DRAFT - CSC 454 Final Project

Dr. Christer Karlsson

CSC 454 - Data Mining Theory

May 3, 2022

Outlier Detection in Detecting Misclassified Music

There seems to be some logical relation between genre and the physical properties of music like tempo, danceability, tone, etc. Apart from that, there's instances where a certain piece of music has been classified as belonging to one genre but feel like they are outliers in that genre.

This project aims at using Outlier Detection Techniques to identify music that stands out of its genre - outliers that blend the lines between genres - and attempt to reclassify the music and provide a logical association between genre and other classifiers like danceability, tempo and so on.

The project achieves this by initially using statistical modeling and unsupervised learning methods like **k-means** and **Hierarchical Clustering** to separate 'far outliers' from the dataset. The 'cleaned dataset' will then be split into a training set and a test set and fed into an Artificial Neural Network (ANN) to recognize trends within a genre, thereby training it to recognize outlets. The projected hypothesis is that if the MLP-ANN is able to achieve an accuracy of 95% and above, it should be able to fairly recognize outliers and reclassify them; and hopefully match our intuition of where these outliers should belong.

Summary:

- Documentation and Visualization of the Dataset
- Developing Dataset
- Documentation of the Study and Results

Documentation of Dataset

My initial choice for the dataset was my own personal music listening history on the music streaming app Spotify. This data contained the following attributes:

- trackName = The name of the track streamed
- artistName = The name of the artist that produced the track
- endTime = The time when I ended the track
- msPlayed = The total time I played the track for

This however was not enough and more attributes like tempo, major, danceability, etc were needed to. Since my music history mostly contained music that was produced more recently, valid data corresponding to the above mentioned attributes was not available.

Therefore the main dataset was changed to something similar, obtained from kaggle.com. This new chosen data was obtained from this open source project - Music Genre Classification - and contains the following attributes

1) **Class -** The genre
2) **Artist Name -** String attribute containing the name of the artist
3) **Track Name -** Name of the track
4) **Popularity -** How popular the song is (metric between 0 and 100)
5) **Danceability -** How easy it is to dance to the trace (metric between 0 and 1)
6) **Energy -** How lively a song is (metric between 0 and 1)
7) **Key -** The key in which the song is (metric between 0 to 11 corresponding to C - B)
8) **Loudness -** How loud the music is (metric in dB)
9) **Mode -** Corresponding to the musical definition of mode (binary)
10) **Speechiness -** Measure of the the presence of spoken words in a track (0 to 1)
11) **Acousticness -** Measure of how acoustic a track is (metric between 0 and 1)
12) **Instrumentalness -** Measure of how much of a track is just instrumentals (0 to 1)
13) **Liveness -** Measure of reverberation time (between 0 and 1)
14) **Valence -** Measure from 0 - 1 describing the musical positiveness of a track.
15) **Tempo -** Beats per minute of a track
16) **Duration -** Time duration of the track in ms
17) **Time Signature -** The musical time signature (3/4 or 4/4)

The **training dataset** contains **17,996 entries** with **17 attributes**, while the testing dataset contains **7,713 entries with 17 attributes** (show above). The datasets can be found at this link - datasets.

The target variable, **Class**, which represents the genre contained values from [0, 10] which corresponded to the following Genre(s):

0 = Acoustic/Folk

1 = Alt Music

2 = Blues

3 = Bollywood

4 = Country

5 = HipHop

6 = Indie Alt

7 = Instrumental

8 = Metal

9 = Pop

10 = Rock

Ethical and Privacy Concerns :- The dataset was obtained from an open source data analysis project from **kaggle.com**

Figures and Visualization of Data

Figures 1 - 4 show Box Plots of the data corresponding to attributes that are easily perceived aurally or physically.

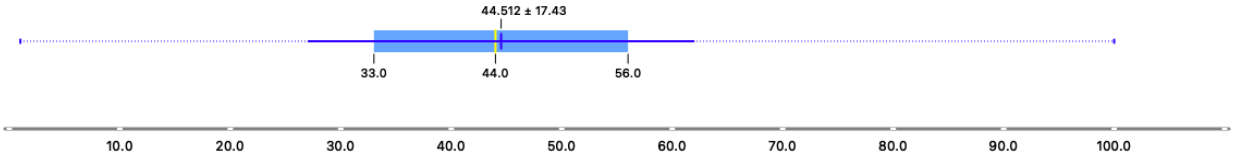

Figure 1: Popularity

From Figure 1 and Figure 2, we can infer that most of the data chosen is not super popular and includes a wide selection with high outliers.

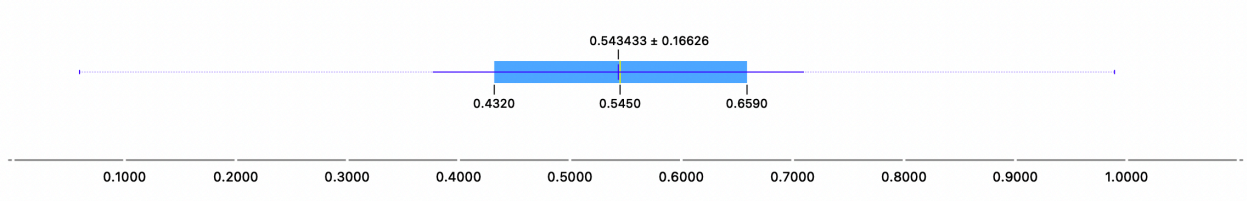

Figure 2: Danceability

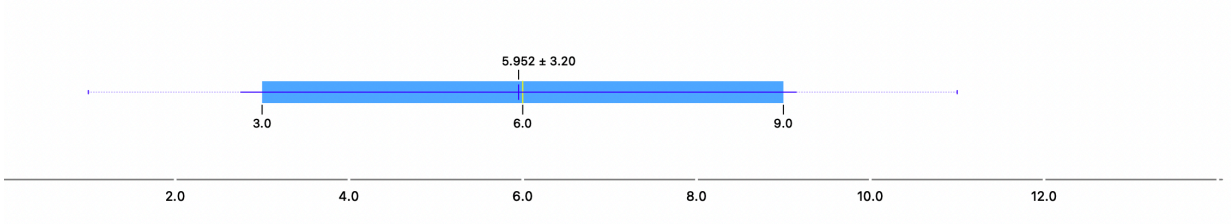

Figure 3: Key

One can infer from Figure 3 that most of the data chose has songs written in the F major scale.

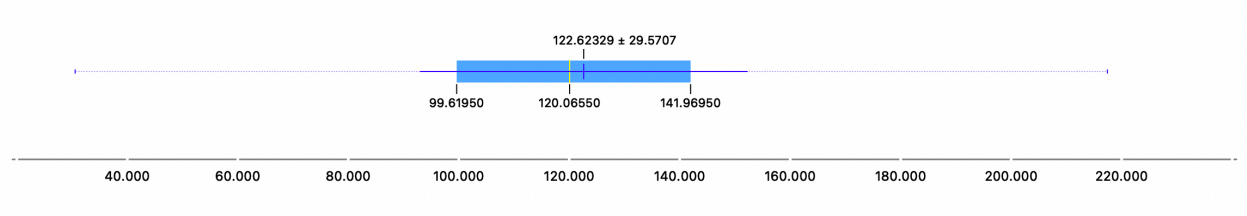

Figure 4: Tempo

From the Box Plot in Figure 4, one can see that most of the music data revolves around fast paced songs.

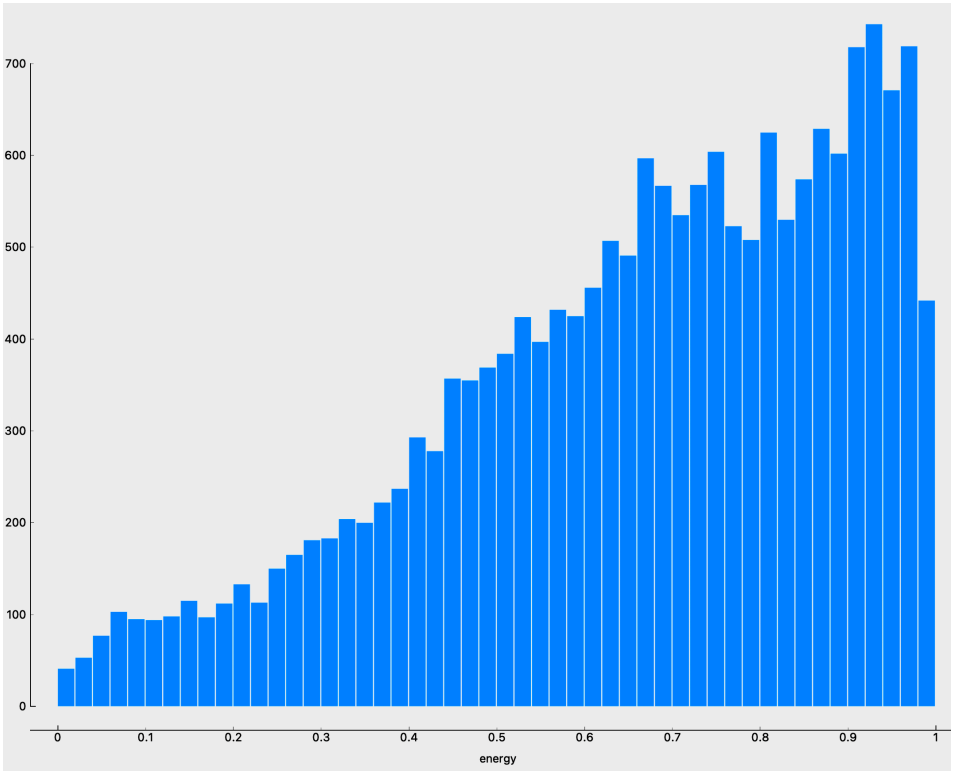

Figure 5. Energy

Figures 5 - 13 show histograms of all the other attributes:

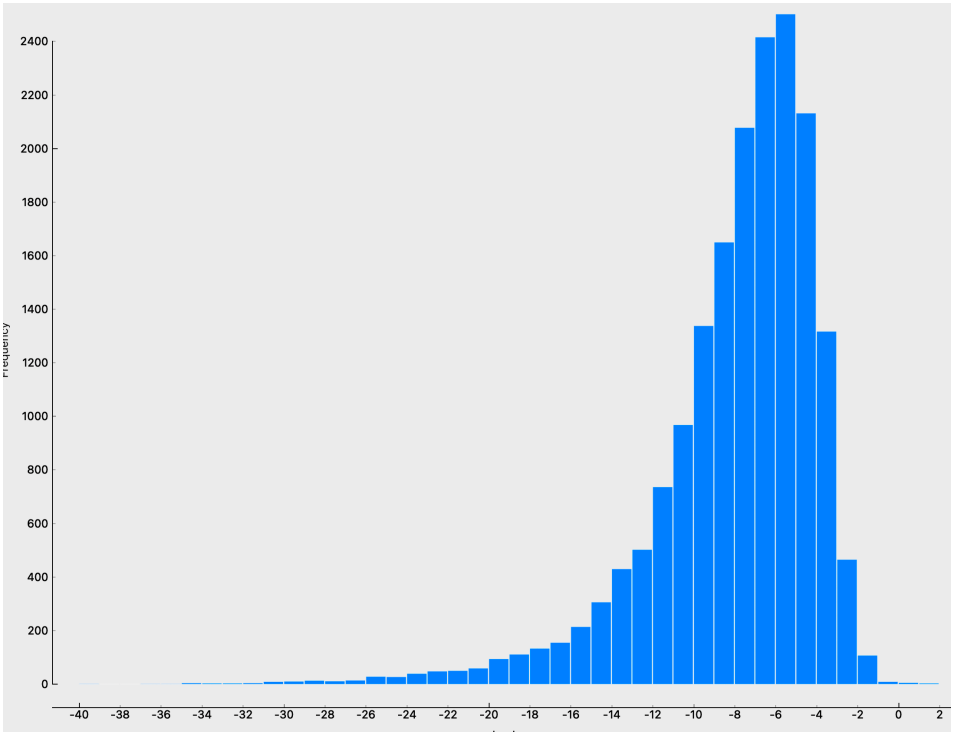

Fig 6: Loudness

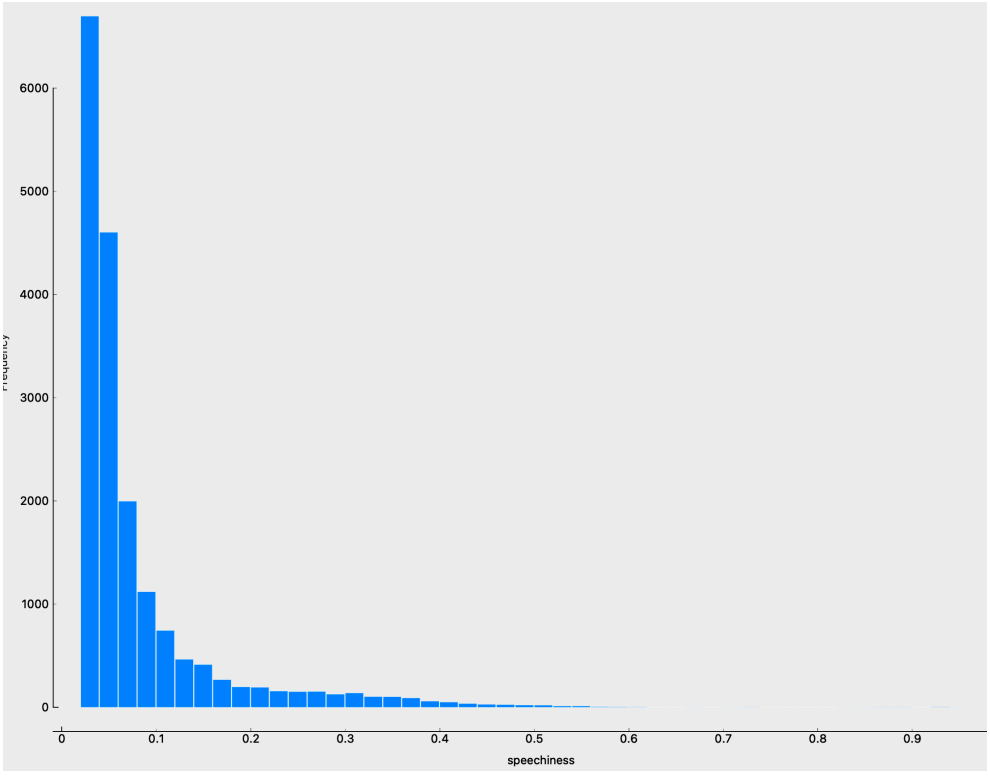

Fig 7: Speechiness

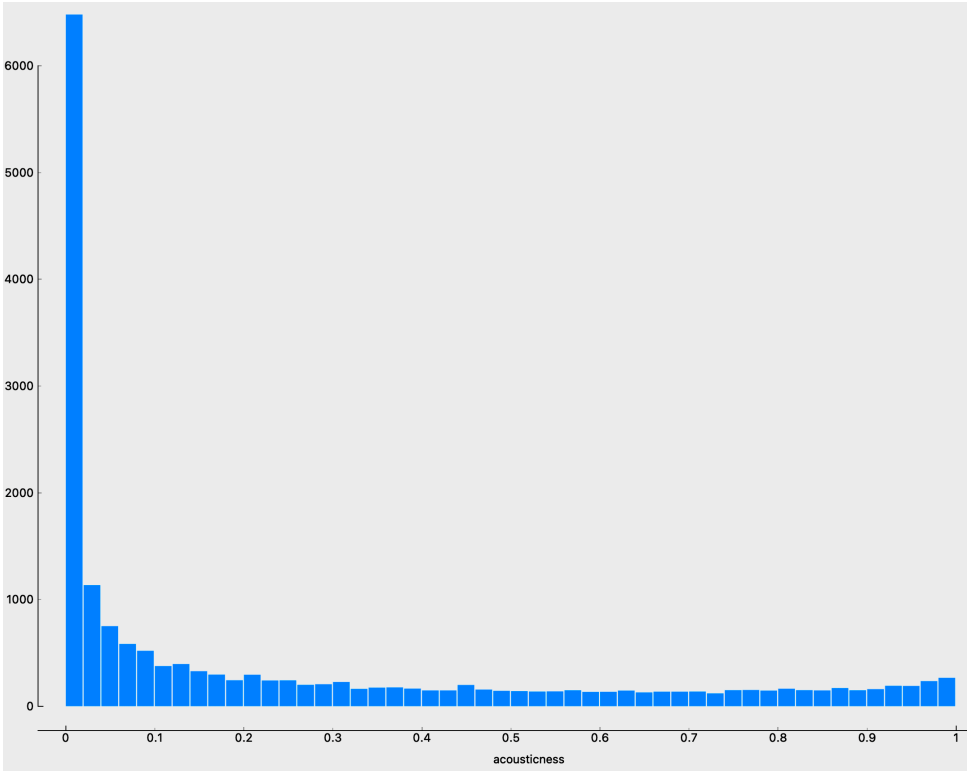

Fig 9: Acousticness

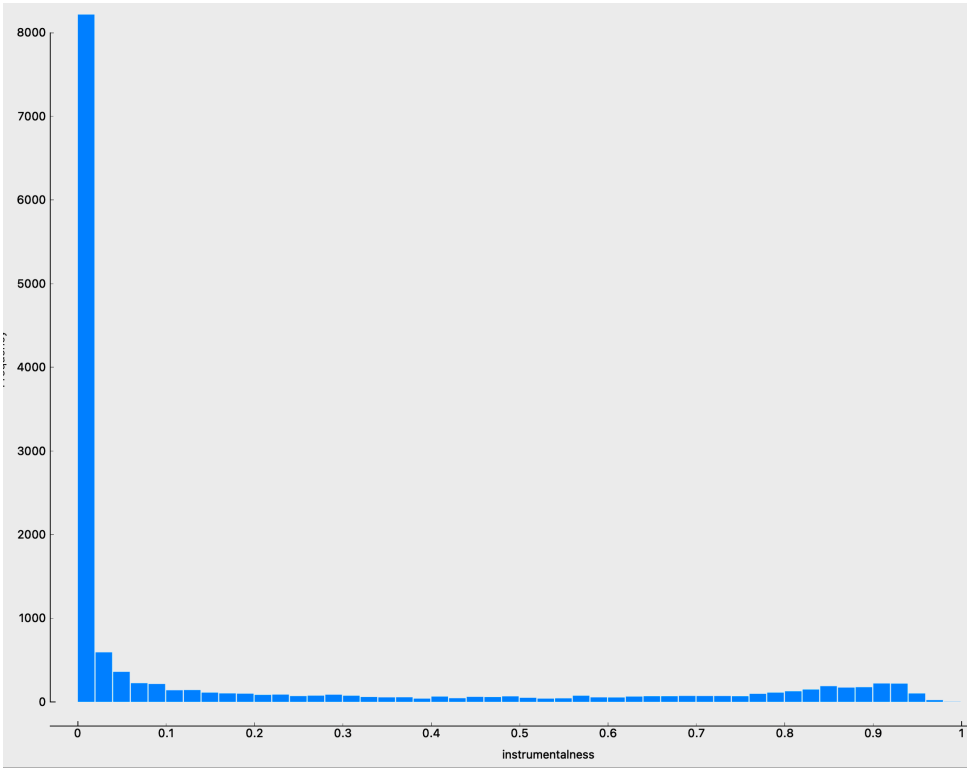

Fig 10: Liveness

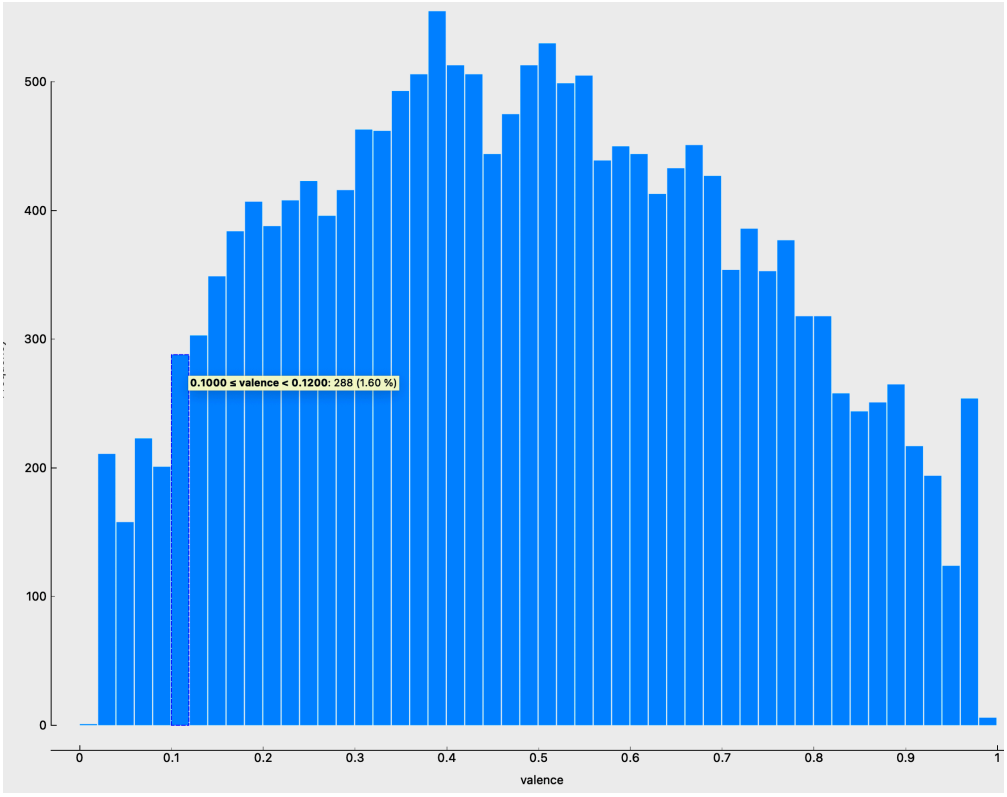

Fig 11: Valence

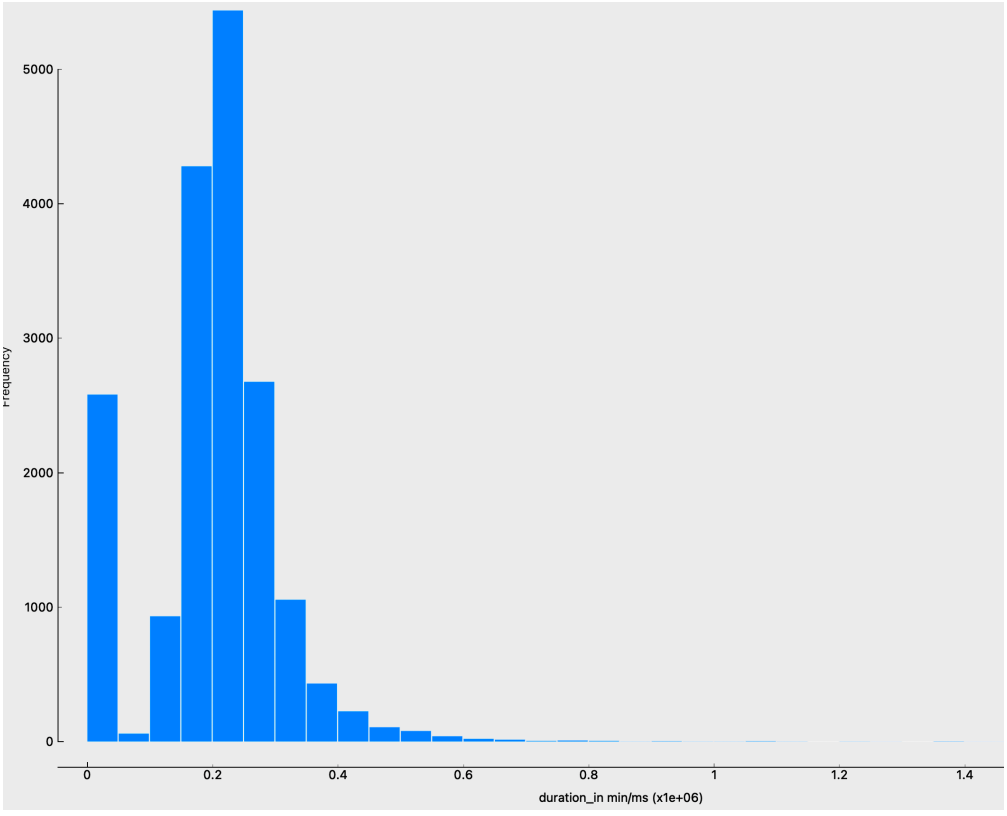

Fig 12: duration

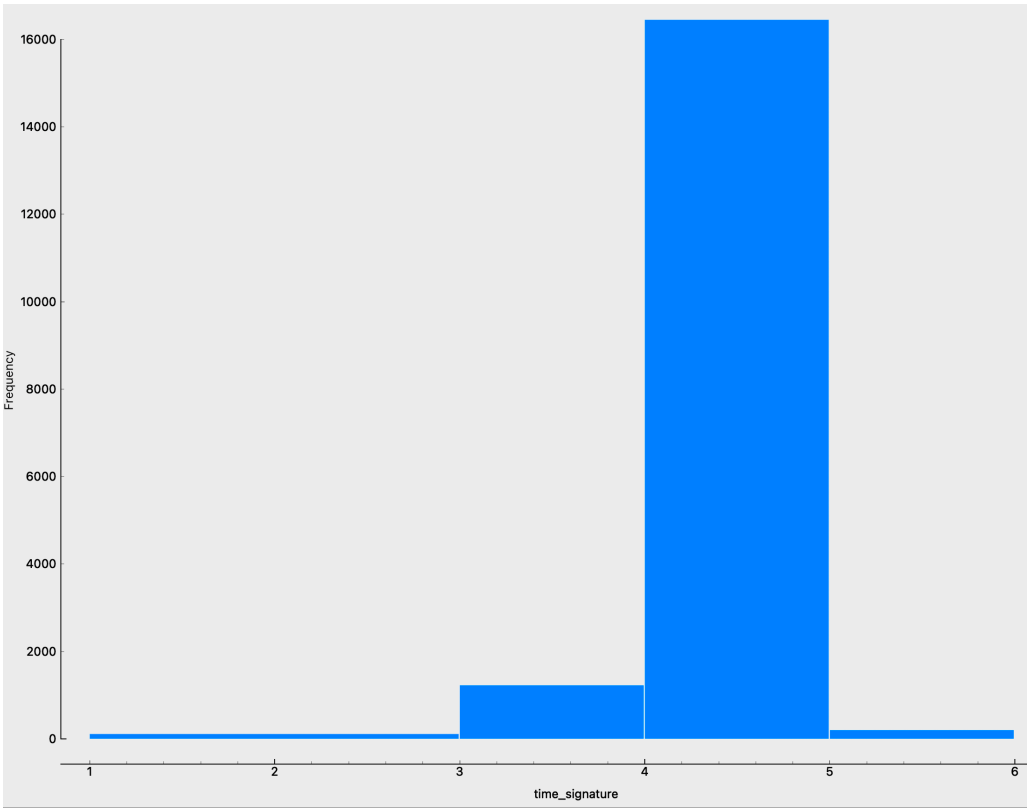

Figure 13: Time Signature

Developing the Dataset

The dataset contained text fields like the Artists Name and the Track Name. Text Fields, unless binary barely contribute to a dataset. Hence these were ignored by skipping them in File Widget.

In addition to removing unwanted features, the dataset had instances with incomplete/ missing attributes. These elements were purged using the impute widget with the settings shown below in Figure 14.

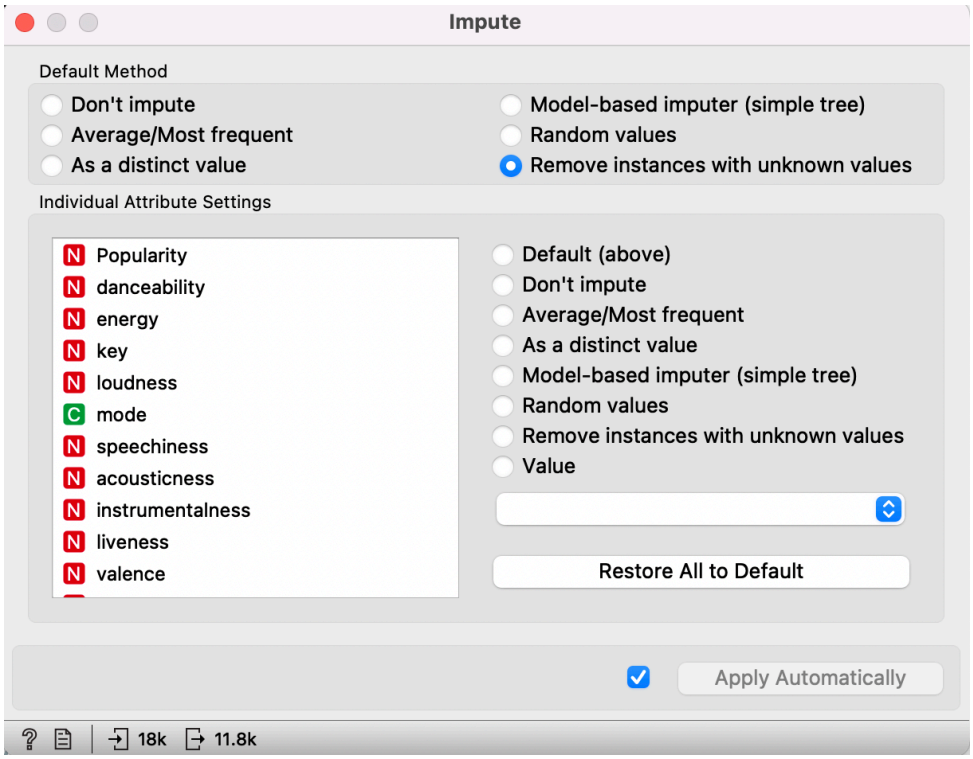

Fig 14. Imputing Missing Elements

This resulted in the dataset being reduced from 18k instances to 11.8k instances. The dataset needs to be preprocessed and normalized before any analysis considering supervised unsupervised learning is performed on it. The aim is to normalize each attribute with a Mean of 0 and a Variation of 1.

The task of Normalization presented itself with two approaches:

- Normalize the dataset as a whole (ignoring Genre or the Class Attribute)
- Split the dataset by Genre and Normalize within a Genre

I went with the latter due to the following reasons:

- The aim of this project is to find music that doesn't belong to its genre, in other words, music thats considered an outlier within its Genre.
- Normalizing over the entire dataset and then using an unsupervised learning algorithm would separate music from non music, for example, animalistic noises from regular music, if any existed.

The Row Selector widget was used to split up data by Genre and then, normalized using the Preprocessor widget giving rise to an Orange Workspace model as shown on the next page (Figure 15).

The next step was is to use an Unsupervised Learning Algorithm like K-Means or hierarchical Clustering to separate the minorities within a class from the majority, when distinct clusters are available. These minorities will be pooled together to form the Outliers dataset. An element in the outliers dataset need not necessarily be an outlier as per our intuitive definition; at this stage it just deviates from the majority norm that defines the genre. In other words this step is just used to separate music that is similar to each other from the outliers within a genre or class.

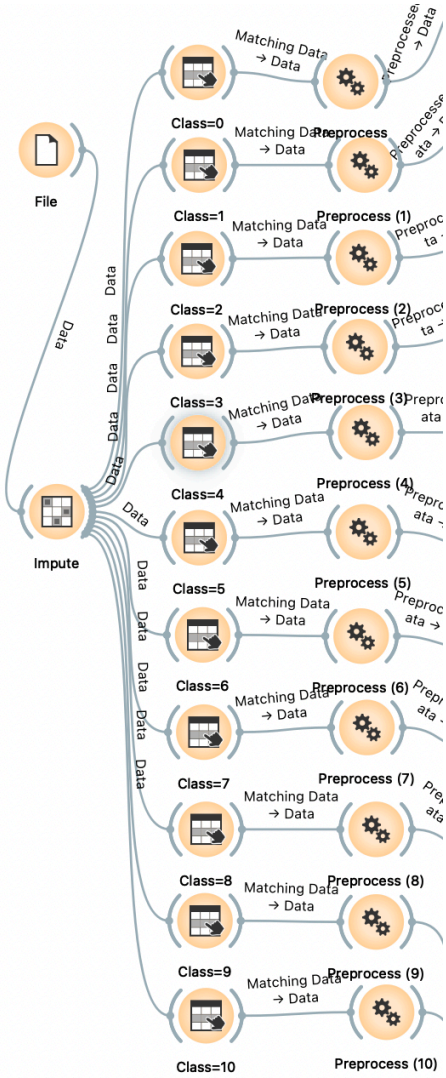

Figure 15. Normalized Data by Genre

K-means was chosen over Hierarchical Clustering as it produced better results. The following were the results obtained per class or genre:

Class 0 = Could not be split into clusters effectively. 386 elements in inlier.

Class 1 = 1037 elements in total. 888 elements in inlier, 149 in outliers

Class 2 = Could not be split into clusters effectively. 956 elements in inlier.

Class 3 = Could not be split into clusters effectively. 270 elements in inlier.

Class 4 = 148 elements in total. 138 elements in inlier, 10 in outlier.

Class 5 = 517 elements in total. 512 inliers, 5 outliers.

Class 6 = 2039 elements in total. 1528 inliers and 511 outliers

Class 7 = 464 elements in total. 354 inliers and 110 outliers.

Class 8 = 1523 elements. 1270 inliers and 253 outliers.

Class 9 = 1099 elements. 1046 inliers and 53 outliers.

Class 10 = 3374 elements. Could not be split up effectively.

The outliers from each class were pooled together and saved into a .csv file with the name Outliers.

The inliers, coming out from each class was connected to the Data Sampler Widget from which **70%** of the data in each class was sampled and pooled together to form the **training set** while the remaining **30%** formed the **test set,** as shown below in Figure 16.

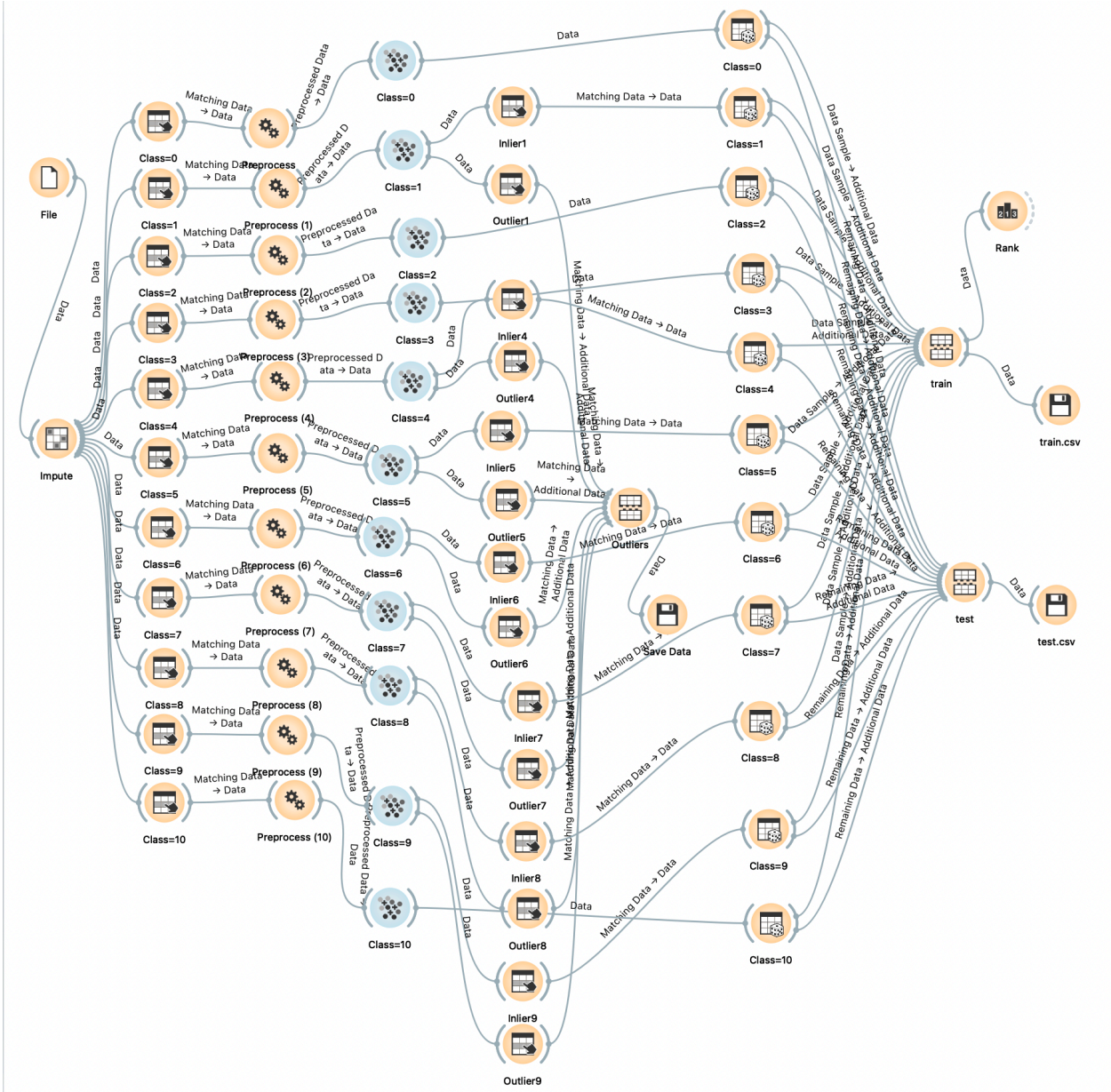

Figure 16. Overall Structure

Documentation of Study and Results

The initial aim of this project was to use Unsupervised Learning Algorithms like DB-SCAN, K-Means or Hierarchical Clustering to find differences between genres. On implementing those models, the results proved inconclusive and there were no real visible patterns that separated Genres based on the 13 other attributes.

I decided to shift to Supervised Learning and implement models like Random Forest, Naive Bayes, AdaBoost and Artificial Neural Networks. **The hypothesis was that if one or more of the Learners achieved a 90% accuracy on the test dataset, it would be able to identify which music truly belongs to a Genre and which does not.** The results obtained from each Learner is shown below.

Artificial Neural Networks

I implemented two of my own Artificial Intelligence Models for this section:

- Multilayer Perceptron with Backpropogation
- Convolutional Neural Network

The code for both these models along with the best learned models can be found at this link: MLP & CNN code. The *.sav* files are the json save states of models that achieved the best accuracy.

The Multilayer Perceptron with Backpropogation's ended up stabilizing at around 1.9 with the best model exhibiting the following accuracy shown in Figure 17.

```
(base) sherwyn_b@Sherwyns-MacBook-Air csc454_final_project % python3 mlp_tester
py

Number Of Images Tested = 3212
Model Accuracy = 55.63511830635118 %
(base) sherwyn_b@Sherwyns-MacBook-Air csc454_final_project %
```

Figure 17. Testing Result of MLP

I compared this result with the Model created using the Neural Network Widget in Orange and they achieved similar accuracies (shown below Figure 19).

The CNN model on the other hand achieved the following results with the Cross Entropy stabilizing at around 2.1.

```
(base) sherwyn_b@Sherwyns-MacBook-Air csc454_final_project % python3 cnn_tester.
py

Number Of Images Tested = 3212
Model Accuracy = 76.70545546476095 %
(base) sherwyn_b@Sherwyns-MacBook-Air csc454_final_project %
```

Figure 18. CNN Model Testing Results

This however is way below our threshold for our hypothesis to hold. Hence this model was discarded along with the MLP Model.

What intrigued me a little was the significantly higher precision achieved by the CNN network which signifies relations between attributes, which was assumed to be independent.

Naive Bayes Classifier

The Naive Bayes Classifier did not achieve results that passed our accuracy threshold of 90% (shown in Figure 19). This model was discarded too.

Model	AUC ▼	CA	F1	Precision	Recall
AdaBoost	0.996	0.993	0.993	0.993	0.993
Random Forest	0.995	0.945	0.943	0.944	0.945
Naive Bayes	0.972	0.743	0.733	0.733	0.743
Neural Network	0.876	0.595	0.560	0.562	0.595

Fig 19. Accuracies of Learning Models

Random Forest Classifier

The Random Forest Classifier Achieved a much higher accuracy the the aforementioned two. It achieved an accuracy of 94.5% which passes our threshold. The confusion matrix corresponding to this can be found in Figure 20. The Random Forest seemed to struggle with classifying items in Classes 2, 3, 5 and 7.

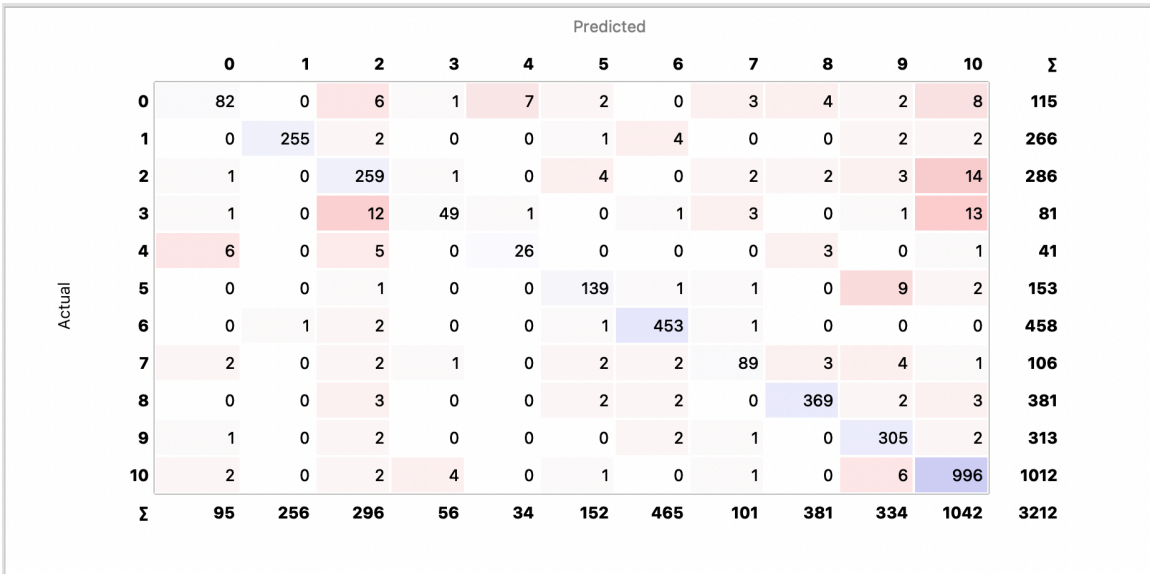

Fig 20. Random Forest Confusion Matrix

AdaBoost Classifier

The AdaBoost Classifier achieved the highest accuracy among the tested classifiers (Fig 19.) It achieved a 99.3% accuracy with a 99.3% precision. The confusion matrix corresponding to this model is shown below (Fig 21).

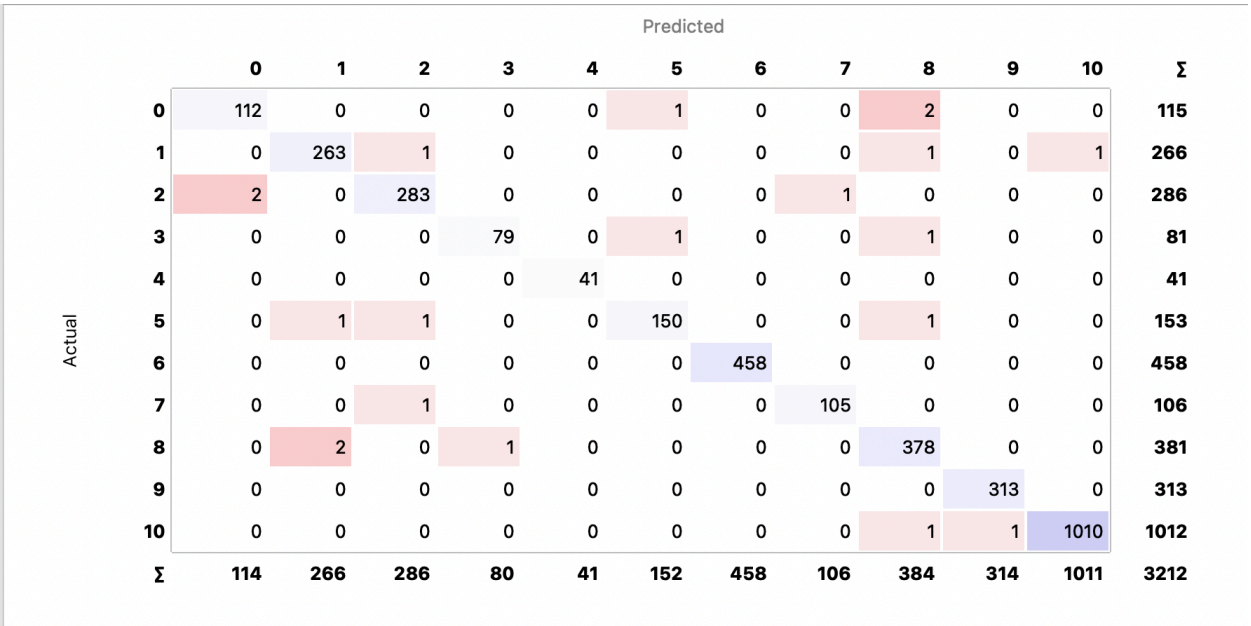

Fig 21. AdaBoost Confusion Matrix

Testing on Outlier Set

Using the AdaBoost Model, it was connected to a Prediction Widget and was used to reclassify our Outlier.csv dataset. On analysis of the results, it was found that most of the Outliers data that we obtained from unsupervised learning methods were not actually outliers and the AdaBoost Model classified them into their listed Genres. This was for most of the dataset except for a few cases that were true outliers, like the ones (show below in Figure 22) which matches our intuition.

159	4	Don Williams	I Believe In Y...	C1	0.56292	9
160	5	Billie Holiday	Strange Fruit	C3	0.600782	1

Conclusion

- Most data classified as outliers by the K-means algorithm didn't turn out to be outliers.
- The AdaBoost Supervised Learning Algorithm is the best suited algorithm for classifying music.
- The Trained AdaBoost Learning Algorithm can be used to find outliers in Music Genres.

Graduate Part - Implementing a Widget in Orange

I am working on implementing a CNN model in Orange. The code for the model can be found in at this link : CNN_model.

Unfortunately, while working on importing a test widget and trying to implement my CNN model, I ended up breaking my version of orange and my project and couldn't figure out how reverse the damage.