.m4a
analyzing
analyzing
          bouba 0200.m4a KikiBoubadata/kikibouba test/bouba\bouba 0200.m4a
```

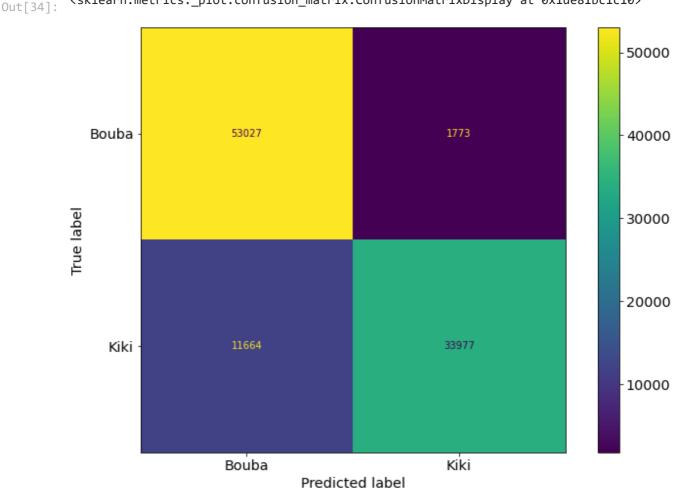
```
In [32]: kiki_test_data = np.stack((kiki_test_ZC, kiki_test_SC), axis=1)
    bouba_test_data = np.stack((bouba_test_ZC, bouba_test_SC), axis=1)
    test_data = np.concatenate((kiki_test_data, bouba_test_data))
```

```
In [34]: # Confusion table

C_TEST = confusion_matrix(true_labels_TEST, predicted_labels_TEST)
    print(C_TEST)

disp = ConfusionMatrixDisplay(C_TEST, display_labels=['Bouba', 'Kiki'])
    disp.plot()
```

[[53027 1773]
[11664 33977]]
<sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at 0x1de81bc1c10>



TEST DATA Accuracy: 0.87,

Balanced accuracy: 0.86,

Precision: 0.95, Recall: 0.74

Again, we see that the classifier performs very well in classifying true Bouba samples as Bouba, with few true Bouba sample classified as Kiki. On the other hand, it classifies true Kiki samples more often as Bouba samples due to the hard decision boundary and overlap between the features.

We see that all performance metrics (confusion matrix, accuracy, balanced accuracy, precision, recall) on the test data are very comparable to the values we saw on the training data, which is a good sign and shows that we are not overfitting to the training data.

### 2.10

Discuss several ways you might build a better Kiki-Bouba classifier.

This classifier has two main drawbacks. Most important is that there is no clear decision boundary in the feature space (zero crossings x spectral centroid), making it impossible to have a perfect classifier from the beginning. One could investigate if this issue could be resolved by choosing more and/or different features that can be extracted from the audio samples. This way we might find a certain feature space in which we can draw a clearer and better decicion boundary between the Kiki and Bouba samples. This would especially reduce the number of true Kiki samples that are classified as belonging to the Bouba class.

One could also check what influence the chosen frame size and hop size have on the classification performance. Perhaps different values would yield a better classifier.

Secondly, the implementation of the classifier can be improved, such that it is faster in predicting which class a sample belongs to. We expect that getting rid of the for-loop that runs over all samples in the classifier call would yield a big improvement in speed already. Also, one could try to find a way to reduce the number of function calls.

In [ ]:			