Audio processing project report

Jere Heimo 50359439, Mai Nguyen 50358236

1. Introduction

In this report we will describe the process of creating a N-nearest neighbour algorithm to predict whether an audio recording is of a car or a tram. Our project started with collecting sound recordings of the two given modes of transportation. After this we created a N-nearest neighbour algorithm using python and trained it with most of the data collected by other members of the course. After this we tested to see the parameters that gave us the best prediction accuracy using a small portion of the data collected by other members of the course. After this we could test the model with our own data. All the steps are described with more detail in this report together with some challenges we encountered, and the steps and assumptions made to address them.

2. Data collection

2.1 Data requirements

There are a few notes on the data that need to be considered.

- Audio contains no speech and no audible artefacts caused by different elements, e.g., winds.
- Audio should be recorded in the same city.
- Audio recording car could contain error if the car had to switch to winter tires.
    - About half the audios from our group will be recorded with cars with winter tires. Similar situation can happen to other groups.

2.2 Collected data information

Our group choose the classes for audio files as “car” and “tram”. Thus, we focused on getting audio files of car and tram moving or about to stop or start to leave from a place.

Time and place of our group recordings: between October and December in 2022 in Tampere.

- Tram recordings were taken along the tram line 3 going between the city centre and Hervanta.
- Car recordings were taken both in city centre of Tampere and Hervanta area.

Device used for our group recordings: Oneplus 6.

Device used for other groups’ recordings for training and validation: iPhone.

More information about the recordings of our group can be found from user ID: jere_heimo [1]

3. Feature extraction

3.1 About features

In this project, to not overconsume time and effort, we used librosa Feature extraction library [2]. The library provides a wide variety of features to extract from audio files. In this project, we choose the five features that are the most studied in the course: Zero crossing rate, Chroma variant “Chroma Energy Normalized”, Mel-frequency cepstral coefficients (MFCC), Spectral centroid.

We chose the first element from the outputs of the four features to compare through histogram.

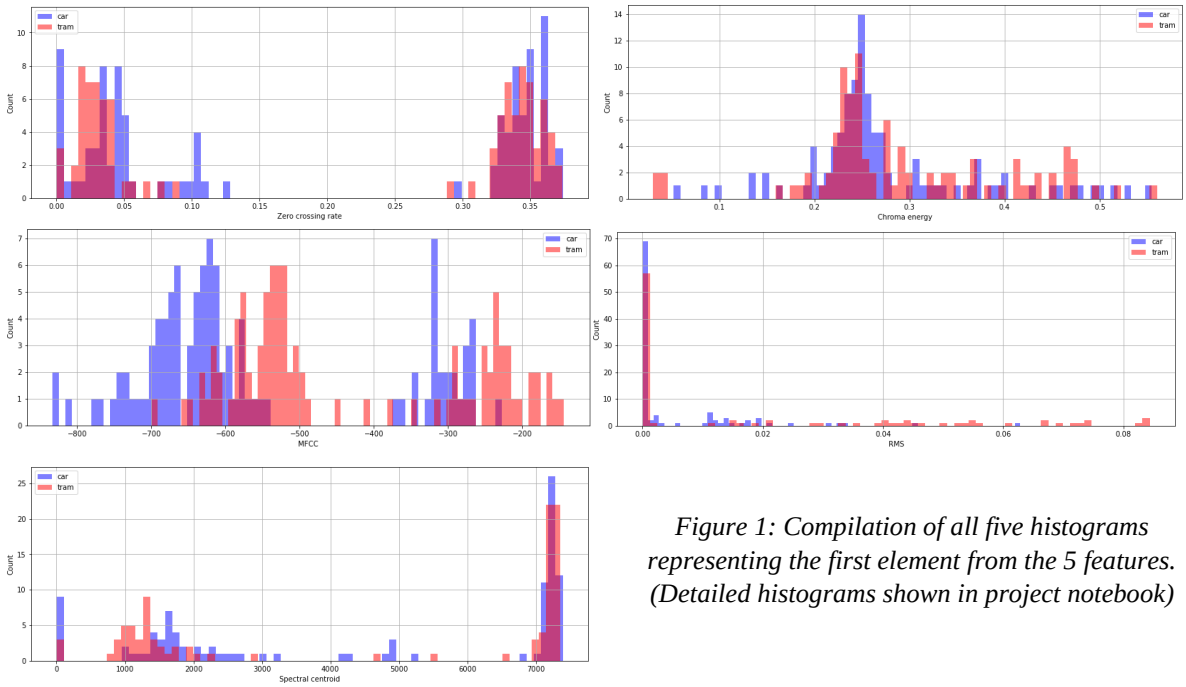

Figure 1: Compilation of all five histograms representing the first element from the 5 features. (Detailed histograms shown in project notebook)

3.2 Chosen feature and explanation

As we have seen from the histograms, a lot of overlapping parts occurred in the histograms. This is because we only took the first element to represent. However, from the histograms, we can see that the MFCCs histogram did not have too much overlapping as the other four.

The MFFC of each audio represent the coefficients that make up the Mel-frequency cepstrum of the signal. The MFC is a power spectrum based on the discrete cosine transform of the logarithmic power spectrum that uses the Mel scale [3]. Normally we get the MFCC of a part of the signal by first taking the Fourier transform of a window of the signal, mapping it to the Mel scale and then putting the scale to logarithmic. After this we can take the DCT and get the coefficients from the amplitudes of the spectrum.

Thus, for the best accuracy, we decided to extract 20 MFCC per frame and then take the mean of each coefficient over all frames. The more coefficients we have, the more accurately we can compare the audio. Due to the lack of time and equipment, we chose not to take more than 20 coefficients to keep the model simple and the computing time short.

4. Splitting data and train the model

4.1 Requirement on splitting data

In this project, we will use the recommended ratio for splitting data [4]. The total dataset will be 70% for training data, 15% for validation data and 15% for testing data.

Training data: Other students' data files

Validation data: Other students' data files

Testing data: Our group's data files

Since we only managed to record a total of 40 audio files (20 files for car sound and 20 files for tram sound), we utilized 103 training data files for car and 96 training data files for tram, and 18 validation data files for car and 21 validation data files for tram.

The purpose of splitting training data from other students' data files into training data and validation data is to prevent the model from being overfitting. [5] When the model is overfitting, it can not perform that well when encountering unseen data. The validation data will act as an independent, unbiased dataset for comparing the performance of different algorithms trained on our training set. [4]

The ground truth data will all be represented as binary numbers: "0" represents tram, "1" represents car.

4.2 Model choice

N Nearest neighbour algorithm is a simple binary model that can be written manually easily or can be used through scikit learn library. The biggest advantage of this algorithm is its simplicity in both implementation and explanation. For this project we used the scikit learn library. However, a simple version of the model will be shown to briefly explain how the model works in the most basic form.

```
#N nearest neighbor model
def classifier_nnn(x, train_data, train_class, n):
    '''
    x: data to classify
    train_data:
    train_class: corresponding to train_data
    '''
    predict_class = []
    for index in range(x.shape[0]):
        audio = x[index, :]
        distance = []
        closest_dist = 0
        for index2 in range(train_data.shape[0]):
            ref_audio = train_data[index2,:]
            compar = np.sum(np.abs(audio - ref_audio)) #Manhattan distance calculation
            distance.append(compar)

        sorted_dist_index = np.argsort(distance).astype(int)
        n_first = sorted_dist_index[:n] #nth nearest neighbor dist
        n_first_label = []
        for index3 in range(n):
            n_first_label.append(train_class[n_first[index3]])
        class_ = np.median(n_first_label).astype(int)
        predict_class.append(class_)
    return predict_class
```

Figure 2: Example code written by our group to demonstrate how the model works

The main idea behind the model is the find the group of N data points that has the shortest distance to the data point that needs to be classified. The distance is usually calculated by Euclidean distance calculation. In our case, using Manhattan distance calculation can still be valid and give the same result.

In scikit learn library, the model can be built in a more complex way with more parameters to customize the model algorithm. [6] However, in this project, we will only focus on evaluate the result between the weights ("uniform" or "distance") and the number of N (radius of the neighbours).

4.3 Validation result and analysation

After implementing the model and running with validation data, we have the report of accuracy as below.

	"uniform"			"distance"		
	Accuracy	Precision "0" / "1"	Recall "0" / "1"	Accuracy	Precision "0" / "1"	Recall "0" / "1"
1NN	0.90	0.81 / 1.00	1.00 / 0.82	0.90	0.81 / 1.00	1.00 / 0.82
3NN	0.92	0.90 / 0.94	0.95 / 0.89	0.92	0.90 / 0.94	0.95 / 0.89
5NN	0.92	0.90 / 0.94	0.95 / 0.89	0.92	0.90 / 0.94	0.96 / 0.89
7NN	0.92	0.86 / 1.00	1.00 / 0.86	0.92	0.86 / 1.00	1.00 / 0.86
9NN	0.97	1.00 / 0.94	0.95 / 1.00	0.97	1.00 / 0.94	0.95 / 1.00

Table 1: Result from classification report using validation data

From Table 1, we can get the f1 score and evaluate which model version is the best. [7]

We can say that both "uniform" and "distance" weights with 9 Nearest neighbours will give the best result.

However, we would like to look back at our data files' actual problems. As said in Chapter 4, part 4.1, the amount of testing data is limited and is not considered to be large enough for the training data to be rationally large. Thus, we can see the accuracy from 3NN to 7NN are very similar and has approximate same result. And given the amount of training, validation, and testing data, it is better to choose N = 7 than 9, as the radius of neighbours is expanded way too large. It is safe to say that our model is not fed with enough data to be the most accurate model which we will discuss later in

Chapter 6. However, due to the lack of time and resources, we will do the testing data on both weights and with N values to be 7 and 9 to further validate the result.

5. Train the model

After using testing data to test our model, our speculation in Chapter 4, part 4.3 is proved correct.

	"uniform"			"distance"		
	Accuracy	Precision "0" / "1"	Recall "0" / "1"	Accuracy	Precision "0" / "1"	Recall "0" / "1"
7NN	0.93	1.00 / 0.85	0.87 / 1.00	0.97	1.00 / 0.95	0.95 / 1.00
9NN	0.82	1.00 / 0.65	0.74 / 1.00	0.88	1.00 / 0.75	0.80 / 1.00

Table 2: Result from classification report using testing data

From Table 2 we can see the result is better for 7NN with weight = "distance".

However, we can still say that the overall result for weight = "distance" is better and both 7NN and 9NN give very much acceptable accuracy.

6. Conclusion

6.1 Error analysis

As we can see from Chapter 4 and Chapter 5, the result is still not completely highly accurate between validation data and testing data. We would now address several problems that can cause such error in our model and result.

- Data size (training, validation, testing) is way too little.
- Technical difference related to the device used for the recordings.
- Inconsistent factors that have been mentioned before, e.g., winds.

6.2 Ending

During the project, there are troubles causing by mistyping code and not analysing audio files used for the project carefully. We have also had troubles on getting the right features and understand the actual problems in our classification features. Fortunately, the problems are solved through peer review between both members of the group and using the course's study material to better understand the actual meaning behind the audio features. We have learnt not only more knowledge about the course (audio processing) but also learnt how to utilize the help from teacher and built-in library availability to save time and effort.

In the end, the result is acceptable, and both members are satisfied with the workload division and experience and learning from the project.

Workload division:

Jere Heimo: collecting audio files and mainly working on the features extraction part, drafting report.

Mai Nguyen: researching and implementing model, working on the rest of the project, finishing report.

7. Reference

[1] https://freesound.org/people/jere_heimo/

[2] https://librosa.org/doc/main/feature.html

[3] https://en.wikipedia.org/wiki/Mel-frequency_cepstrum

[4] https://mlu-explain.github.io/train-test-validation/

[5] https://www.ibm.com/cloud/learn/overfitting

[6] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier

[7] https://www.scikit-yb.org/en/latest/api/classifier/classification_report.html#:~:text=The%20F1%20score%20is,and%20recall%20into%20their%20computation.