Class Assignment (CA) 1 Report

On

MUSIC RECOMMENDATION SYSTEM

Submitted by:

Eunos Ali Mullah

Registration No:- 11917882

Section:- KM055

Roll. No: -50

DECLARATION

I hereby declare that I have completed this project and this report on my won under the guidance of our teacher Dr. Sagar Pande, except for some extracts and summaries for which the original references are stated herein.

Name of Student – Eunos Ali Mullah
Registration no: 11917882

Date – 29 July. 2021

ACKNOWLEDGEMENT

I would like to express my gratitude toward my university and my teacher, Dr. Sagar Pande for providing me the golden opportunity to do this wonderful Machine Learning project, which also helped me in doing a lot of homework and learning. As a result, I came to know about so many new things. So, I am truly thankful to them.
Moreover, I would like to thank my friends who helped me a lot whenever I got stuck in some problem related to my course. I am thankful to have such good support as they always have my back whenever I need it.

Also, I would like to mention the support system and consideration of my parents who have always been there in my life to make me choose the right thing and oppose the wrong. Without them, I could never have learned and become the person I am now.
I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

ABSTRACT

Recommendation systems attempt to predict the preference or rating that a user would give to an item and thus save our time.

Along with the rapid expansion of digital music formats, managing and searching for songs has become significant. Though music information retrieval (MIR) techniques have been made successfully in the last ten years, the development of music recommender systems is still at a very early stage.

In this project “ Music recommendation System” using Machine Learning, three main resources are used. They are – AudioSet Dataset, Max Audio Embedding Generator, and Annoy.

This system for new audio recommends similar types of music which are present in the AudioSet Dataset.

This model is simple, easy to understand, and superfast.

CONTENTS

1. INTRODUCTION

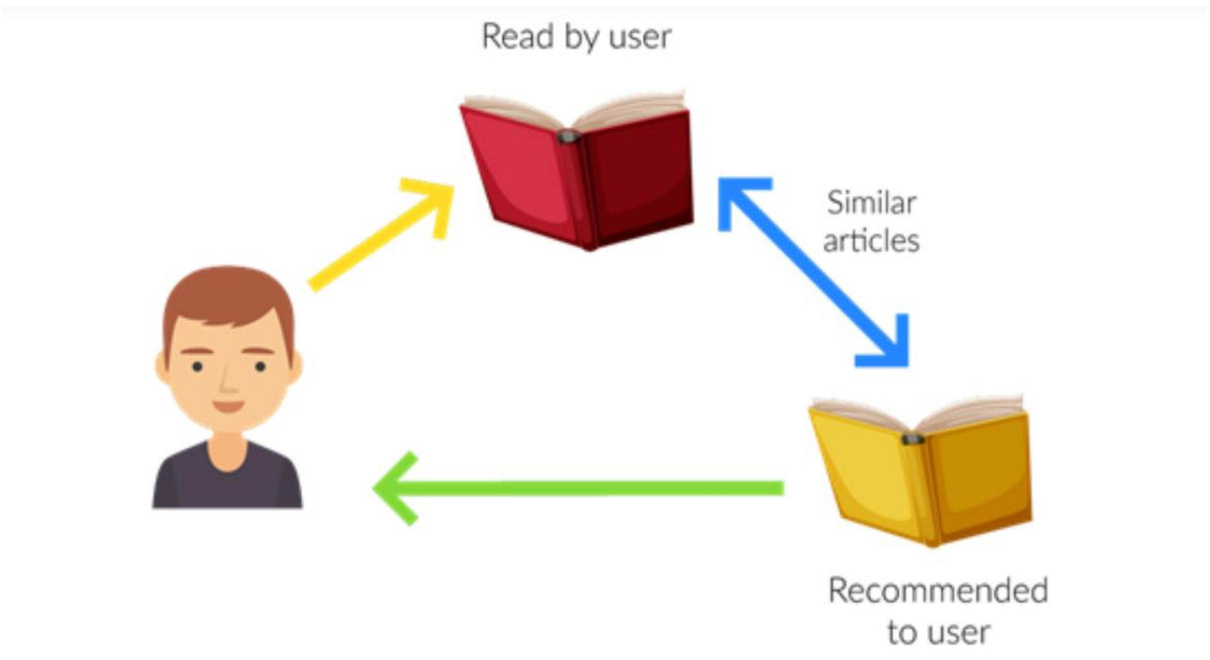
A recommender system, or a recommendation system (sometimes replacing 'system' with a synonym such as platform or engine), is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item.

Recommender systems are used in a variety of areas, with commonly recognized examples taking the form of playlist generators for video and music services, product recommenders for online stores, or content recommenders for social media platforms and open web content recommenders.

These systems can operate using a single input, like music, or multiple inputs within and across platforms like news, books, and search queries. There are also popular recommender systems for specific topics like restaurants and online dating. Recommender systems have also been developed to explore research articles and experts, collaborators, and financial services.

2. TYPES OF RECOMMENDATION SYSTEM

1. Content-Based Filtering



In this type of recommendation system, relevant items are shown using the content of the previously searched items by the users. Here content refers to the attribute/tag of the product that the user like. In this type of system, products are tagged using certain keywords, then the system tries to understand what the user wants and looks in its database and finally tries to recommend different products that the user wants.

Let us take an example of the movie recommendation system where every movie is associated with its genre which in the above case is referred to as tag/attributes. Now let's assume user A comes and initially system doesn't have any data about user A. So initially, the system tries to recommend popular movies to the users or the system tries to get some information about the user by getting a form filled by the user. After some time, users might have given a rating to some of the movies like it gives a good rating to movies based on the action genre and a bad rating to the movies based on the anime genre. So here system recommends action movies to the users. But here you can’t say that the user dislikes animation movies because maybe the user dislikes that movie due to some other reason like acting or story but likes animation movies and needs more data in this case.

Advantage:

- Model doesn’t need data of other users since recommendations are specific to a single user.
- It makes it easier to scale to a large number of users.
- The model can capture the specific interests of the user and can recommend items that very few other users are interested in.

Disadvantage:

- Feature representation of items is hand-engineered to some extent, this tech requires a lot of domain knowledge.
- The model can only make recommendations based on the existing interest of a user. In other words, the model has limited ability to expand on the user’s existing interests.

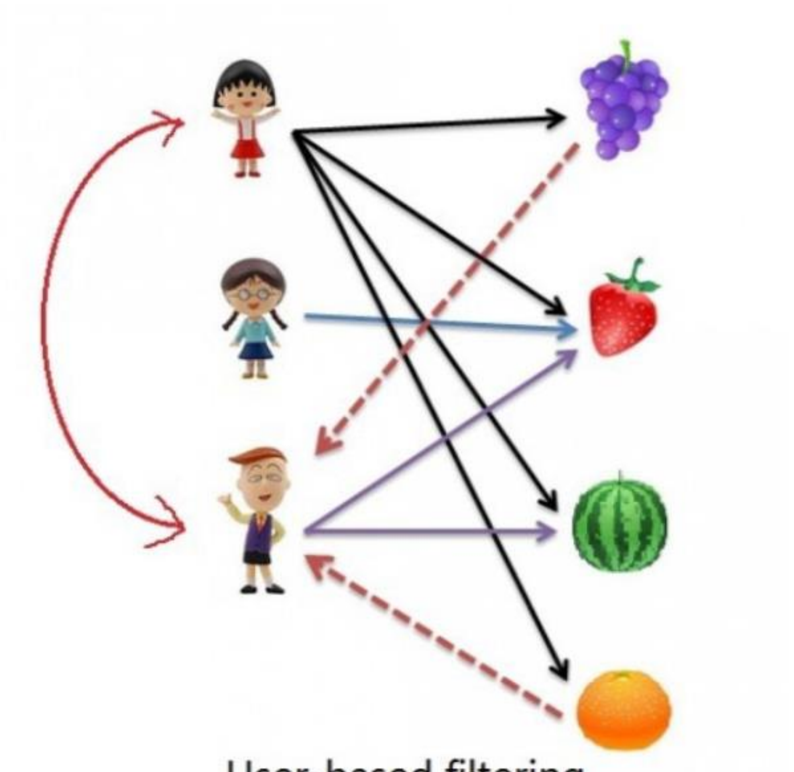
2. Collaborative Based Filtering

Recommending the new items to users based on the interest and preference of other similar users is basically collaborative-based filtering. For eg:- When we shop on Amazon it recommends new products saying “Customer who brought this also brought” as shown below.

This overcomes the disadvantage of content-based filtering as it will use the user Interaction instead of content from the items used by the users. For this, it only needs the historical performance of the users. Based on the historical data, with the assumption that user who has agreed in past tends to also agree in future.

There are 2 types of collaborative filtering:-

A. User-Based Collaborative Filtering

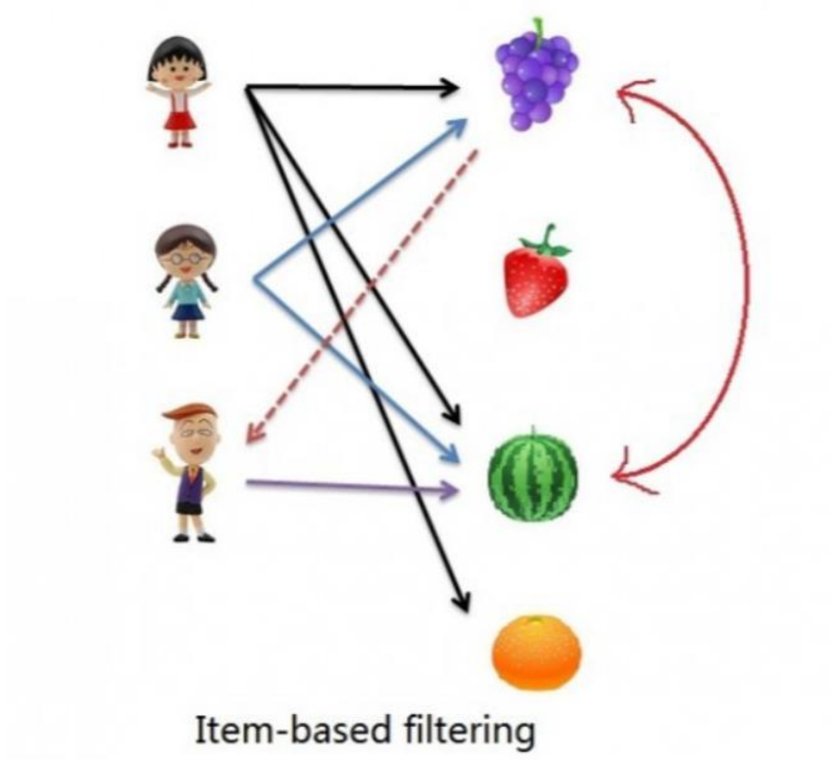

User-based filtering

Rating of the item is done using the rating of neighboring users. In simple words, It is based on the notion of users' similarity.

Let's see an example. On the left side, you can see a picture where 3 children named A, B, C, and 4 fruits i.e, grapes, strawberry, watermelon, and orange respectively.

Based on the image let's assume A purchased all 4 fruits, B purchased only strawberries and C purchased strawberries as well as watermelon. Here A & C are similar kinds of users because of this C will be recommended Grapes and Orange as shown in dotted line.

B. Item-Based Collaborative Filtering


Item-based filtering

The rating of the item is predicted using the user's own rating on neighboring items. In simple words, it is based on the notion of item similarity.

Let us see with an example as told above about users and items. Here the only difference is that we see similar items, not similar users like if you see grapes and watermelon you will realize that watermelon is purchased by all of them but grapes are purchased by Children A & B. Hence Children C is being recommended grapes.

Now after understanding both of them you may be wondering which to use when. Here is the solution if No. Of items is greater than No. Of users go with user-based collaborative filtering as it will reduce the computation power and If No. Of users is greater than No. Of items go with item-based collaborative filtering. For Example, Amazon has lakhs of items to sell but has billions of customers. Hence Amazon uses item-based collaborative filtering because of less no. Of products compared to its customers.

Advantage

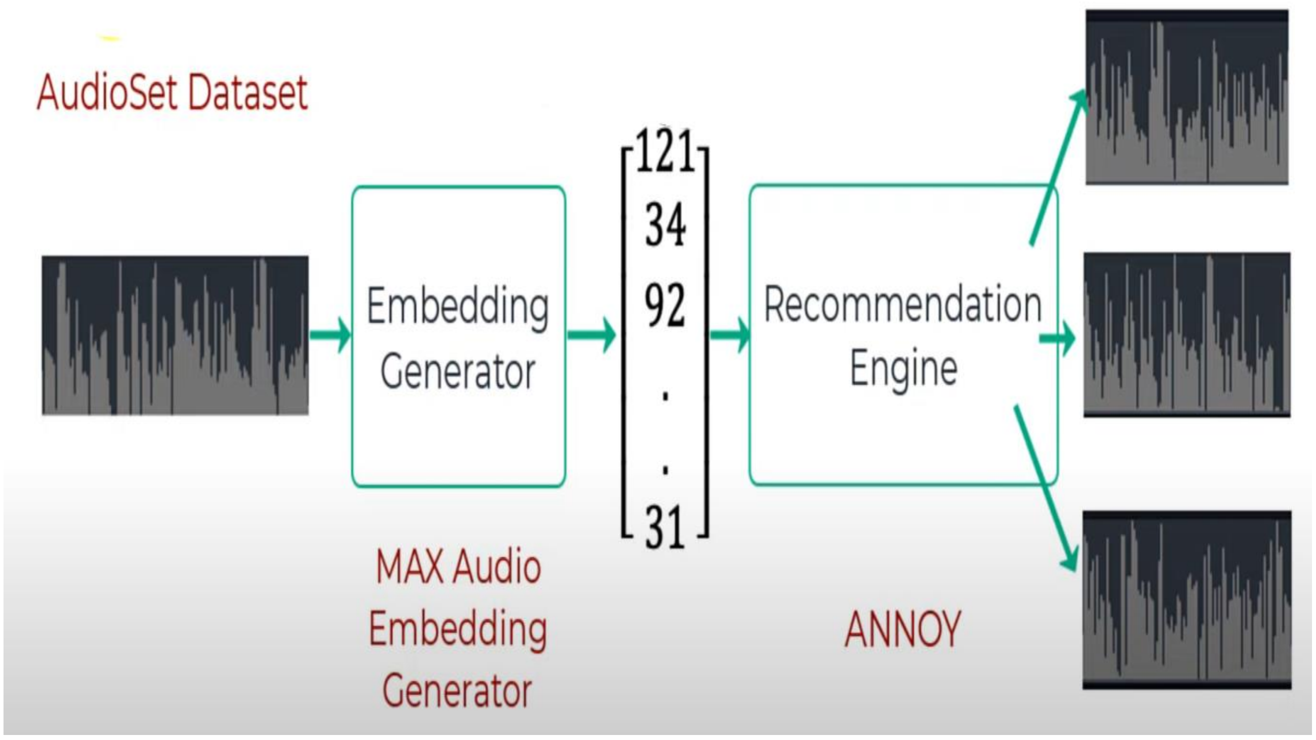
- It works well even if the data is small.
- This model helps the users to discover a new interest in a given item but the model might still recommend it because similar users are interested in that item.
- No need for Domain Knowledge.

Disadvantage

- It cannot handle new items because the model doesn't get trained on the newly added items in the database. This problem is known as Cold Start Problem.
- Side Feature Doesn't have much importance. Here Side features can be actor name or release year in the context of movie recommendation.

3. THE MUSIC RECOMMENDATION MODEL

The music recommendation system is made using the model shown in the following figure.



Steps :

1. A piece of audio is passed to the “Embedding Generator“, which vectorizes the audio or converts the audio into a byte vector.
2. The vector is passed into the “Recommendation Engine”, which recommends similar audio as the input audio.

In order to build the entire model and all these components, 3 major resources are used.

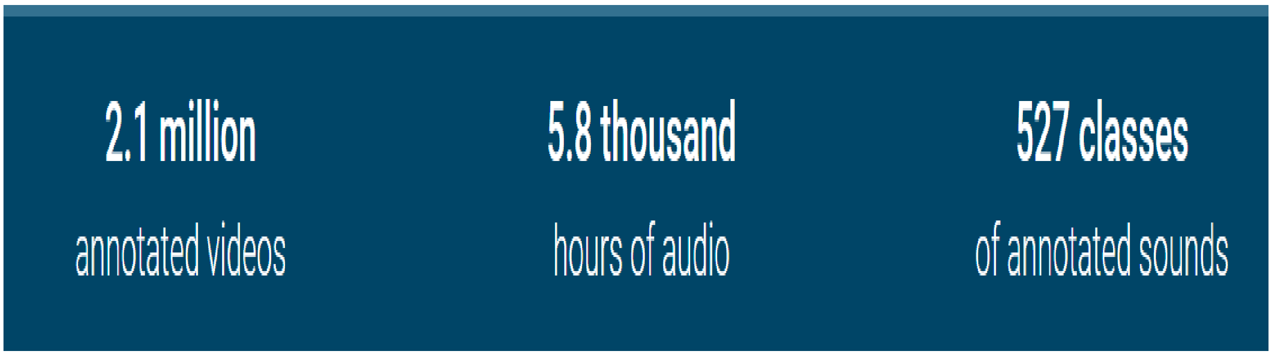
They are as follows –

1. AudioSet Dataset.
2. Max Audio Embedding Generator.
3. Annoy.

1. AudioSet Dataset

AudioSet consists of an expanding ontology of 632 audio event classes and a collection of 2,084,320 human-labeled 10-second sound clips drawn from YouTube videos. The ontology is specified as a hierarchical graph of event categories, covering a wide range of human and animal sounds, musical instruments and genres, and common everyday environmental sounds.

By releasing AudioSet, we hope to provide a common, realistic-scale evaluation task for audio event detection, as well as a starting point for a comprehensive vocabulary of sound events.



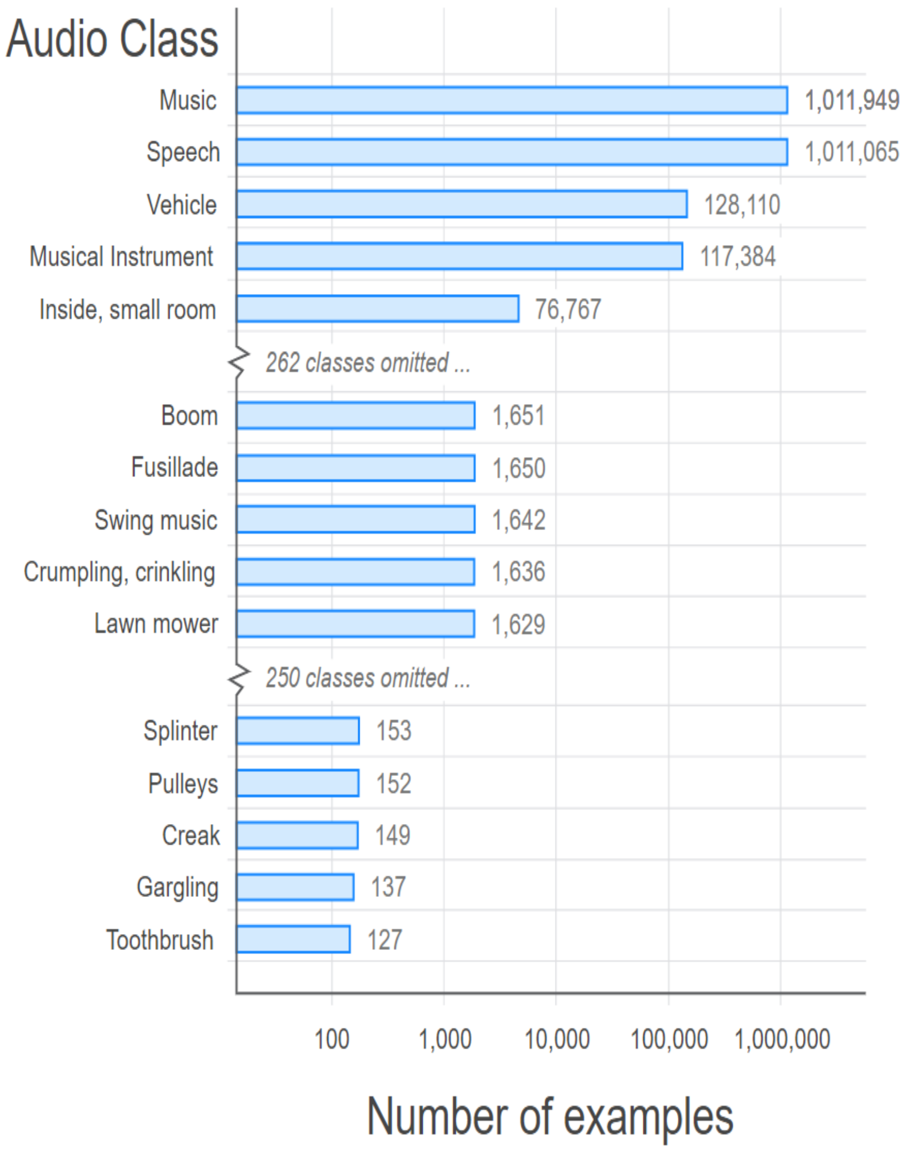
2.1 million annotated videos

5.8 thousand hours of audio

527 classes of annotated sounds

To collect all the data the company worked with human annotators who verified the presence of sounds they heard within YouTube segments. To nominate segments for annotation, they relied on YouTube metadata and content-based search.

The resulting dataset has excellent coverage over the audio event classes in ontology.


Audio Class
Music
1,011,949
Speech
1,011,065
Vehicle
128,110
Musical Instrument
117,384
Inside, small room
76,767
262 classes omitted ...
Boom
1,651
Fusillade
1,650
Swing music
1,642
Crumpling, crinkling
1,636
Lawn mower
1,629
250 classes omitted ...
Splinter
153
Pulleys
152
Creak
149
Gargling
137
Toothbrush
127
100
1,000
10,000
100,000
1,000,000
Number of examples

Features dataset

Frame-level features are stored as TensorFlow.SequenceExample protocol buffers. A TensorFlow. SequenceExample proto is reproduced here in text format:

```
context: {
  feature: {
    key  : "video_id"
    value: {
      bytes_list: {
        value: [YouTube video id string]
      }
    }
  }
  feature: {
    key  : "start_time_seconds"
    value: {
      float_list: {
        value: 6.0
      }
    }
  }
  feature: {
    key  : "end_time_seconds"
    value: {
      float_list: {
        value: 16.0
      }
    }
  }
  feature: {
    key  : "labels"
      value: {
        int64_list: {
          value: [1, 522, 11, 172] # The meaning of the labels can be found
here.
        }
      }
    }
}
feature_lists: {
  feature_list: {
    key  : "audio_embedding"
    value: {
      feature: {
        bytes_list: {
          value: [128 8bit quantized features]
        }
      }
      feature: {
        bytes_list: {
          value: [128 8bit quantized features]
        }
      }
    }
    ... # Repeated for every second of the segment
  }
}
```

The total size of the features is 2.4 gigabytes. They are stored in 12,228 TensorFlow record files, sharded by the first two characters of the YouTube video ID, and packaged as a tar.gz file.

The labels are stored as integer indices. They are mapped to sound classes via class_labels_indices.csv.

The first line defines the column names

```
index,mid,display_name
```

Subsequent lines describe the mapping for each class. For example:

```
0,/m/09x0r,"Speech"
```

which means that “labels” with value 0 indicate segments labeled with “Speech”

2. Max Audio Embedding Generator

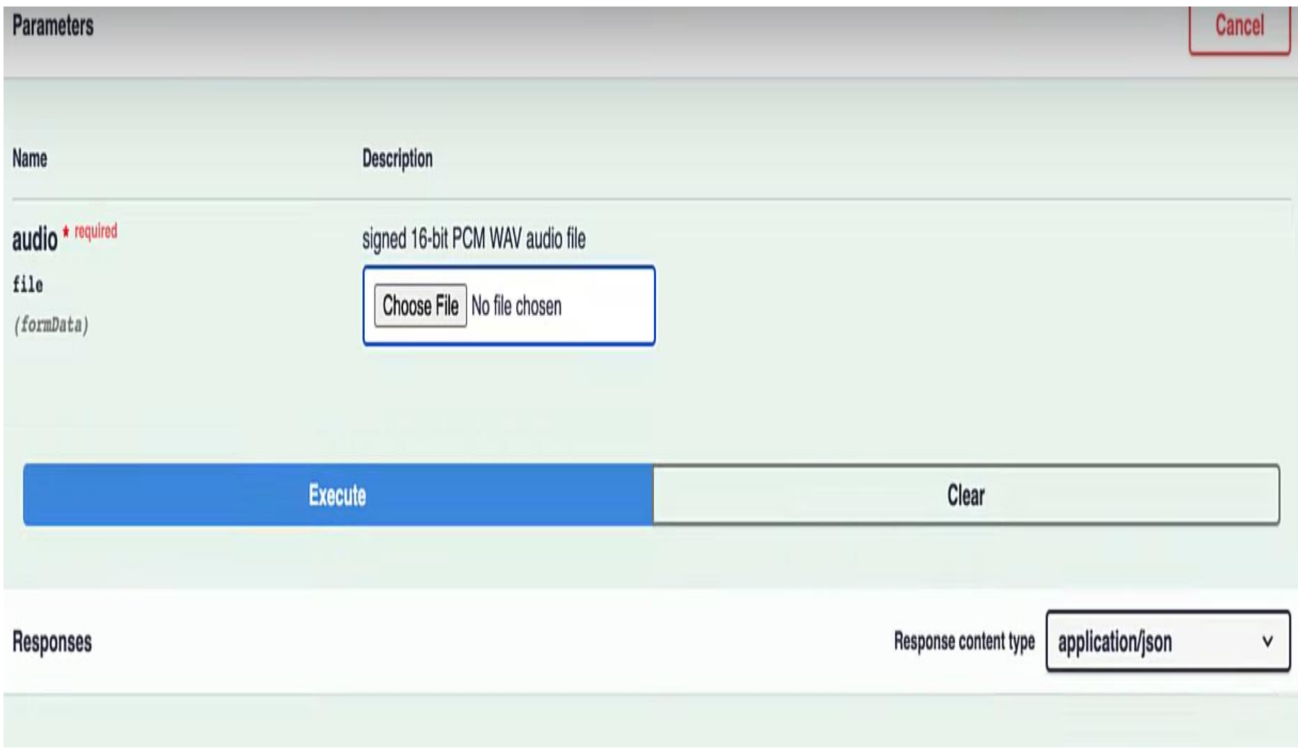
IBM Max Audio embedding Generator is used as the embedding generator in AudioSet Dataset.

It takes a piece of audio or wave file and produces embedding for every single second of the audio.

It basically converts the audio into the form of numbers so that the machine learning algorithm model understands it.

The steps are shown with the help of one example below –

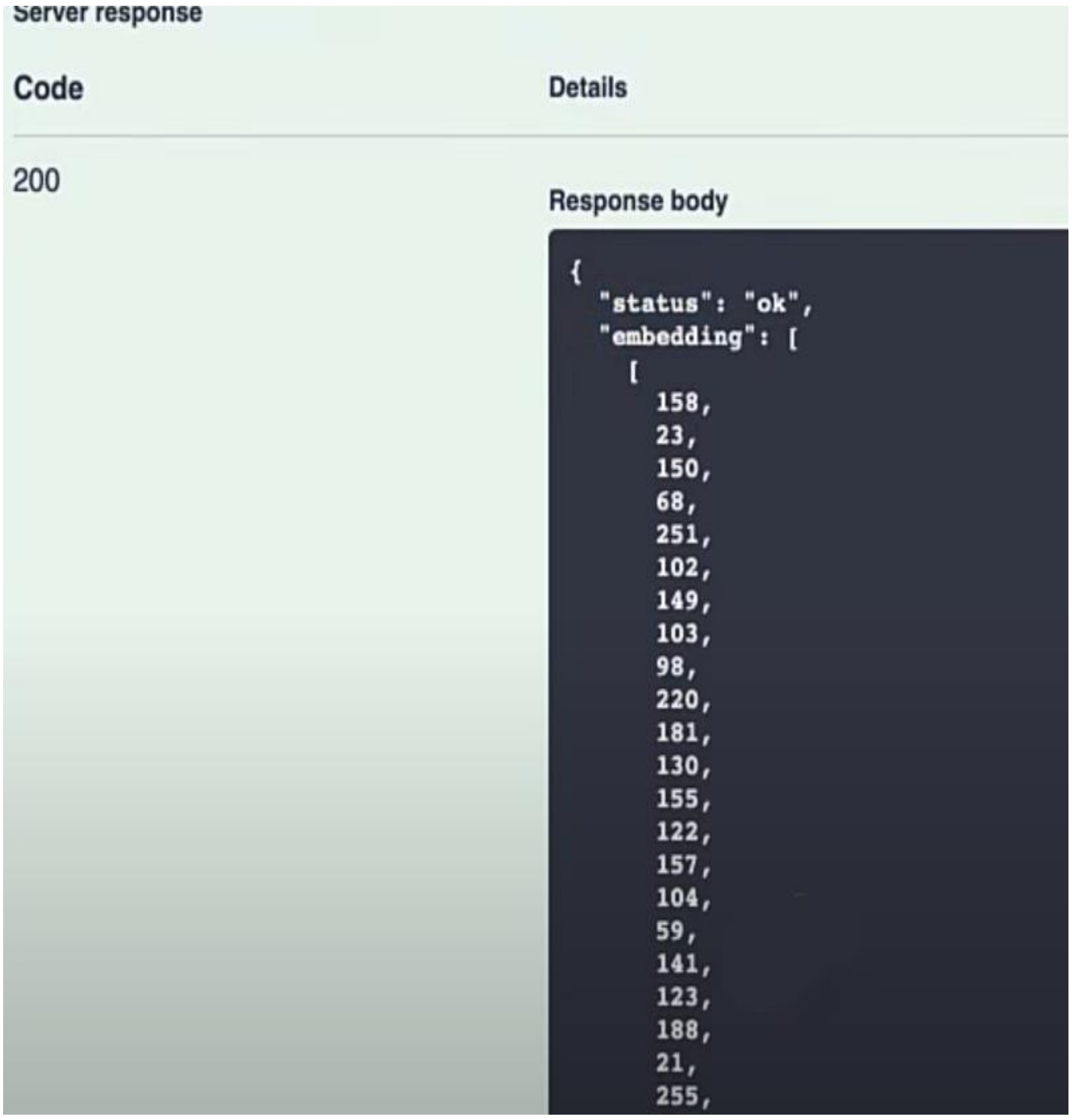
a. Receives an audio or wave file



Name	Description
audio * required file (formData)	signed 16-bit PCM WAV audio file Choose File No file chosen

Responses — Response content type: application/json

b. After clicking on "execute" it gives you an embedded form of audio. It gives a list of lists, each list containing 128 bytes of numbers. Every 128 bytes represents one second of audio.


Server response

Code | Details

200

Response body

```
{
  "status": "ok",
  "embedding": [
    [
      158,
      23,
      150,
      68,
      251,
      102,
      149,
      103,
      98,
      220,
      181,
      130,
      155,
      122,
      157,
      104,
      59,
      141,
      123,
      188,
      21,
      255,
```

3. <u>ANNOY</u>

"Annoy" is used as Music Recommendation System in this Model.

Erik Bernhardsson built Annoy Index while he was working with Spotify.

It helps us find the approximate nearest neighbors.

It is so much faster than other algorithms to find neighbors (like Brute Force Linear Scan).

The steps involved in Annoy Index Algorithm are as follows –

A. Building an Annnoy Index
B. Searching

(A). Building an Annoy Index

The construction of Annoy is explained below with the help of diagrams –

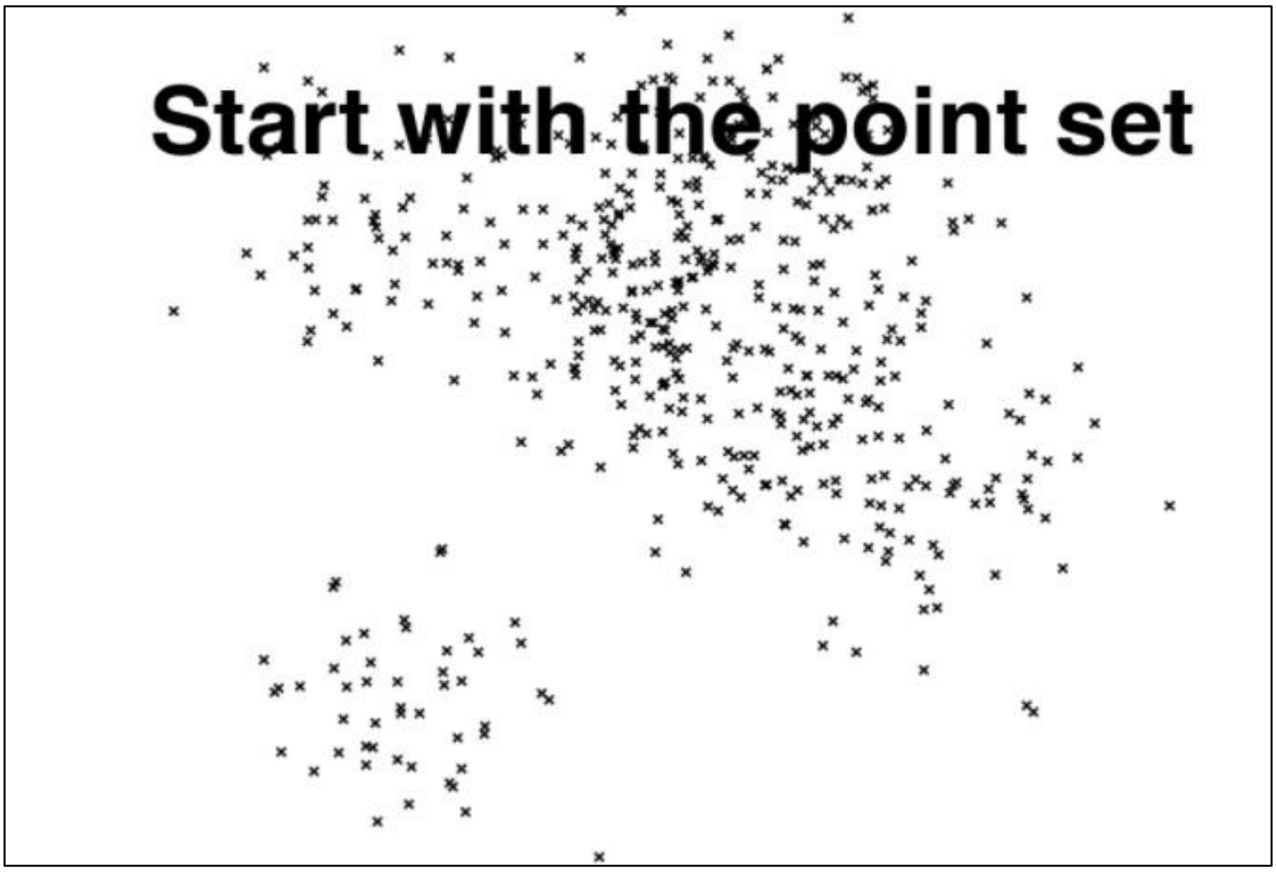


Fig-The Embedding Space

Take two random points and split the data space into halves.

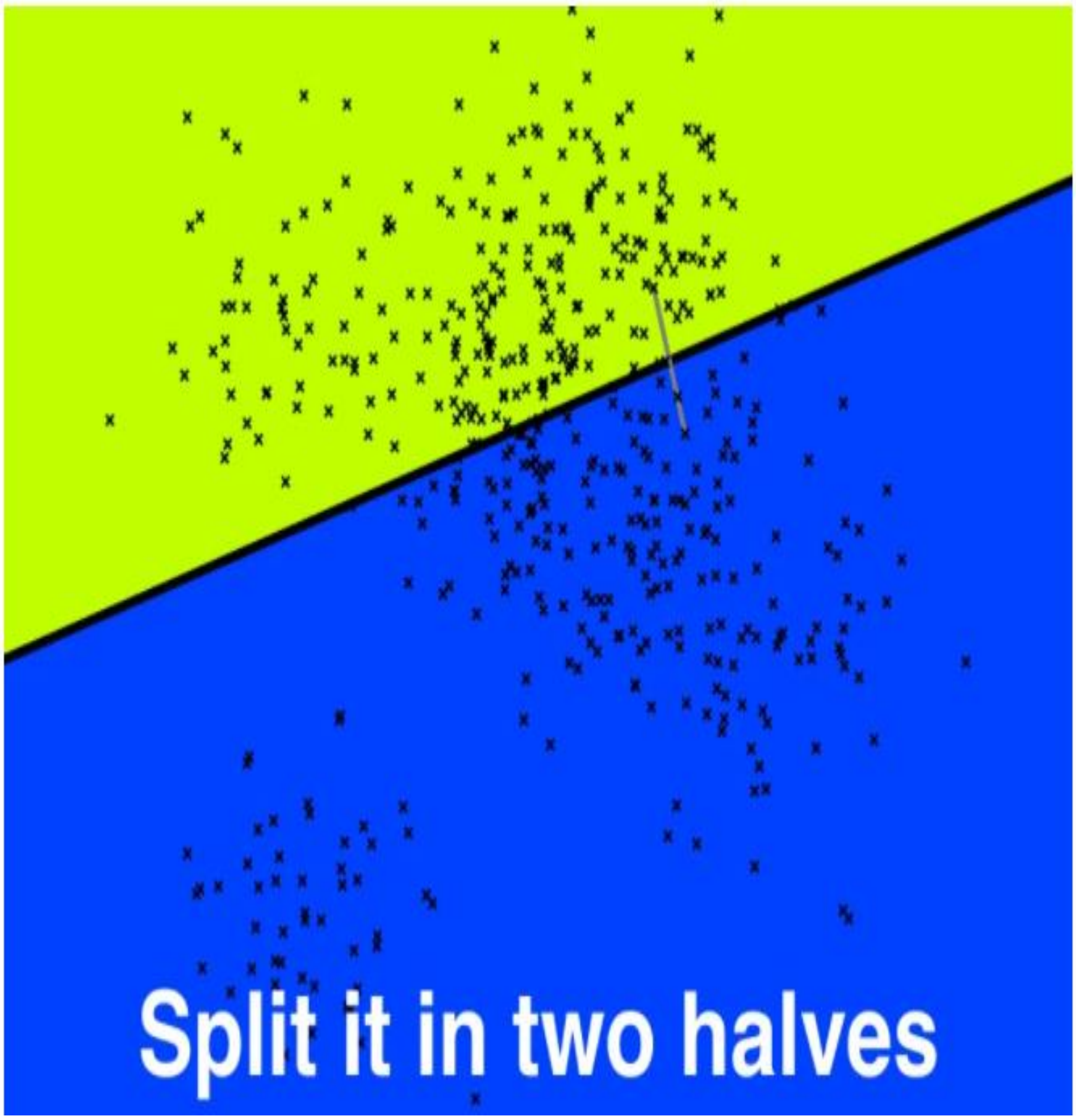
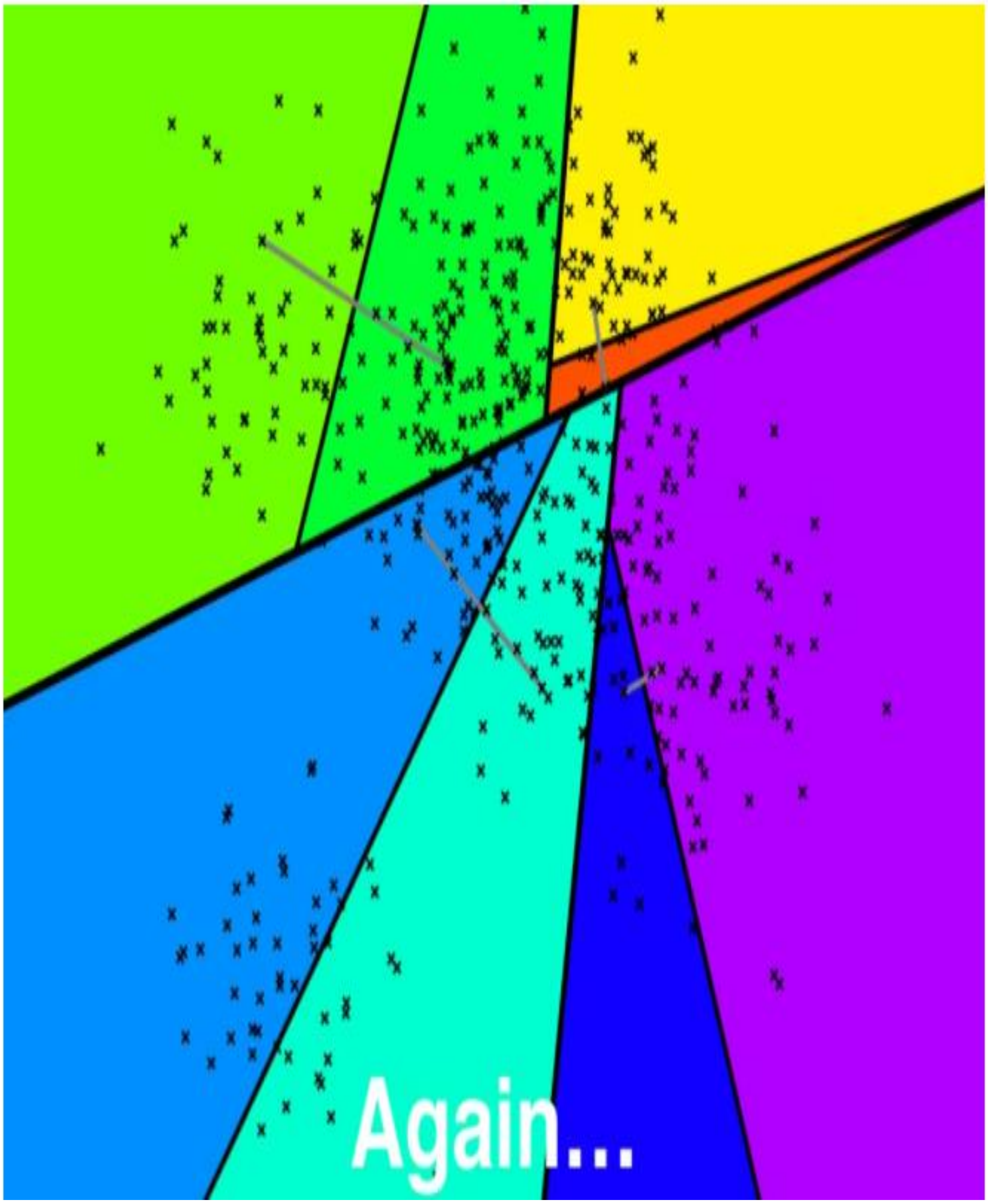
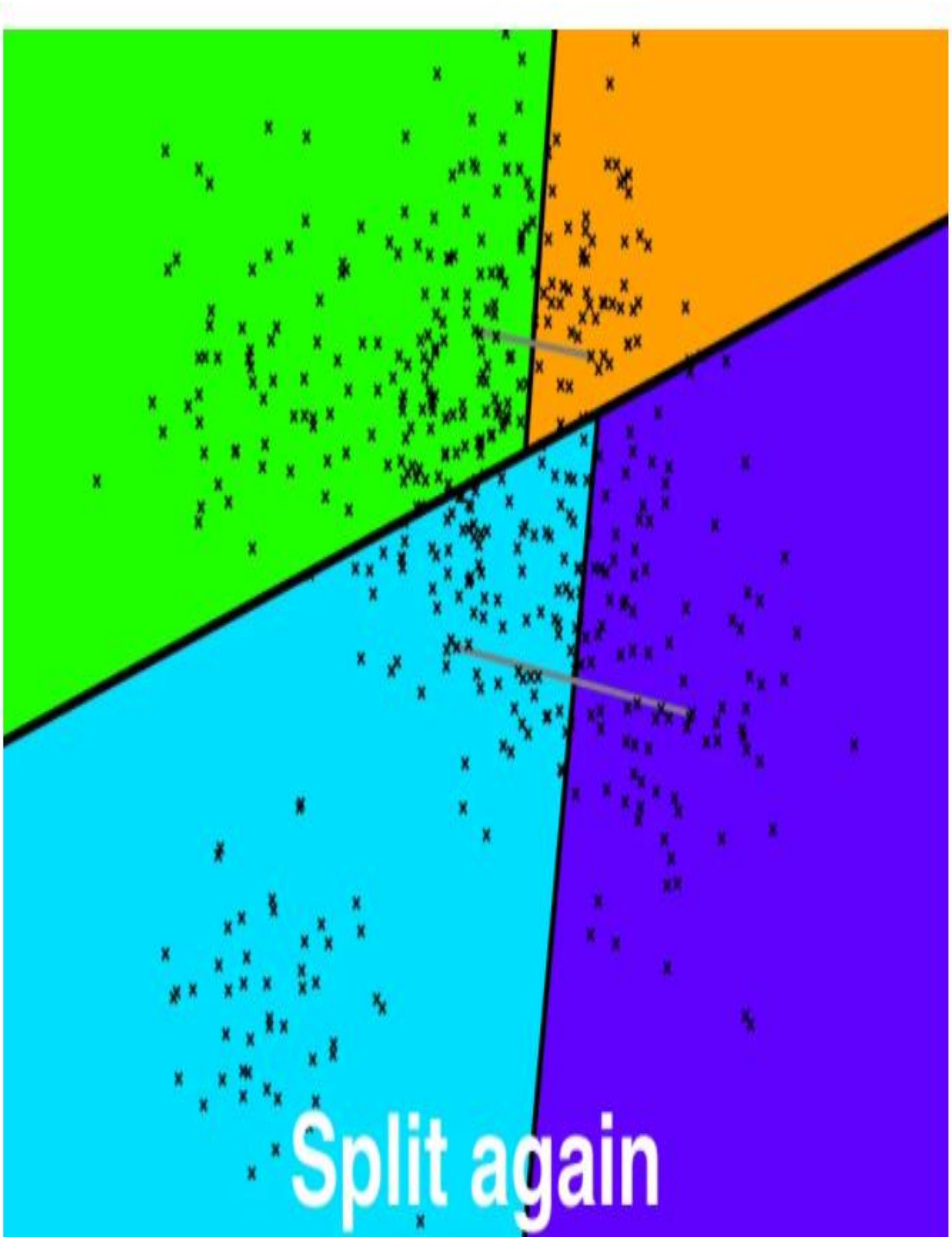
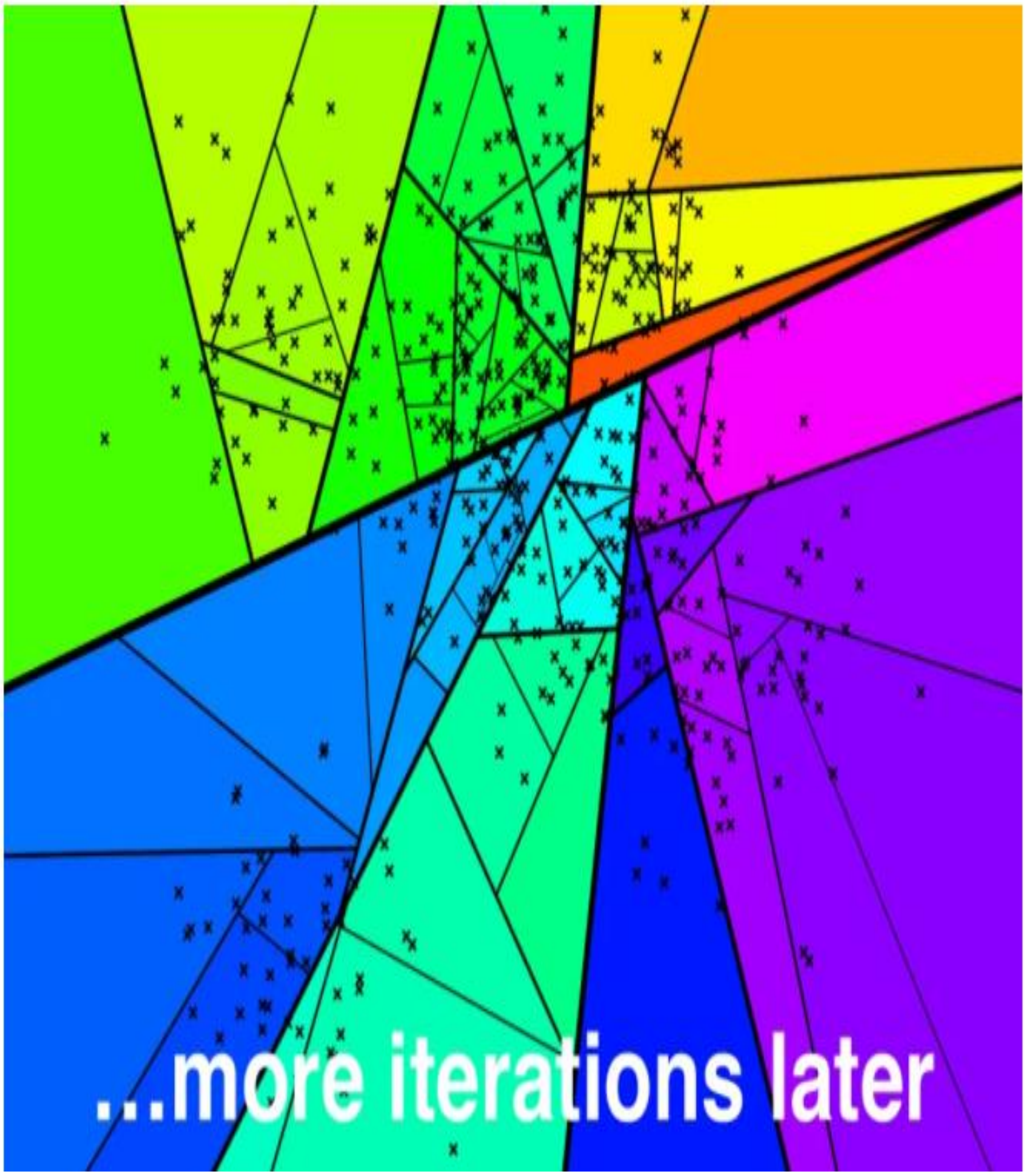


Fig-The Embedding Space


Again...

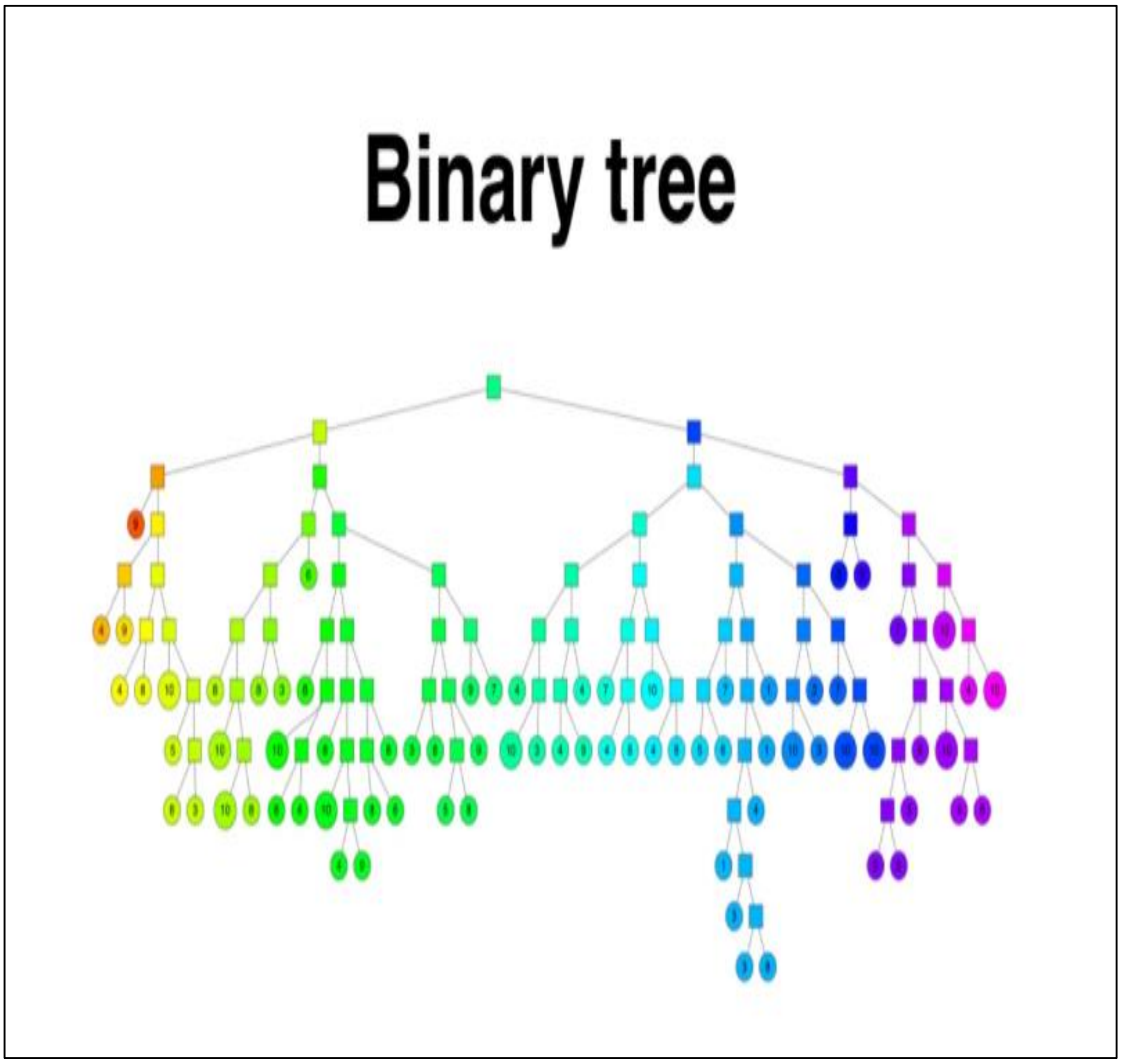

Split again

Continue this process until you get these small box regions.



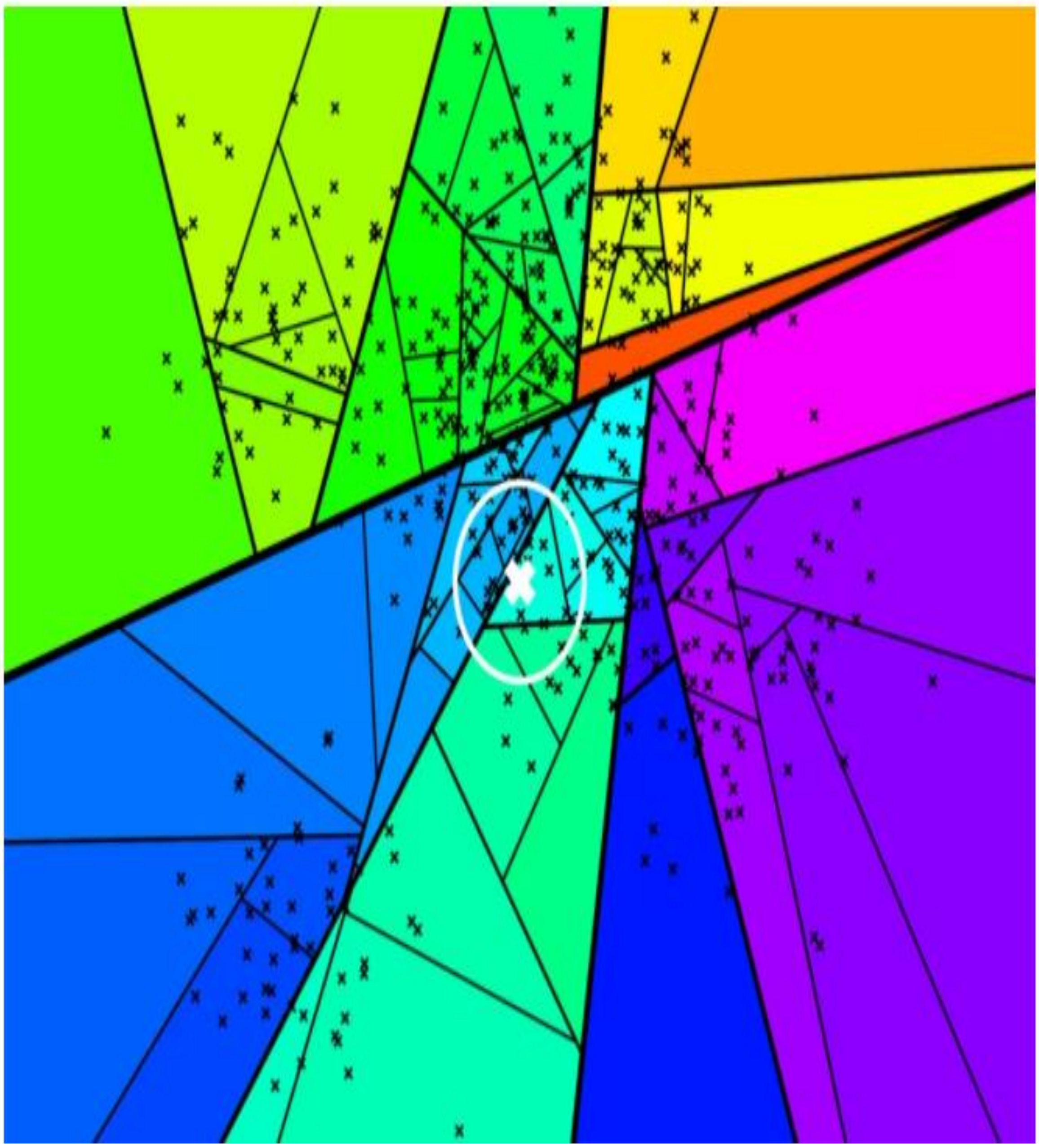
Making Binary Tree

Now a Binary tree is made from these splitting, where every split corresponds to the two nodes.



(B). Searching

Let's assume the actual point in the embedding space lies somewhere (x) as shown in the figure and we want to find the nearest neighbors for that data point.



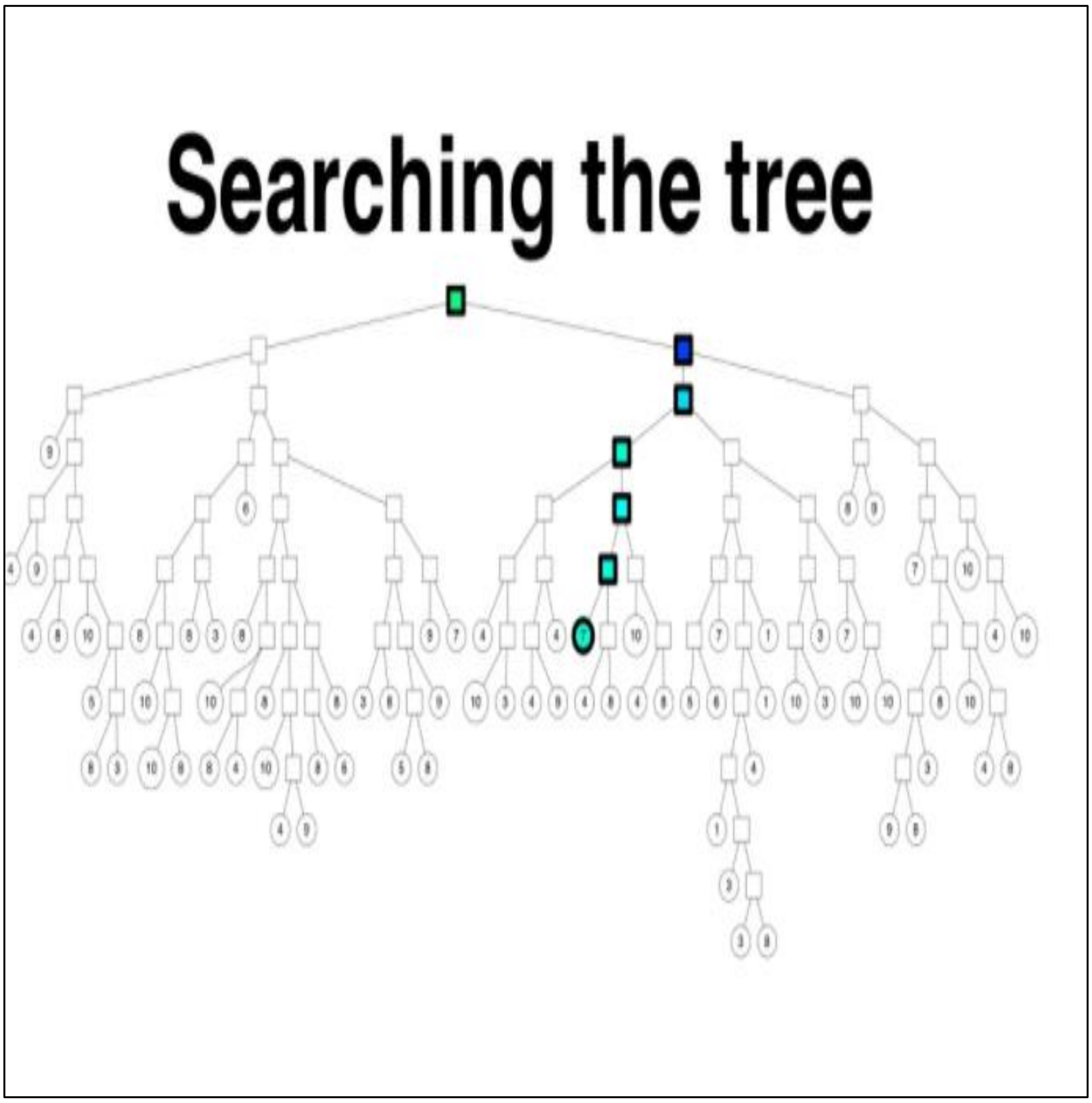
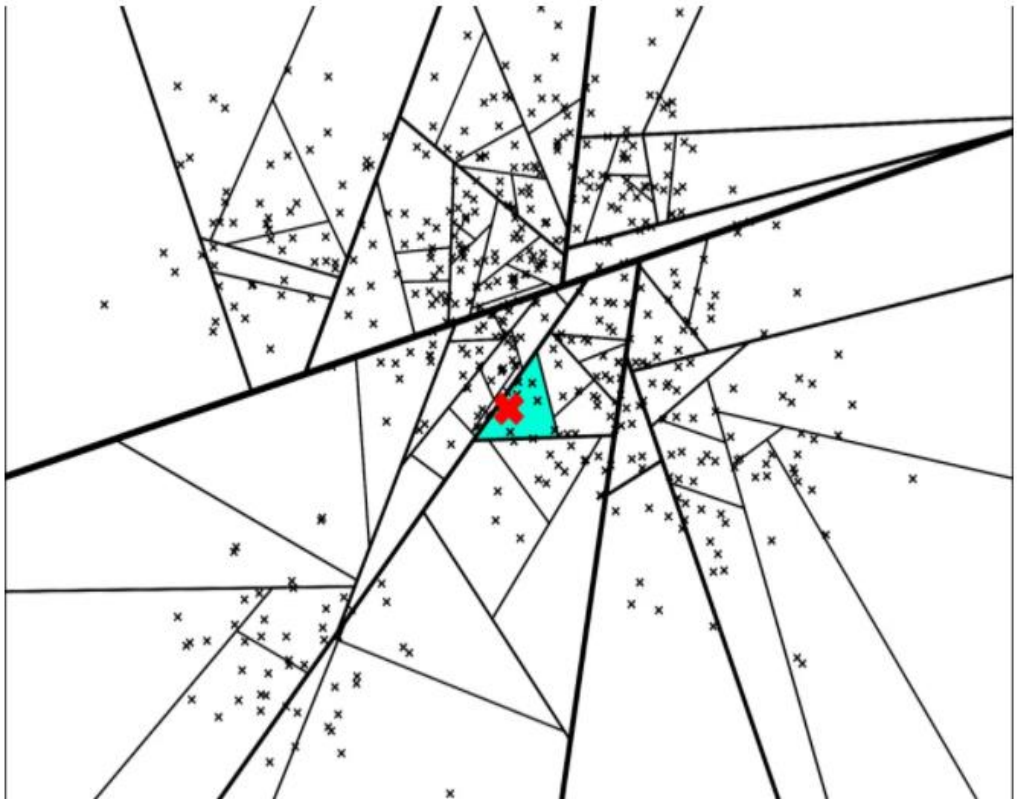


Fig:- searching using Binary Tree

Here, we found the nearest neighbors highlighted by blue color-



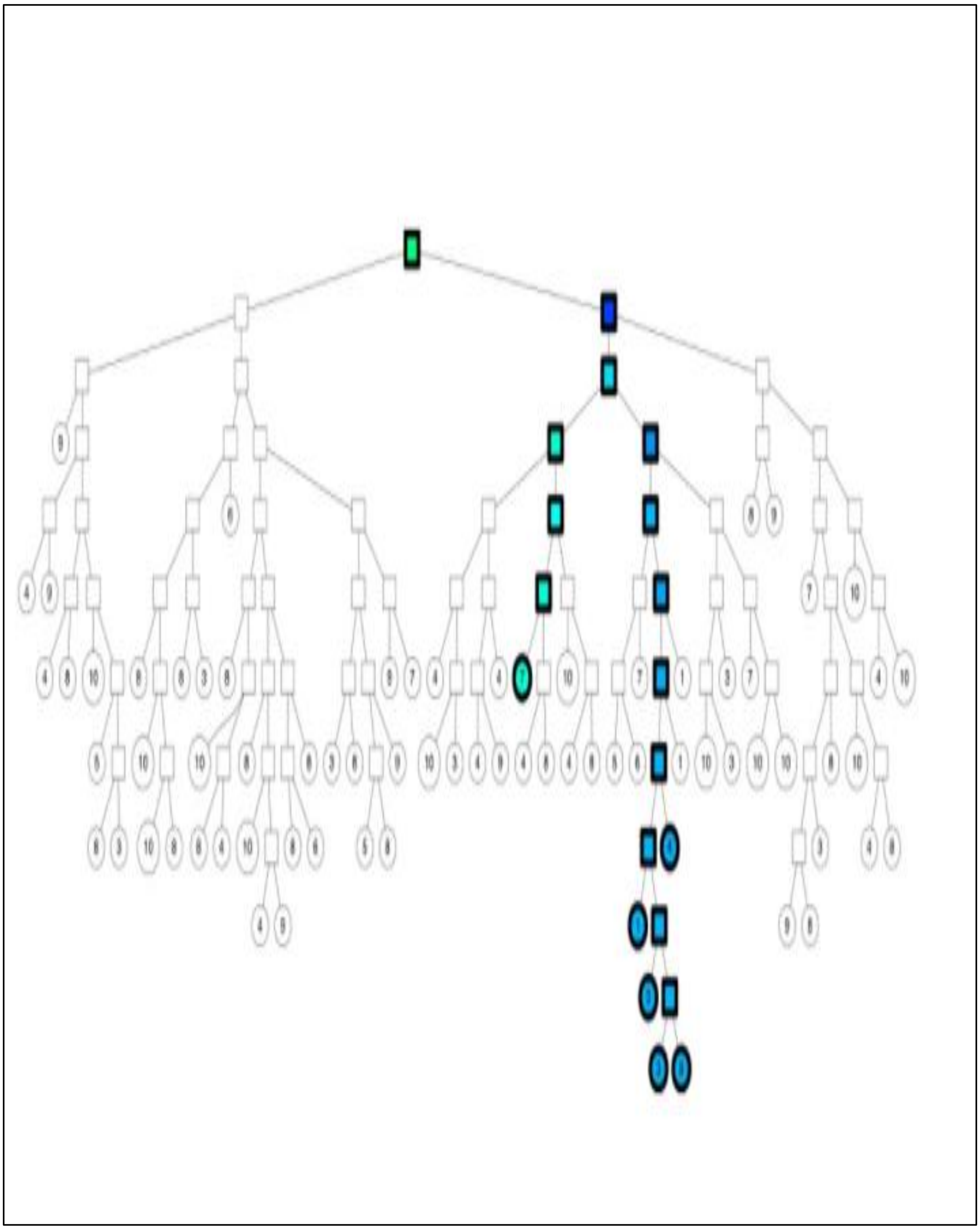
Problems :

- The point that’s the closest isn’t necessarily in the same leaf of the binary tree.
- Two points that are really close may end up on different sides of a split.

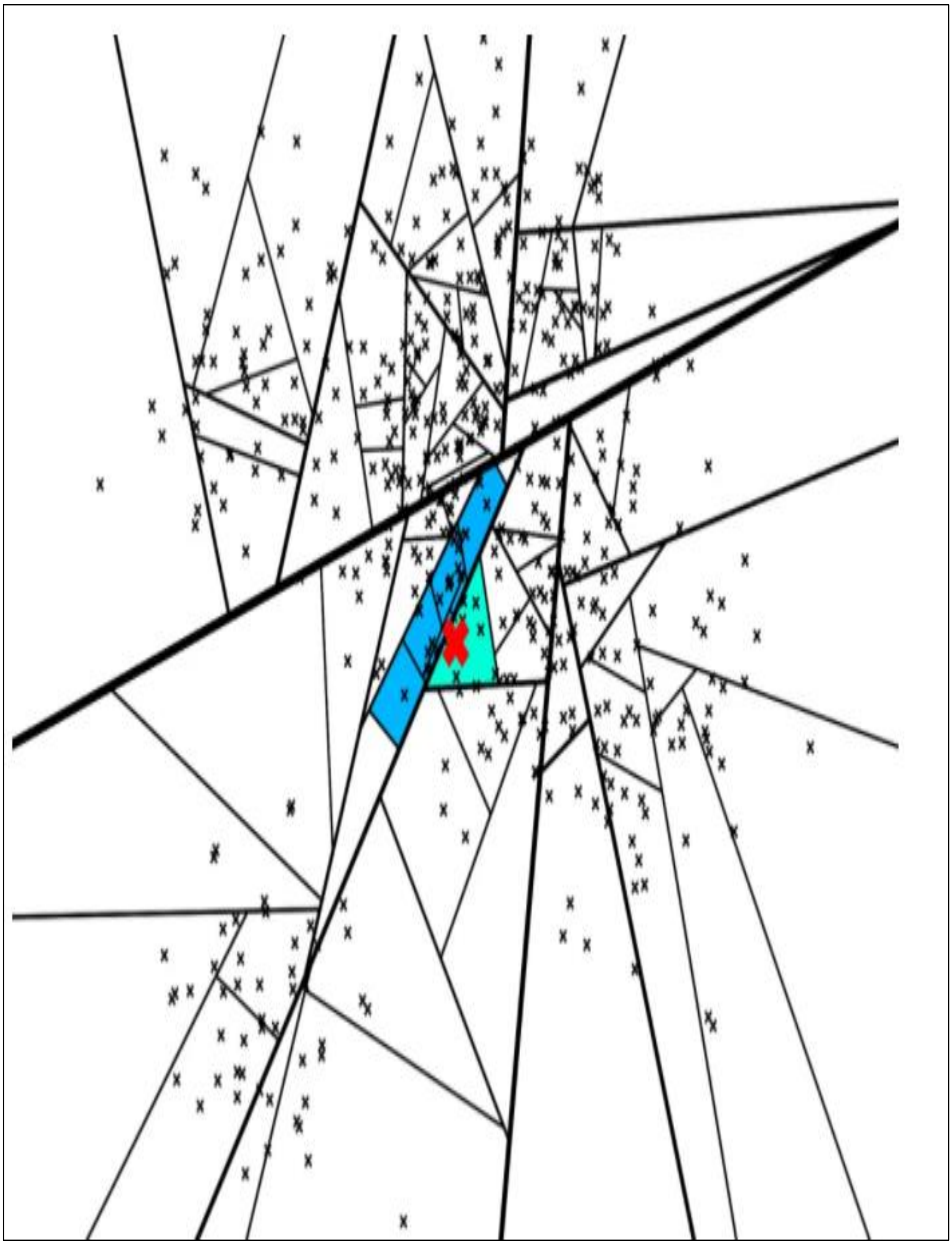
Solutions:

Go to both sides of a split if it’s close.

Hence, the following search is more accurate –



The following Figure shows more accurate nearest neighbors.

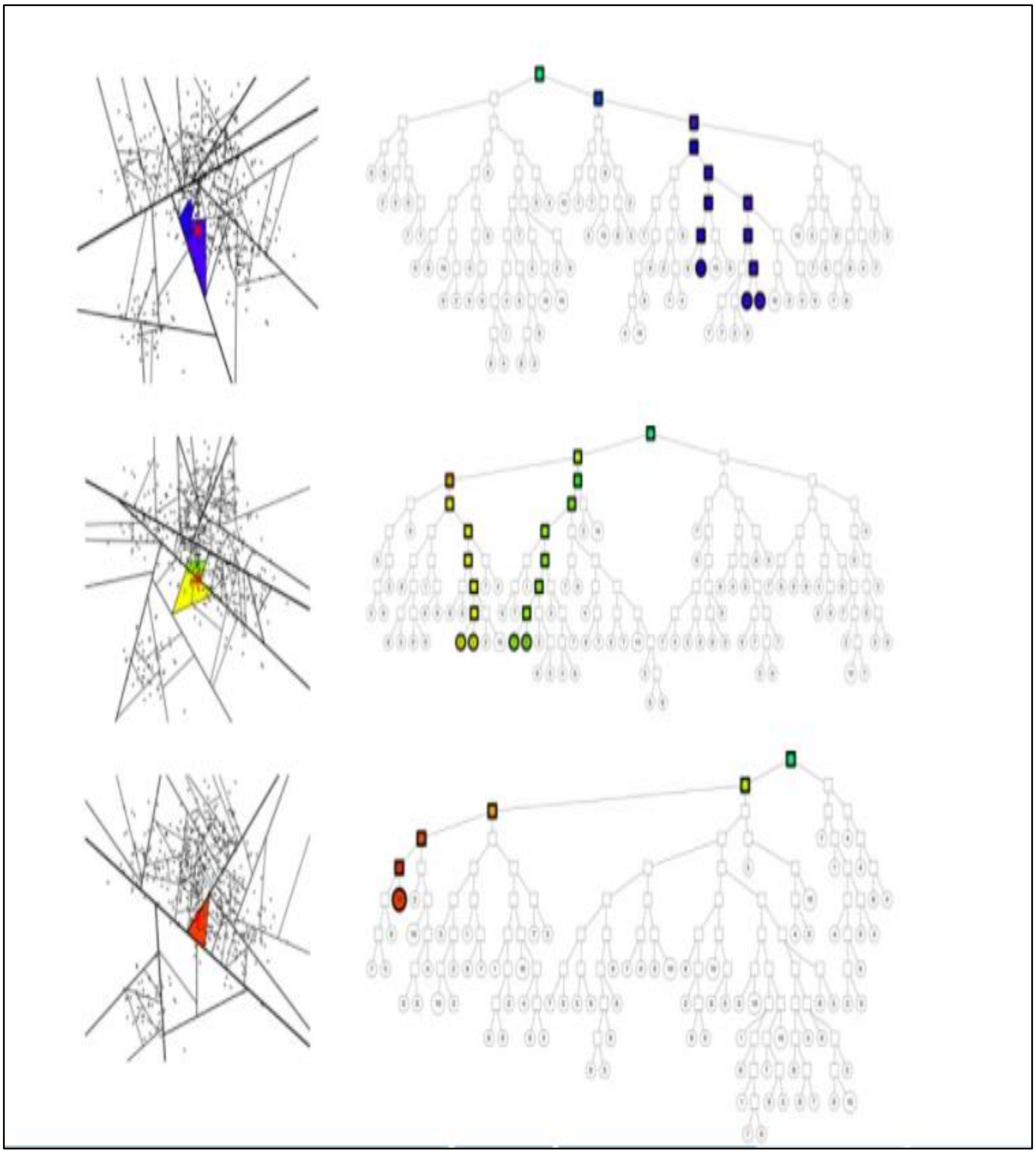


Implementation of the Tree

The entire tree can be implemented by priority queues which are implemented by heaps.

Many Trees

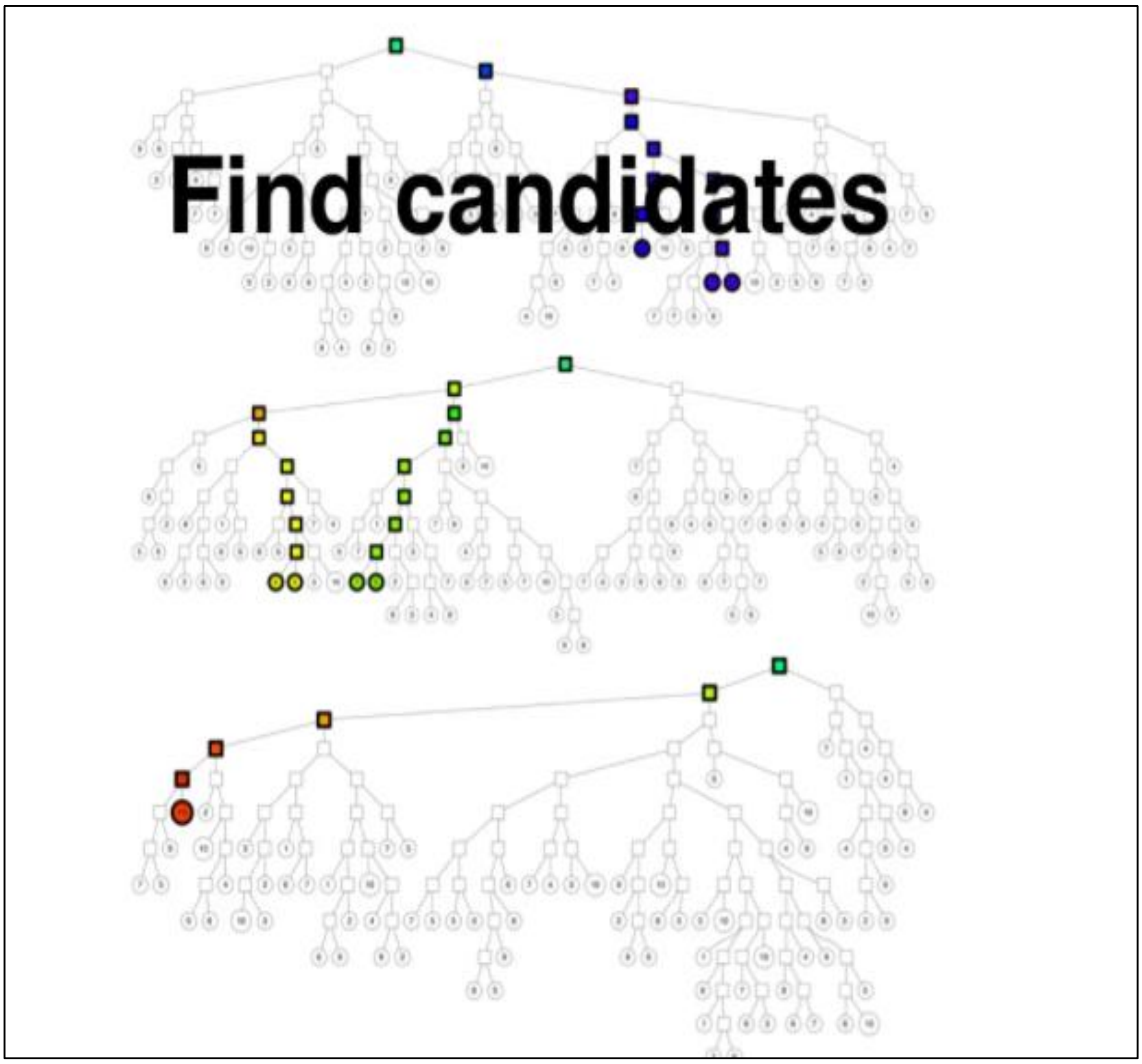
Try to build as many trees as possible so that we don't miss any neighbors.



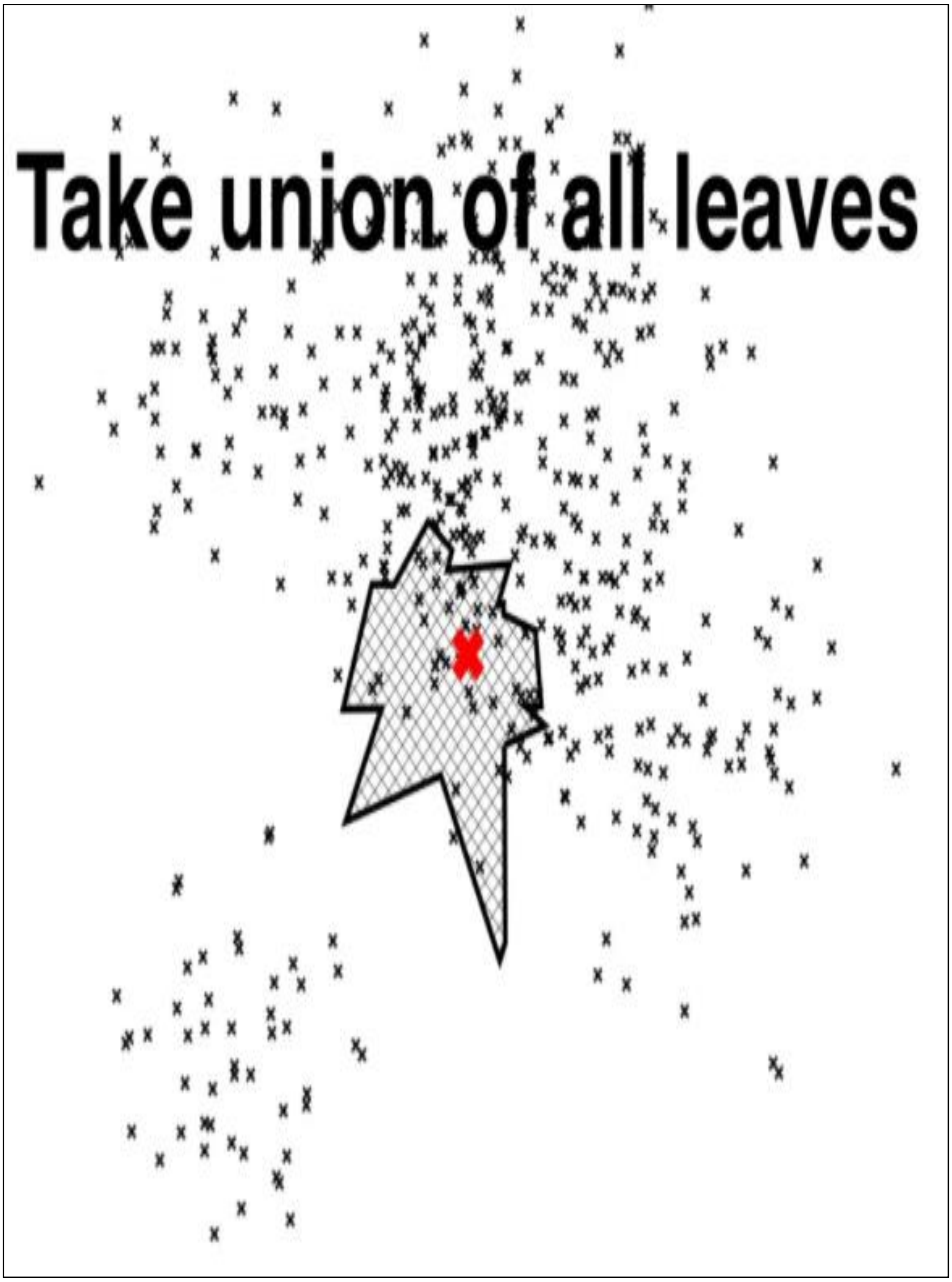
Annoy Query Structure

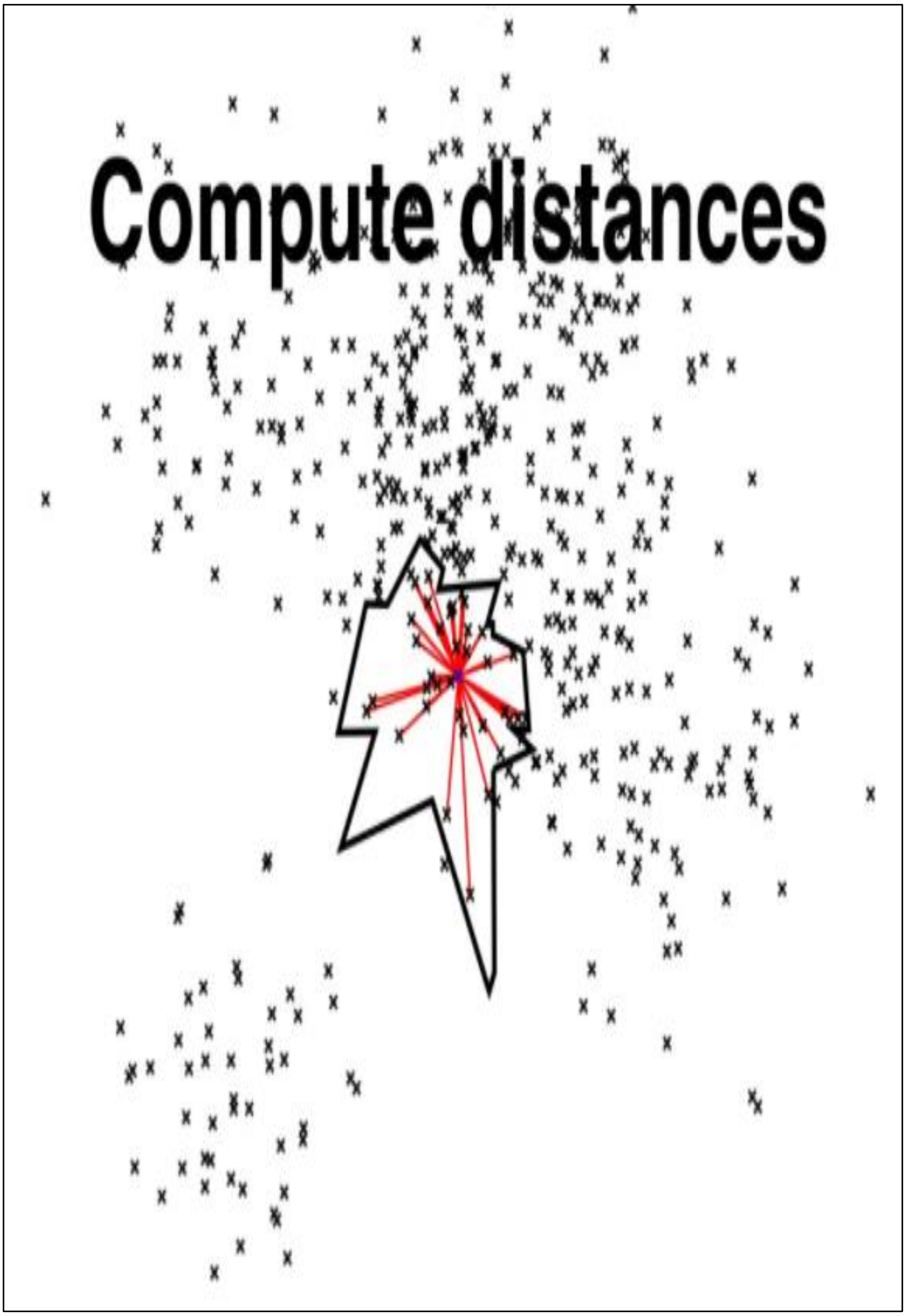
1. Use priority queue to search all trees until we've found k items.
2. Take union and remove duplicates (a lot).
3. Compute distance for remaining items.
4. Return n nearest items.

The steps are represented visually with the help of the following diagrams -



Take union of all leaves

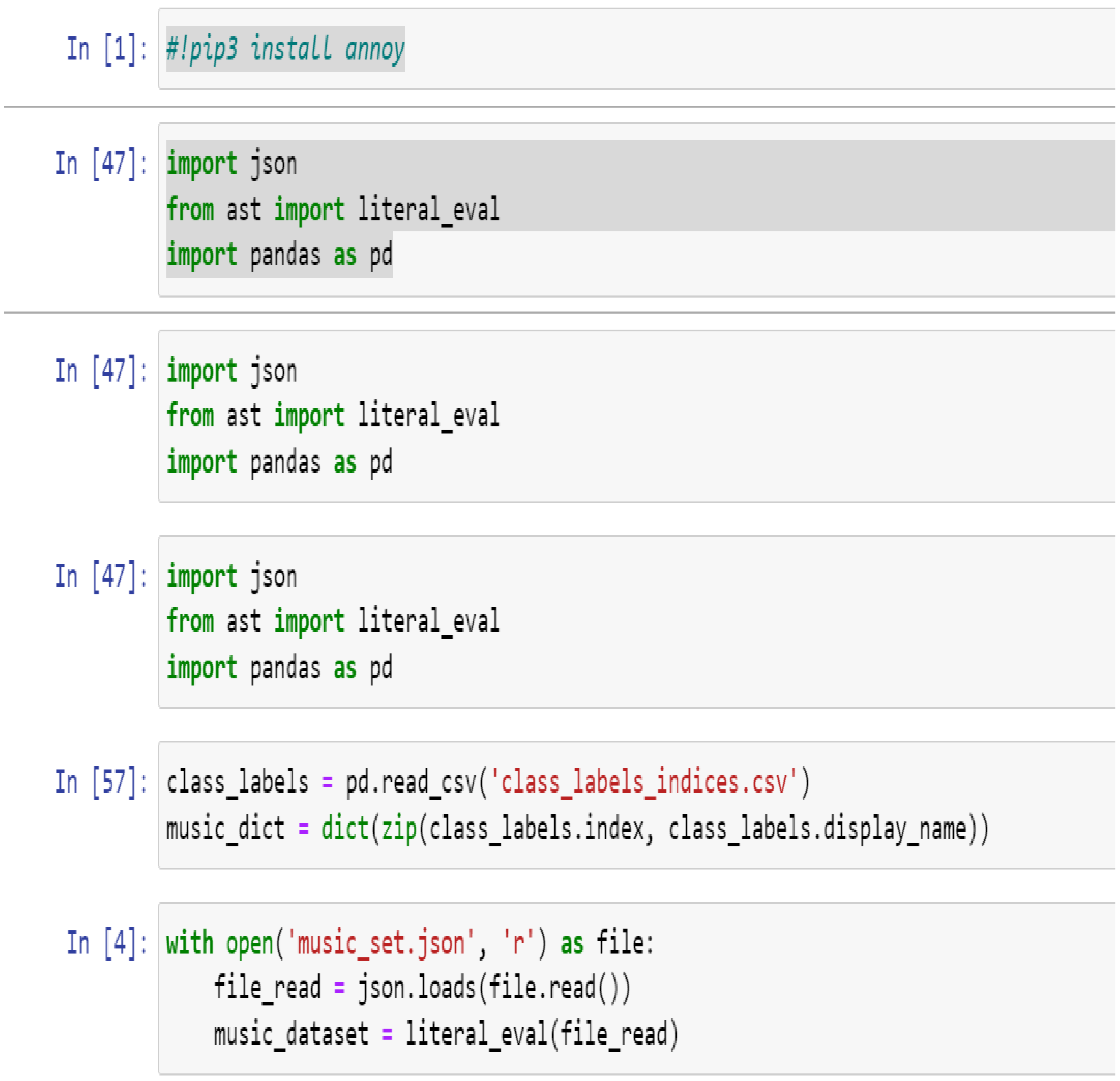




Compute distances

7. RECOMMENDATION SYSTEM CODE

Prerequisite:- music_set.json, which is created from AudioSet DataSet.



```
In [1]: #!pip3 install annoy
```

```
In [47]: import json
         from ast import literal_eval
         import pandas as pd
```

```
In [47]: import json
         from ast import literal_eval
         import pandas as pd
```

```
In [47]: import json
         from ast import literal_eval
         import pandas as pd
```

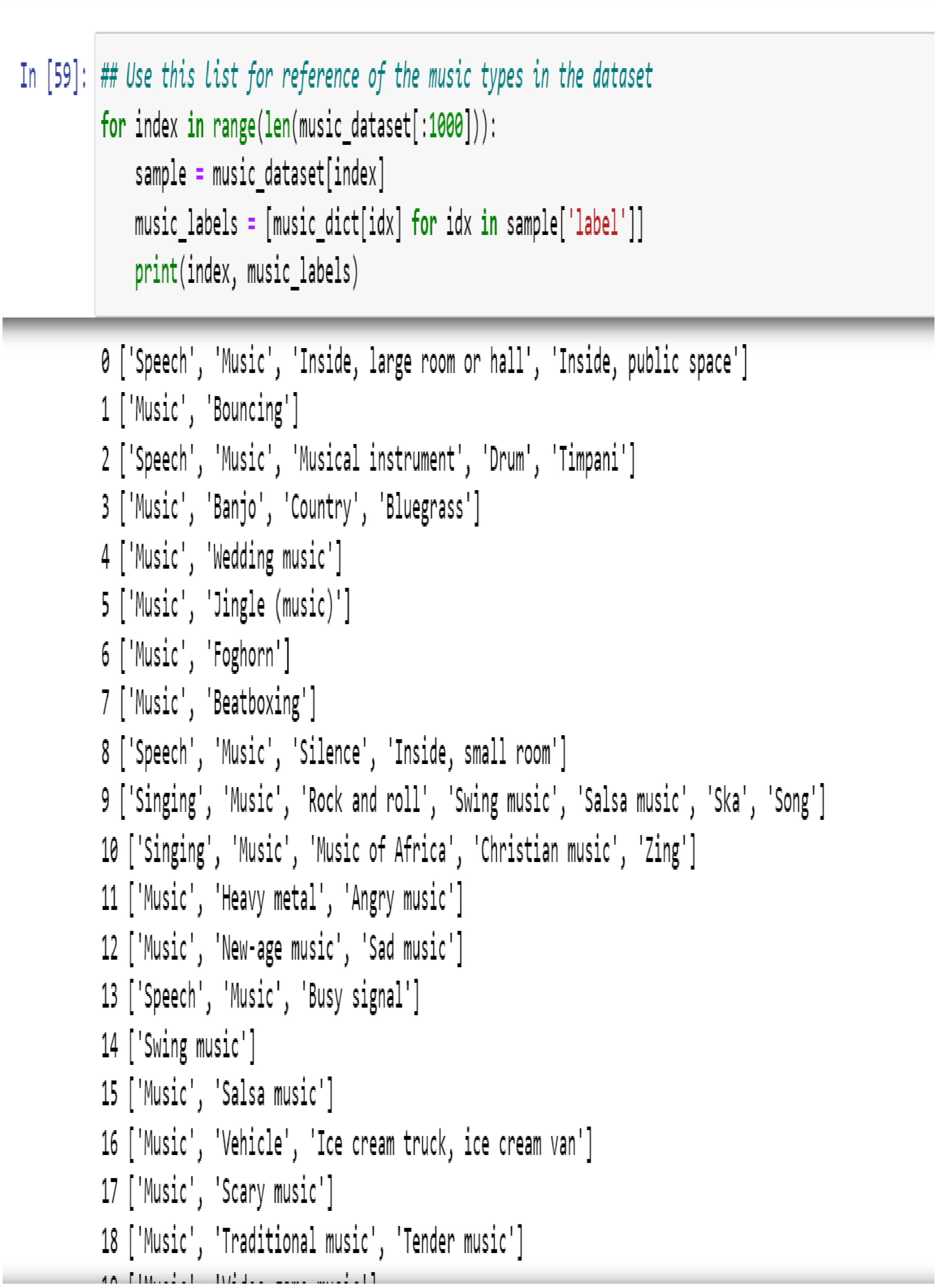
```
In [57]: class_labels = pd.read_csv('class_labels_indices.csv')
         music_dict = dict(zip(class_labels.index, class_labels.display_name))
```

```
In [4]: with open('music_set.json', 'r') as file:
            file_read = json.loads(file.read())
            music_dataset = literal_eval(file_read)
```

```
In [5]: len(music_dataset)

Out[5]: 2071
```

```
In [27]: from annoy import AnnoyIndex

         audio_dim = 1280
         annoy_index = AnnoyIndex(audio_dim, 'angular')  # Length of item vector that will be indexed
         for index in range(len(music_dataset[:1000])):
             vector = music_dataset[index]['data']
             annoy_index.add_item(index, vector)

         annoy_index.build(50) # 50 trees
         annoy_index.save('nearest_neightbor_graph.ann')

Out[27]: True
```

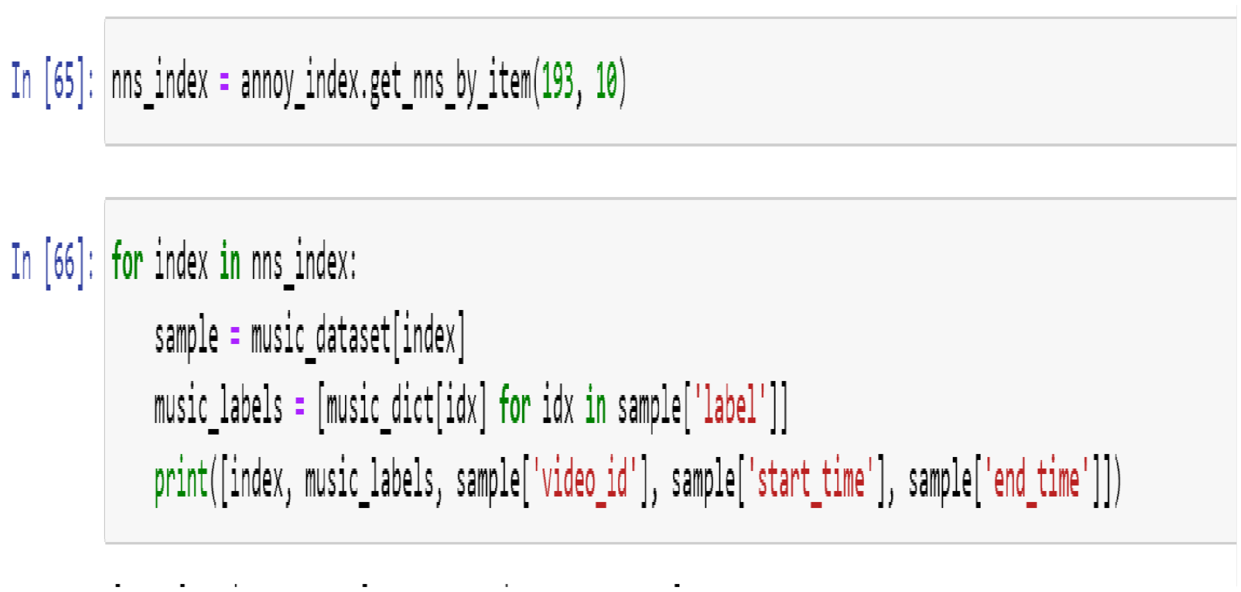
```
In [67]: annoy_index = AnnoyIndex(audio_dim, 'angular')
         annoy_index.load('nearest_neightbor_graph.ann')

Out[67]: True
```



```
In [59]: ## Use this list for reference of the music types in the dataset
         for index in range(len(music_dataset[:1000])):
             sample = music_dataset[index]
             music_labels = [music_dict[idx] for idx in sample['label']]
             print(index, music_labels)
```

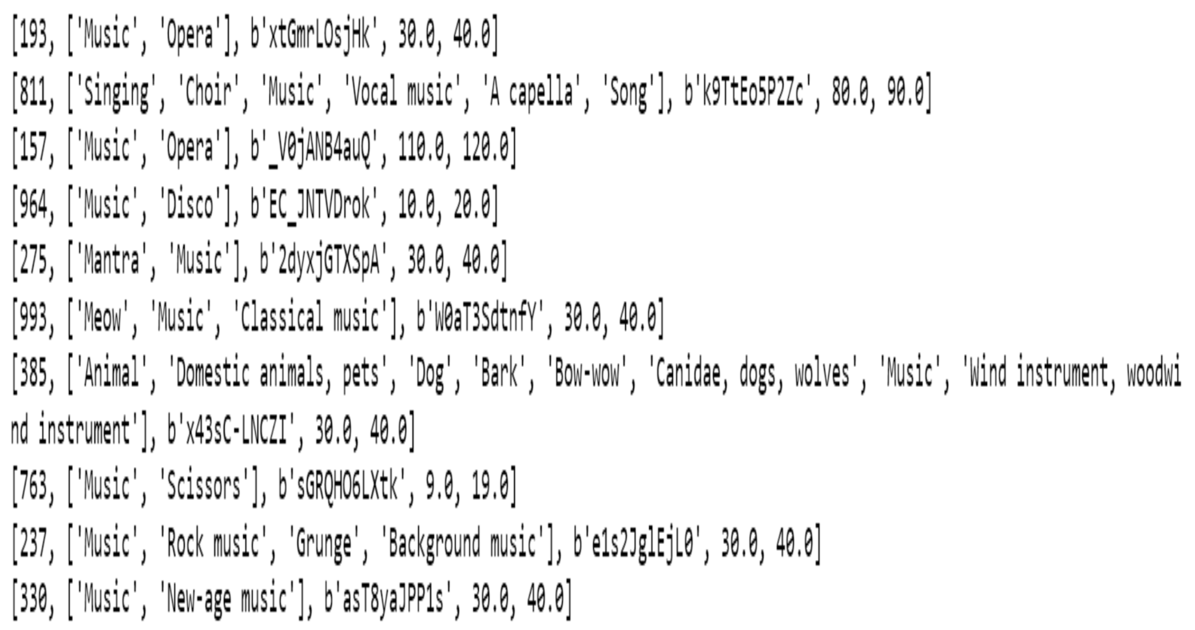
```
0 ['Speech', 'Music', 'Inside, large room or hall', 'Inside, public space']
1 ['Music', 'Bouncing']
2 ['Speech', 'Music', 'Musical instrument', 'Drum', 'Timpani']
3 ['Music', 'Banjo', 'Country', 'Bluegrass']
4 ['Music', 'Wedding music']
5 ['Music', 'Jingle (music)']
6 ['Music', 'Foghorn']
7 ['Music', 'Beatboxing']
8 ['Speech', 'Music', 'Silence', 'Inside, small room']
9 ['Singing', 'Music', 'Rock and roll', 'Swing music', 'Salsa music', 'Ska', 'Song']
10 ['Singing', 'Music', 'Music of Africa', 'Christian music', 'Zing']
11 ['Music', 'Heavy metal', 'Angry music']
12 ['Music', 'New-age music', 'Sad music']
13 ['Speech', 'Music', 'Busy signal']
14 ['Swing music']
15 ['Music', 'Salsa music']
16 ['Music', 'Vehicle', 'Ice cream truck, ice cream van']
17 ['Music', 'Scary music']
18 ['Music', 'Traditional music', 'Tender music']
19 ['Music', 'Video game music']
```

Asking for Recommendation for index -193:



```
In [65]: nns_index = annoy_index.get_nns_by_item(193, 10)

In [66]: for index in nns_index:
             sample = music_dataset[index]
             music_labels = [music_dict[idx] for idx in sample['label']]
             print([index, music_labels, sample['video_id'], sample['start_time'], sample['end_time']])
```

Output



```
[193, ['Music', 'Opera'], b'xtGmrLOsjHk', 30.0, 40.0]
[811, ['Singing', 'Choir', 'Music', 'Vocal music', 'A capella', 'Song'], b'k9TtEo5P2Zc', 80.0, 90.0]
[157, ['Music', 'Opera'], b'_V0jANB4auQ', 110.0, 120.0]
[964, ['Music', 'Disco'], b'EC_JNTVDrok', 10.0, 20.0]
[275, ['Mantra', 'Music'], b'2dyxjGTXSpA', 30.0, 40.0]
[993, ['Meow', 'Music', 'Classical music'], b'W0aT3SdtnfY', 30.0, 40.0]
[385, ['Animal', 'Domestic animals, pets', 'Dog', 'Bark', 'Bow-wow', 'Canidae, dogs, wolves', 'Music', 'Wind instrument, woodwi
nd instrument'], b'x43sC-LNCZI', 30.0, 40.0]
[763, ['Music', 'Scissors'], b'sGRQHO6LXtk', 9.0, 19.0]
[237, ['Music', 'Rock music', 'Grunge', 'Background music'], b'e1s2JglEjL0', 30.0, 40.0]
[330, ['Music', 'New-age music'], b'asT8yaJPP1s', 30.0, 40.0]
```

8. CONCLUSION

There are many more types and approaches to building out strong recommendation engines which are not covered in this project. Some notable ones are generating recommendations through link prediction, algorithmic approaches, Bayesian models, Markov models, etc.

This project was focused on efficiency and simplicity and it was a beginner-level recommendation system, as Music Recommendation System using Annoy Algorithm is easy to understand and very fast to use.

Also, DataSet and resources used in the model are freely available and open to everyone.

9. Future Scope

Machine Learning and Deep Learning have been all the rage the last couple of years in many different fields, and it appears that they are also helpful for solving recommendation system problems.

Ben Allison, a Principal Machine Learning Scientist at Amazon, gave a great talk earlier this year at Amazon's re: MARS conference about building recommender systems using Recurrent Neural Networks and Deep Learning.

Incorporating time into a recommender system is important because there are often preference seasonal effects. For example, it is likely that in December, more people are going to be watching holiday-themed movies and buying home decorations.

Another point that Ben Allison brought up is the need to see what would happen if a customer was shown a sub-optimal recommendation. This is taking a reinforcement learning approach since the goal, in this case, would be to show customers a recommendation, and then record what the customer does.

At times, customers can be recommended something that does not seem like the best option, just to see how the customer reacts which will improve the learning in the long-term.

Recommender systems can be a very powerful tool in a company's arsenal, and future developments are going to increase the business value even further. Some of the applications include being able to anticipate seasonal purchases based on recommendations, determine important purchases, and give better recommendations to customers which can increase retention and brand loyalty.

Most businesses will have some use for recommender systems, and we encourage everyone to learn more about this fascinating area.

10. REFERENCES

1. https://github.com/eunosalimullah/INT247-Project

2. https://www.slideshare.net/erikbern/approximate-nearest-neighbor-methods-and-vector-models-nyc-ml-meetup

11. SOURCES OF INFORMATION

1. Wikipedia
2. Analytics Vidhya
3. Google
4. YouTube
5. Class Notes (Dr. Sagar Pande)
6. SlideShare