CS-UY 4563: Machine Learning

Final Project Written Report

Genre Classification: Audio and Feature Engineering

2:00 PM Section

April 27, 2023

Professor Linda N. Sellie

David Chang, Denizhan Ene

1. Introduction

1.1 Project Introduction

This project focuses on using machine learning techniques to predict the genre of a sound sample. Neural networks, logistic regression, and support vector machines were used to classify music samples from out of 10 categories. All models were run multiple times using different parameters and regularization techniques to identify the optimal model.

1.2 Dataset

Our data set is the GTZAN dataset which is the most-used public data set for Music Genre Recognition. The dataset was collected during the 2000-2001 year from a variety of sources including microphone recordings, plaque records, CD's, radio etc. By using various supervised machine learning models on 58 features extracted from sounds, the goal of this study is to accurately predict the genre labels of music from unknown sounds and compare their performance and accuracy.

While a superficial analysis of a music sample would be to look at the loudness (amplitude), pitch, and length of the piece, Python libraries allow for a more holistic understanding of the audio data itself. Next, we'll define some attributes of sound that are less understood that are used in the dataset to better understand the samples we're working with.

1.3 Key Terms

Chroma short-term Fourier Transform (STFT): The chroma of an audio represents the intensity of each distinct pitch class at any given time. Simply put, this represents how much each pitch is used in a sample. The mean and variance for this continuous data is taken and used as features.

Root Mean Squared(RMS): The average RMS of an audio represents the loudness of an audio sample. It provides an objective measure of sound loudness for the sample.

Spectral Centroid: An aspect of music timbre, the spectral centroid provides a quantified value of "brightness" in the sample. Higher-frequency content in the sound contributes more to this value than lower frequencies.

Spectral Bandwidth: Derived from the spectral centroid, this represents the difference between upper and lower frequencies, the variance from the spectral centroid.

Mel-frequency Cepstrum Coefficient (MFCC): Used often in speech recognition systems, the coefficients represent the overall shape of the spectrum.

The dataset includes a CSV file, it contains 1000 samples of 30 seconds of sounds with labels. It contains 58 features, and all the sounds belong to 1 of 10 genres which are: *blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, rock*.

2. Data Preparation

2.1 Data Cleaning

From the dataset csv, we recognized there was plenty of metadata that wasn't needed for the model and only serves as an extra feature contributing nothing. These features, the "filename" and "length" of the sample were not needed for the training. Thus, we removed them by dropping them from the dataframe. There were no null or invalid values in the dataset.

2.2 Data Scaling

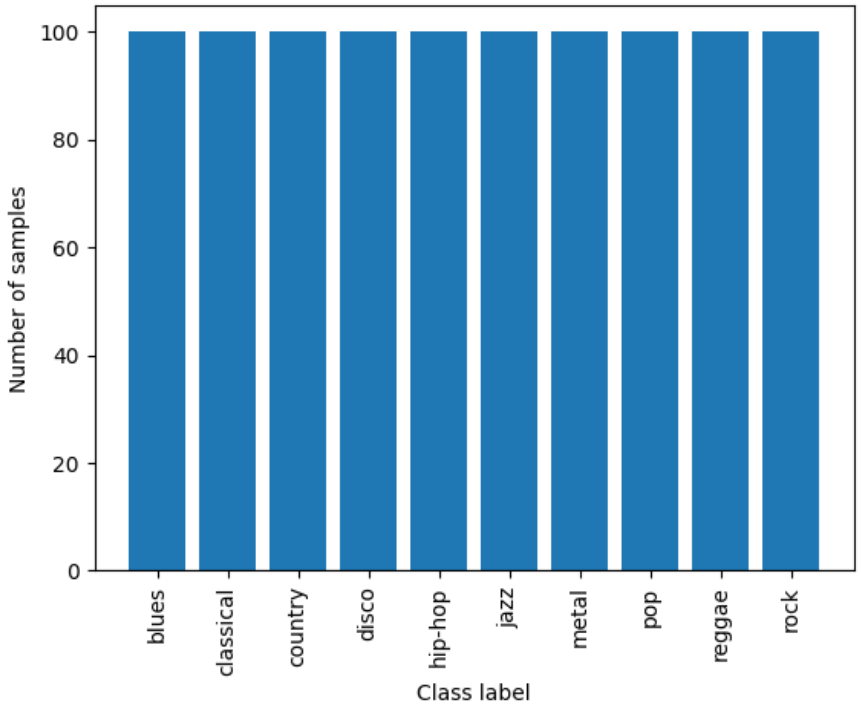
The scikit-learn preprocessing library was used to scale the dataset by normalizing the features to a zero-centered mean and unit variance. This would in turn improve the performance of the model and inadvertently help reduce overfitting and underfitting, through being less sensitive to outliers.

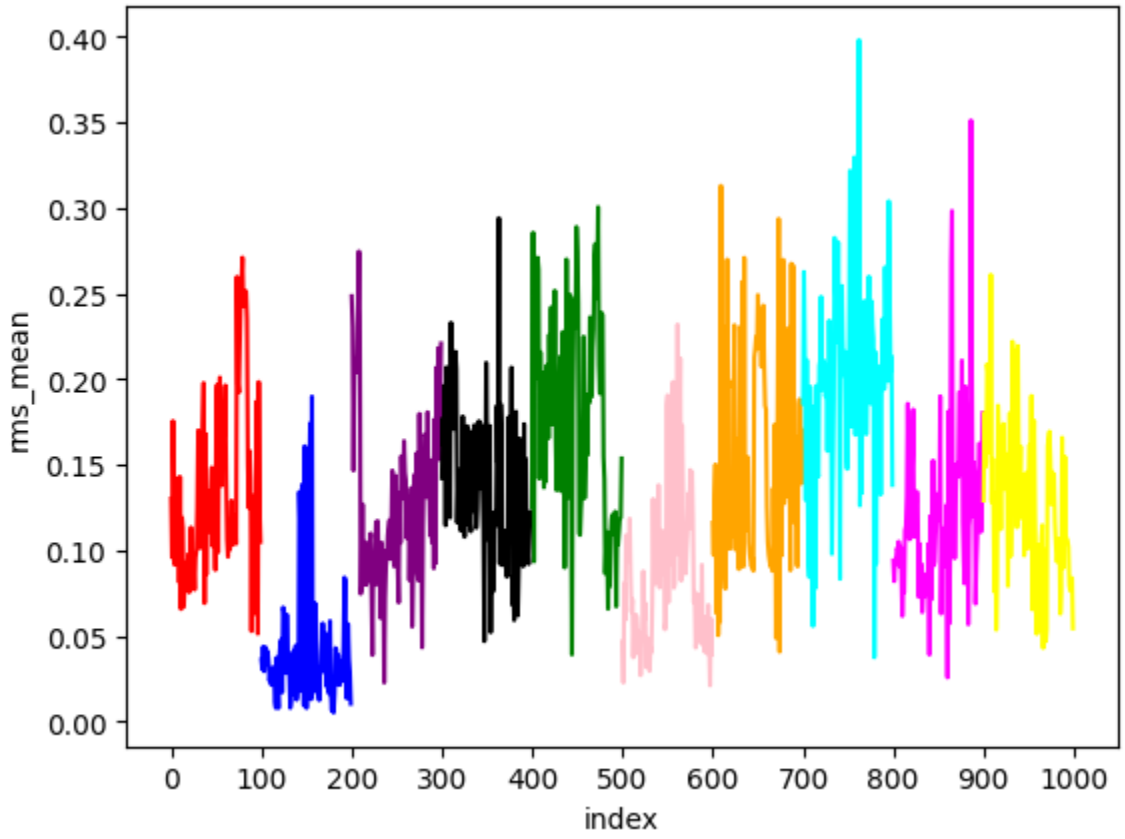
2.3 Data Splitting

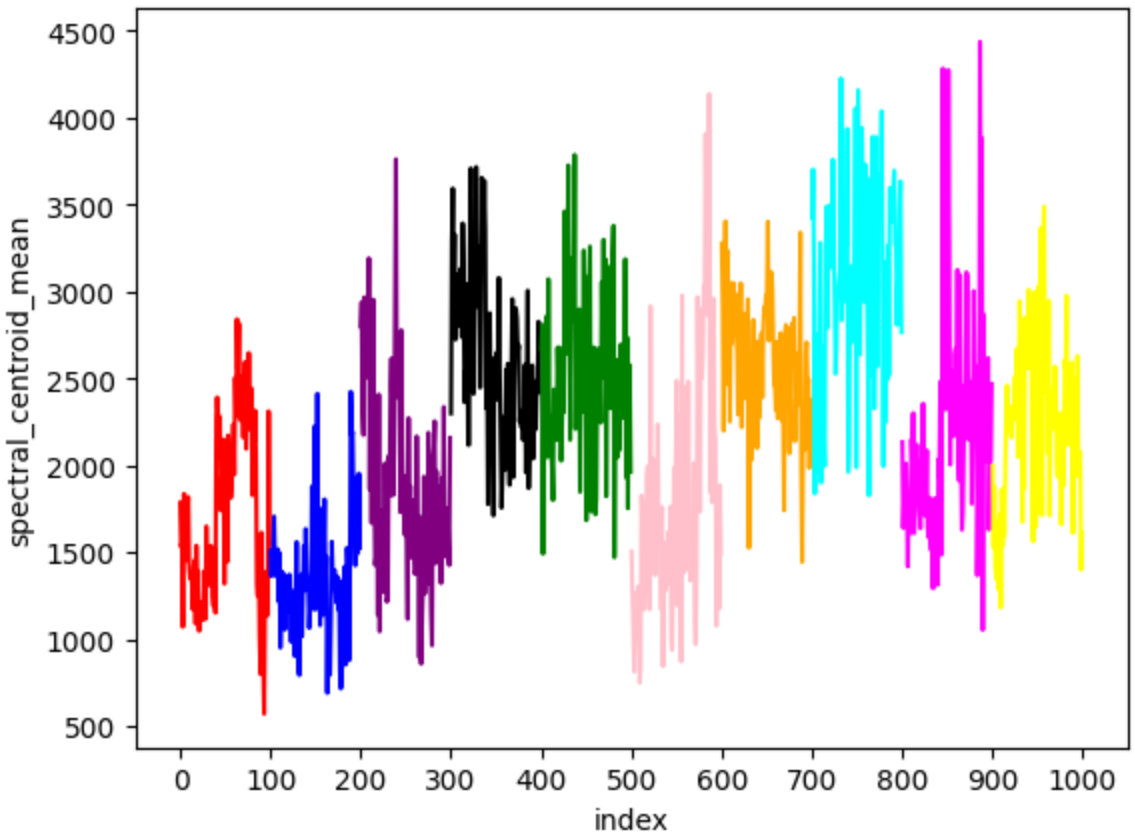
The data was split into a training set and a testing set. 80% of the data was allocated to training while 20% was allocated to testing. This was done using the train_test_split() function from the sklearn library. The random_state parameter was kept at '42' to ensure consistent values for every iteration.

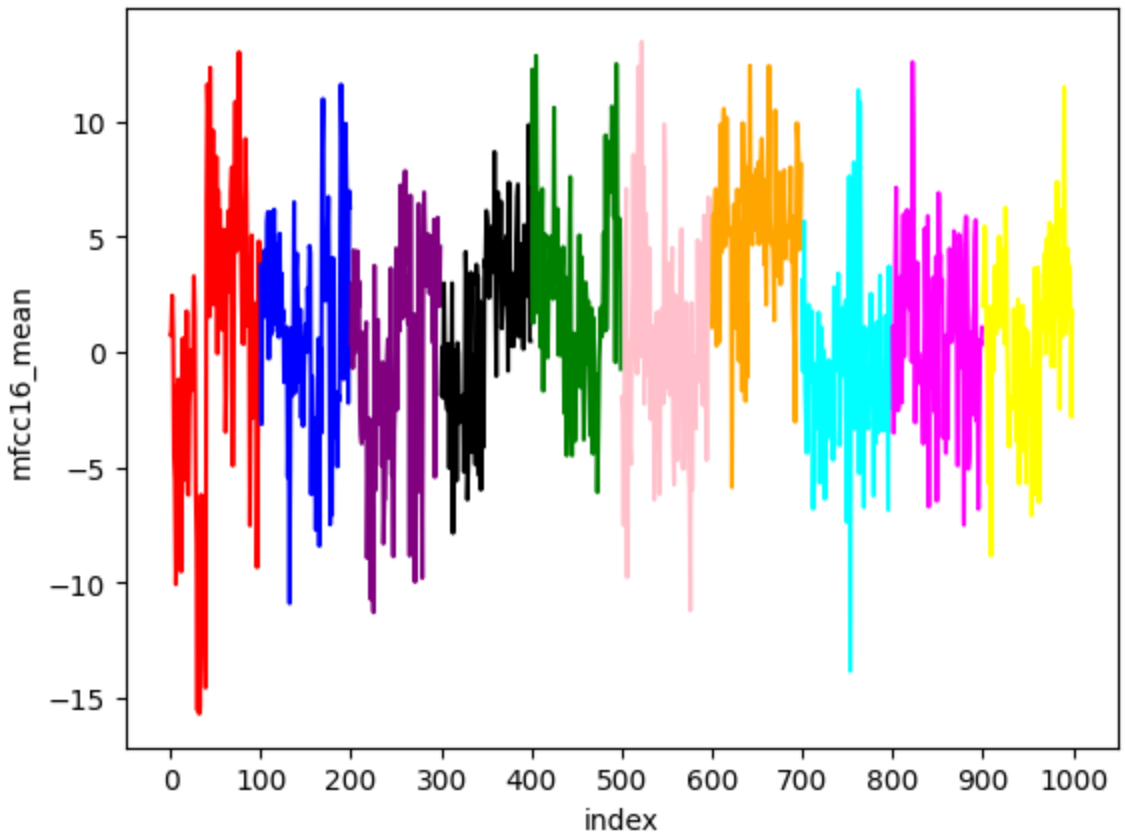
2.4 Understanding

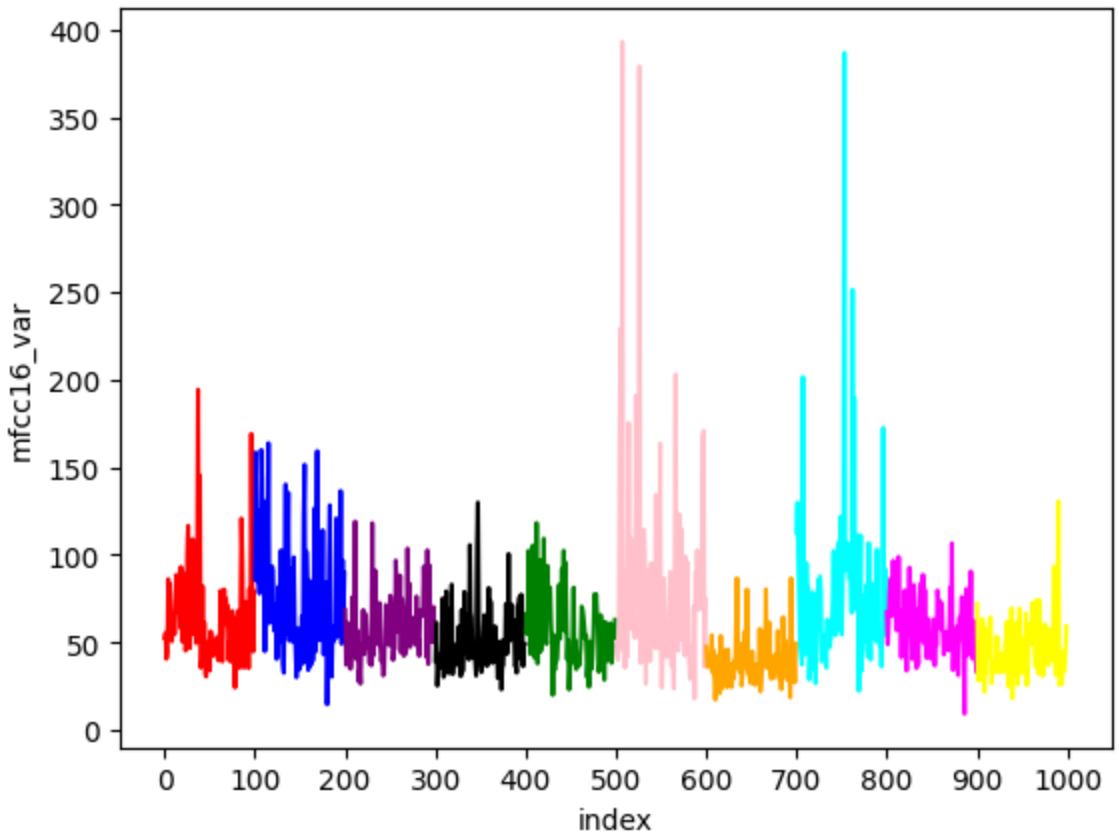
There are 100 samples from each class. Each sample has 58 columns of recorded data.


Number of samples
100
80
60
40
20
0
blues
classical
country
disco
hip-hop
jazz
metal
pop
reggae
rock
Class label


rms_mean
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
100
200
300
400
500
600
700
800
900
1000
index


spectral_centroid_mean
4500
4000
3500
3000
2500
2000
1500
1000
500
0
100
200
300
400
500
600
700
800
900
1000
index


mfcc16_mean
10
5
0
−5
−10
−15
0
100
200
300
400
500
600
700
800
900
1000
index



The graphs shown above represent the data values for each of the corresponding labels.

blues, = red

classical, = blue

country, = purple

disco, = black

hip-hop, = green

jazz, = pink

metal, = orange

pop, = cyan

reggae, = magenta

rock = yellow

3. Models

3.1 Neural Network

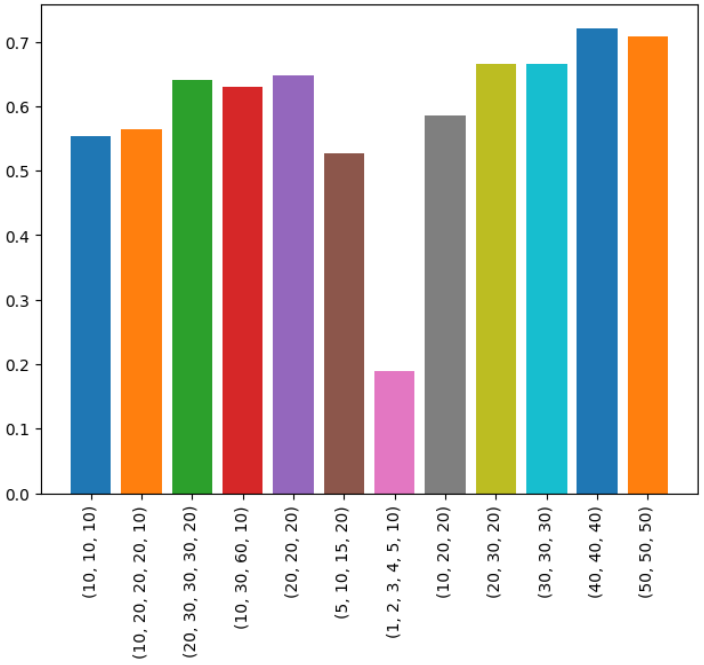
The first model we decided to use was neural networks. The intuition was comparing the problem to a 10 number classification problem where the target vector was the likelihood that a sample would be a certain label.

We also utilized the sklearn GridSearchCV library to run through multiple hyperparameters to search multiple parameter spaces. The following are the parameters we decided to see if either would improve the model. This model uses K-fold validation to determine the score of a function. By default, it sets k equal to 5. Our neural network will be using 5 fold validation to try to find the optimal parameters.

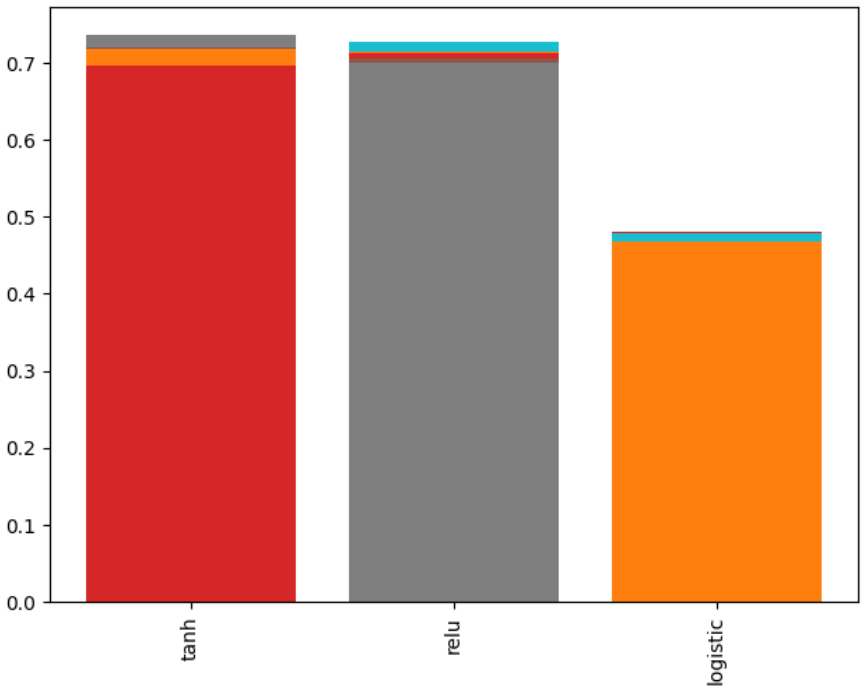
```
parameter_space={

    'hidden_layer_sizes':
[(10,10,10),(10,20,20,20,10),(20,30,30,30,20),(10,30,60,10), (20,20,20),
(5,10,15,20),(1,2,3,4,5,10),(10,20,20),(20,30,20),(30,30,30),(40,40,40),
(50,50,50)],

    'activation':['tanh','relu','logistic'],

    'solver':['sgd','adam'],

    'learning_rate':['constant','adaptive'],

    'alpha': [0.01,0.005,0.001]

}
```

The following are the graphs generated by matplotlib of each parameter and its effect on the mean_test_score. The higher the score, the better it performed on the validation sets within each K-fold model.

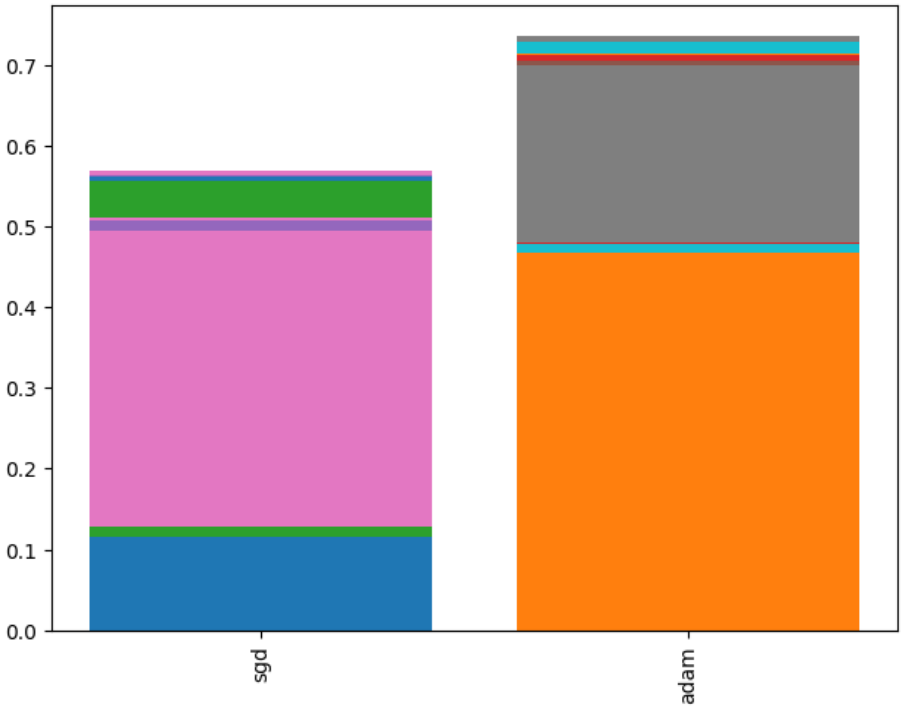
Hidden Layer Size



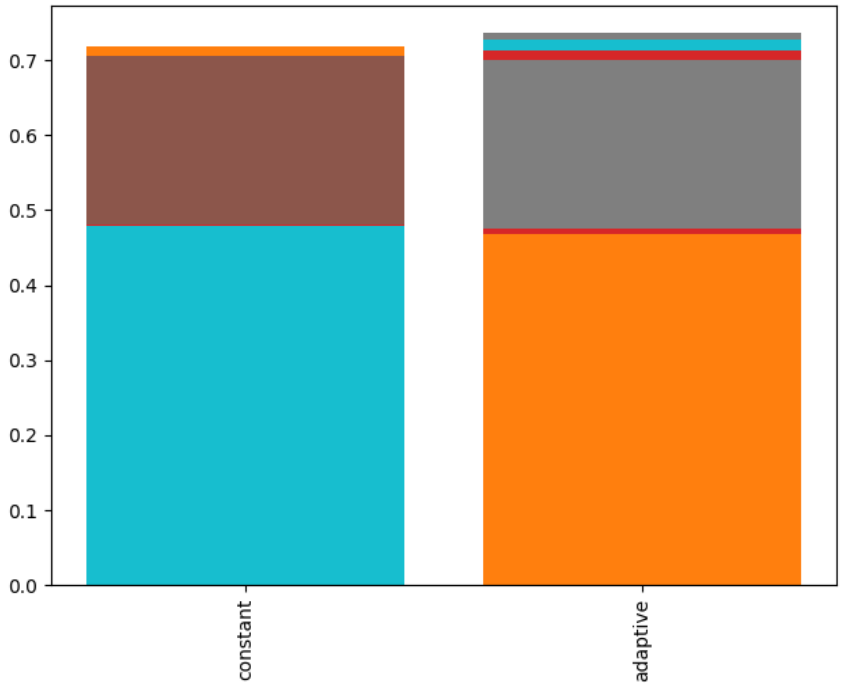
Activation Function



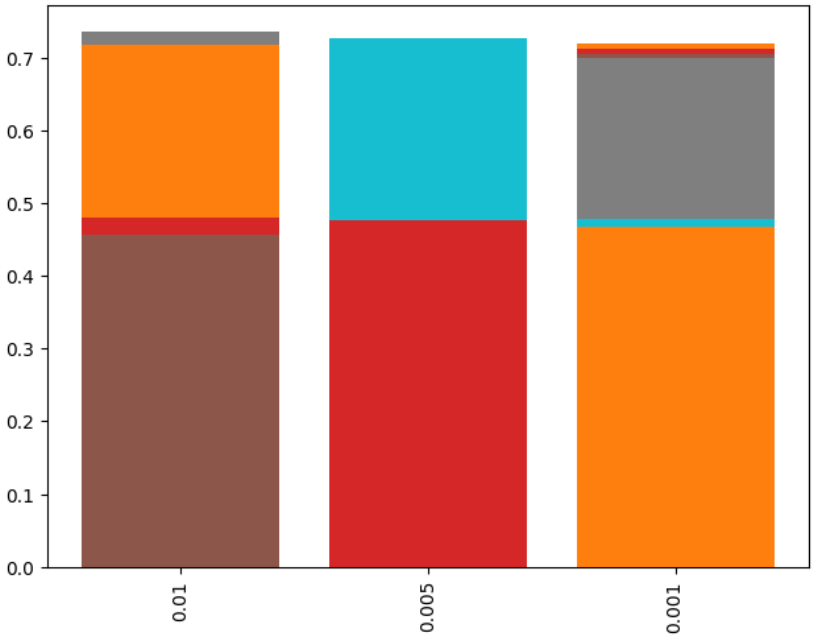
Solver



Learning Rate



Alpha



Based on the analysis of the data, the optimal hyperparameters have been determined for both the 30-second and 3-second samples.

For the 30-second samples, the optimal hyperparameters are **tanh** activation function, an regularization alpha of **0.01**, hidden layer of **(50, 50, 50)** neurons, **adaptive** learning rate, and the **adam** solver. Adam is an optimization algorithm used for stochastic gradient descent, well suited for large datasets and a high number of parameters.

On the other hand, for the 3-second samples, the optimal hyperparameters are also **tanh** activation, alpha of **0.01** , hidden layer of **(50, 50, 50)**, a **constant** learning rate, and the **adam** solver. It's interesting to see the learning rate prefer something constant over an adaptive rate. We expected the models to use the same parameters since they were both sampled from the same underlying data, but we speculated the 3 second samples are more discrete with more variance in data, thus making it a simpler model needing a constant learning rate.

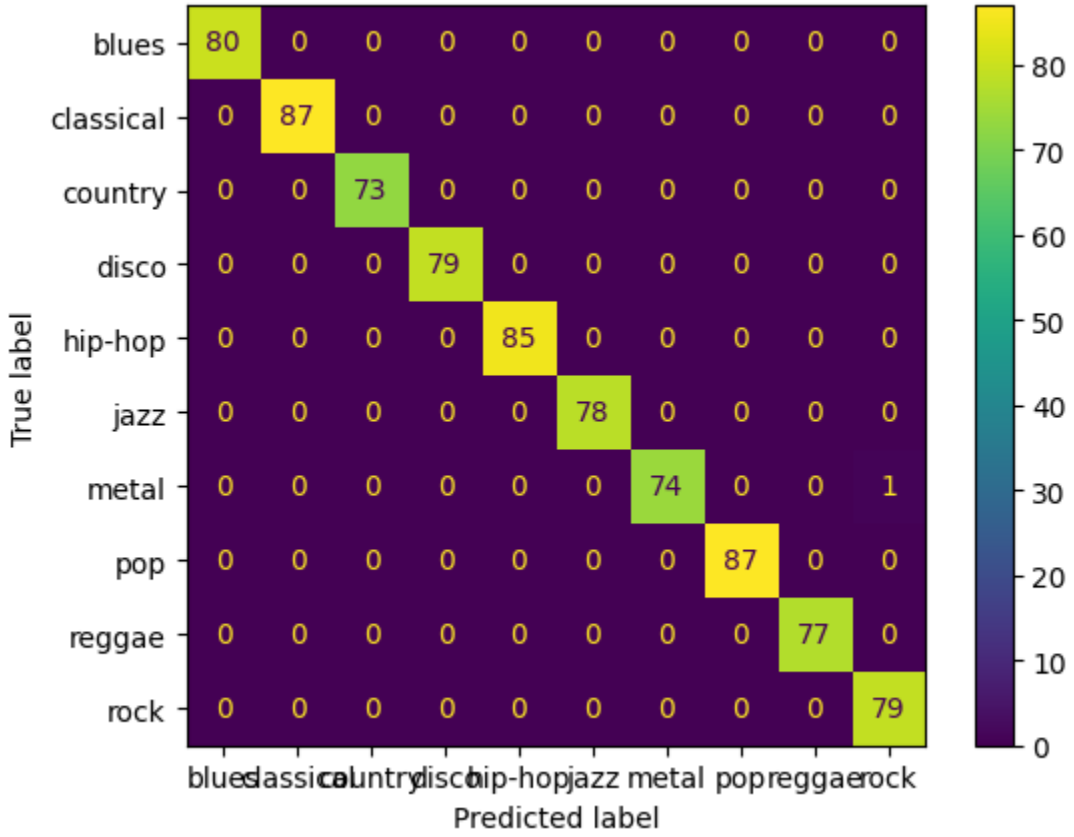
These hyperparameters were determined after running several experiments and evaluating the model's performance on the test data. By using these optimal hyperparameters, the model can achieve the highest accuracy and generalization performance on the given dataset.

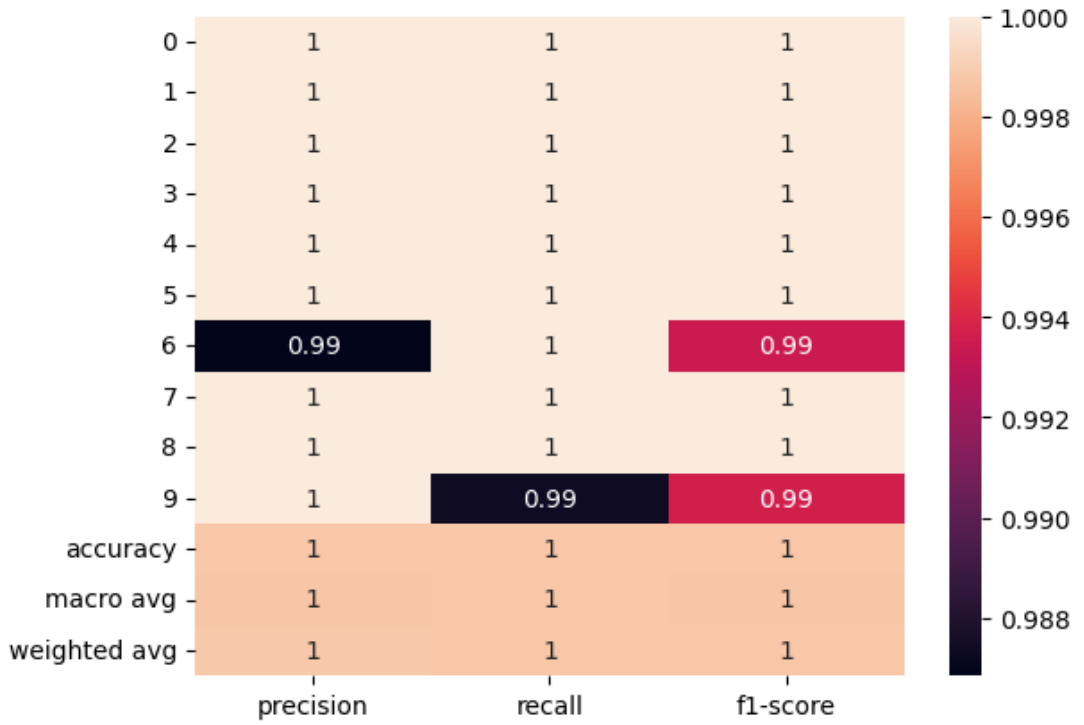
Model

The neural network uses the MLPClassifier class, implementing a multi-layer perceptron algorithm to classify data. The parameters are the optimized parameter set found in the previous section.

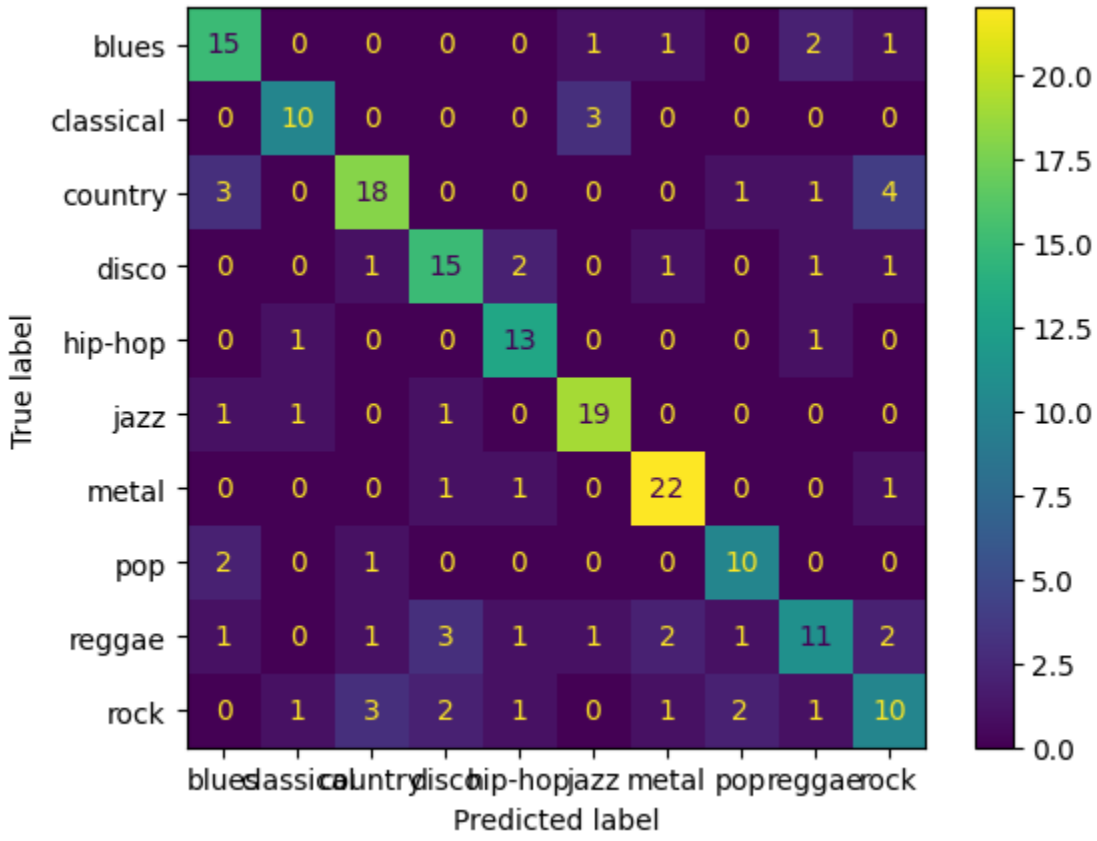
The confusion matrix is a 10x10 matrix. Each row represents the total amount of samples in class number 1. Out of the 80 samples in the set, 52 were classified correctly as class 1. 10 were incorrectly classified as label 3.

Confusion Matrix - Classification Matrix(30 second - training set)

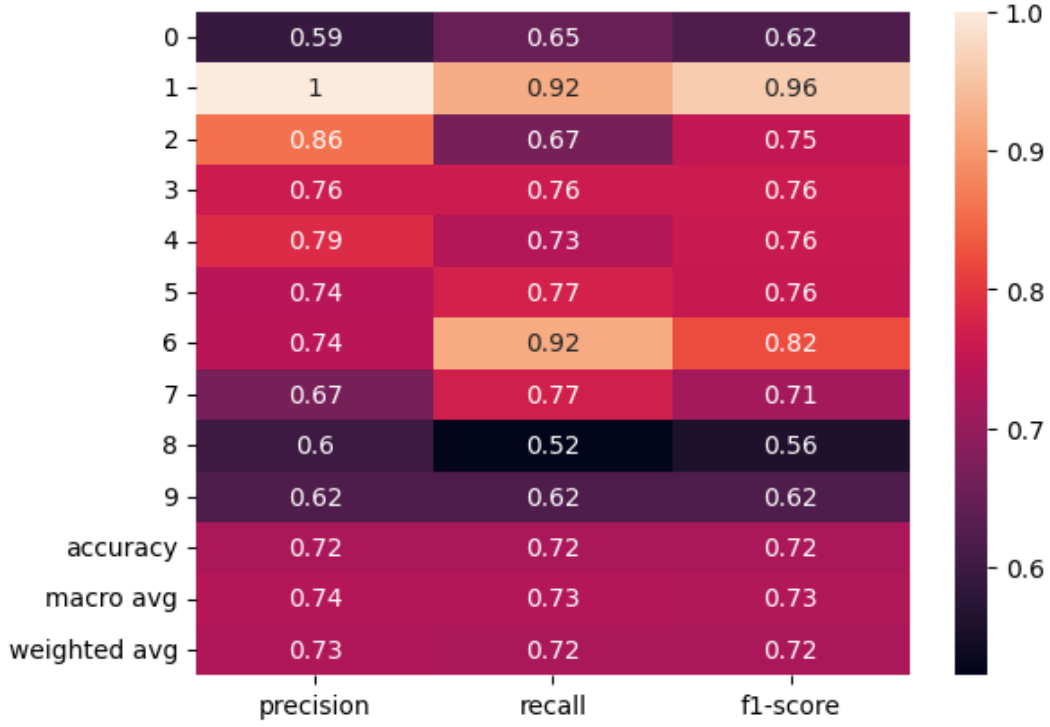




Confusion matrix - Classification Matrix (30 second - testing set)



Classification report (30 second - testing set)



Training Loss Plot (50,50,50 neural network w/ optimal parameters)

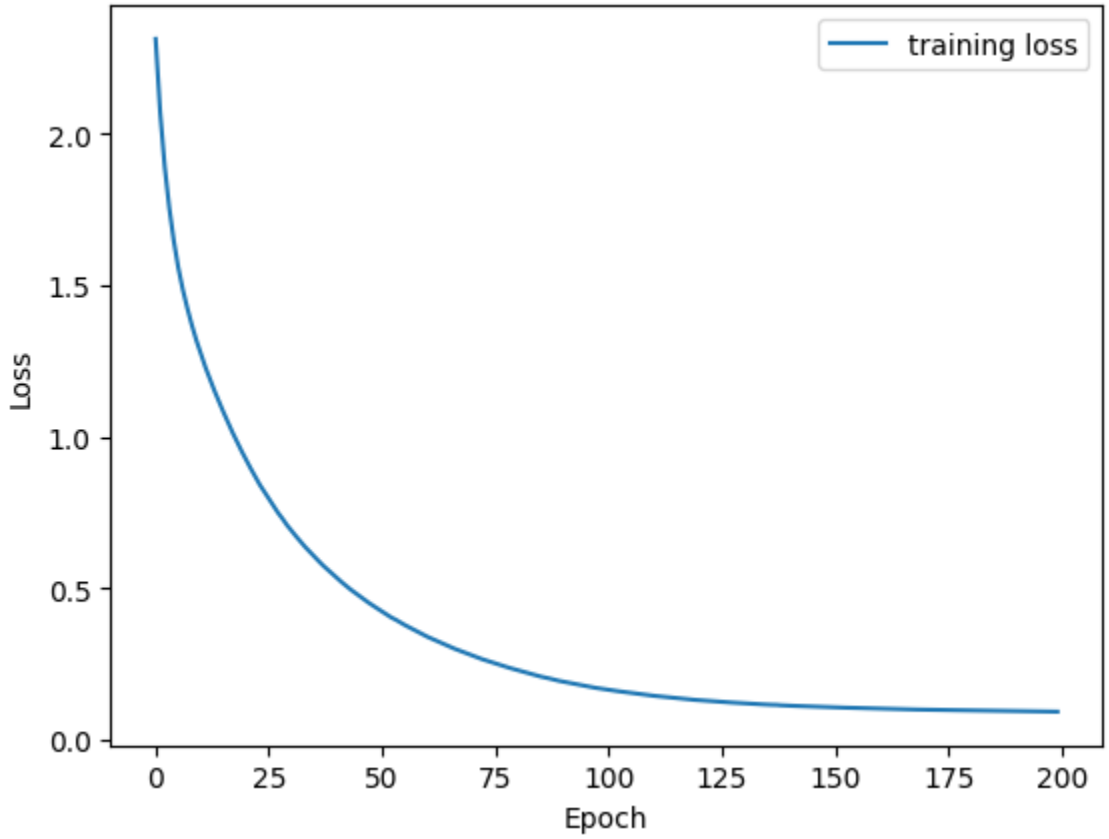


Table 1 - Measuring different hidden layer sizes

Model **{** **activation:'tanh',** **solver: 'adam',** **learning_rate: 'adaptive'** **alpha: 0.01** **}**	**Best Training Accuracy**	**Best Testing Accuracy**
(50,50,50)	1.00	0.70
(40,40,40)	1.00	0.69
(20,30,30,30,20)	0.99	0.66
(20,30,20)	0.97	0.70

Table 2 - Measuring the effect of regularization

Model **{** **activation:'tanh',** **solver: 'adam',** **learning_rate: 'adaptive'** **hidden_layer: (50,50,50)** **}**	**Best Training Accuracy**	**Best Testing Accuracy**
alpha = 0.01	1.00	0.70
alpha = 0.1	1.00	0.71
alpha = 0.05	0.99	0.70

Conclusion

In conclusion, the best neural network for this dataset is a 3 layer network with the optimal parameters. The sklearn classifier uses L2 normalization to improve the network. The number of neurons in the hidden layers are 50,50,50 respectively. This model had a maximum testing accuracy of 71% with less than 0.5% loss after 200 epochs. After evaluating the training loss chart, epoch 100 is around the cutoff for a 0.5% loss.

We noticed the training accuracy using these optimal settings is extremely high, 99% to nearly 100% accuracy modeling the training dataset. We believe that there was a high probability of overfitting the training dataset. When we utilized a different hidden-layer neuron configuration (20,30,20), we noticed that the testing accuracy hardly went down yet the training accuracy dropped. The hypothesis was tested by changing the strength of the regularization term 'alpha'. Keeping every other parameter consistent with the results of GridSearchCV, we observed there was no significant difference in testing accuracy with the change of the regularization term. Thus, we reasoned that the model could not be improved with the dataset. An improvement could be made to the model if we increased the size of the dataset. Additionally, K-fold cross validation could have been utilized to improve the overfitting issue of the model, but we wanted to focus on the parameters that held more weight such as varying the hidden layers.

3.2 Logistic Regression

The second model we used was logistic regression. We used sklearn's library for Logistic Regression. We tried polynomial transformations on the data to see if it would increase the model's performance. On the table below, we can see that there was actually an increase in the model's accuracy when we transformed the data to a third degree polynomial. We were unfortunately not able to use any higher polynomial transformations as the runtime grew exponentially.

Various Feature Transformations Performance On Test Data

Poly Degree	**1**	**2**	**3**
F-1 Score	0.71	0.74	0.74
Recall	0.71	0.74	0.74
Precision	0.70	0.74	0.75
Accuracy	0.69	0.73	0.73

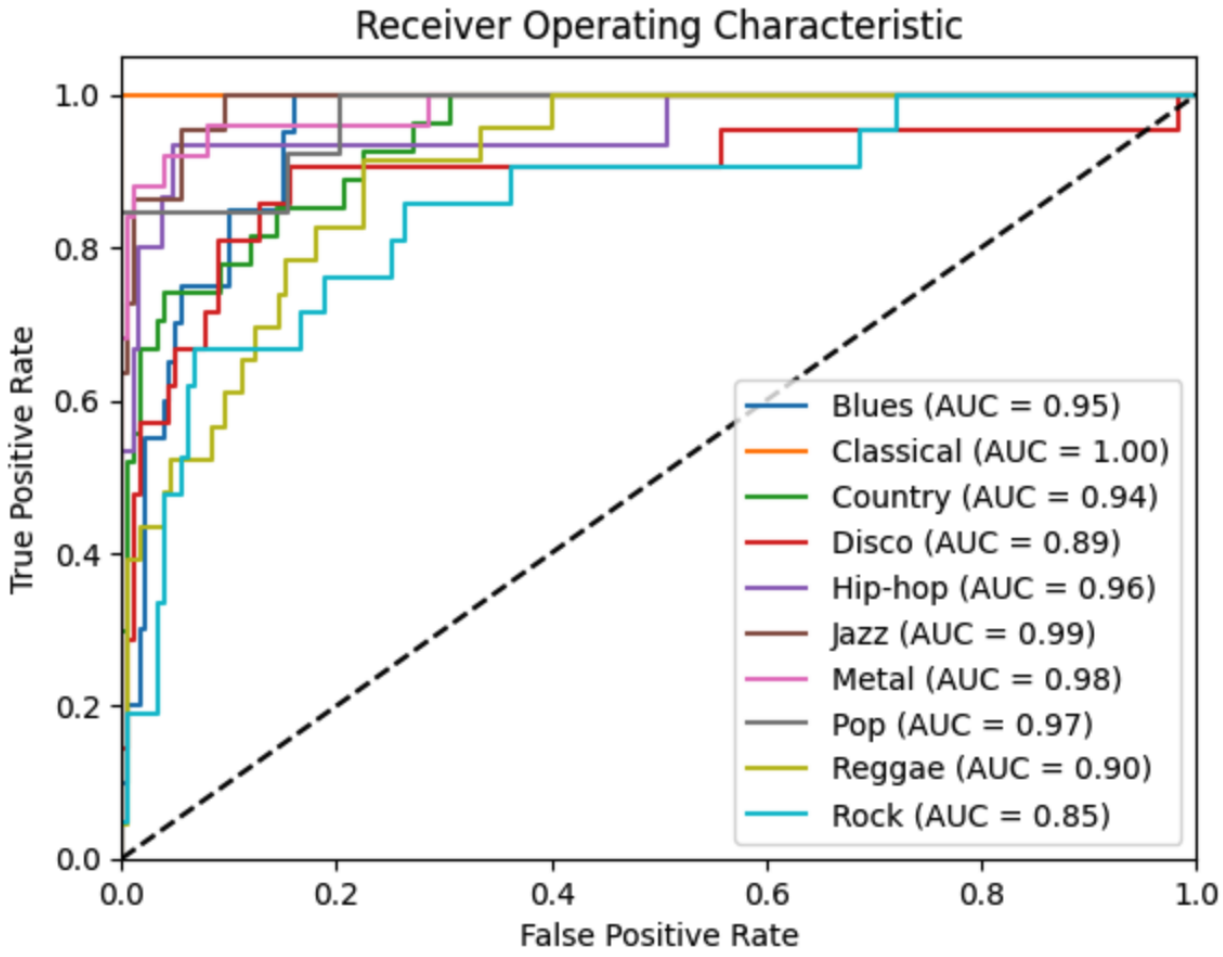
We also used the K-Fold cross-validation to choose the best value of C. The following table shows the accuracy for the 8 c-values that were tested using K-Fold cross validation:

Optimal C values of L2 Regularization Using K-fold Cross-Validation

K	**Best C**	**Accuracy**
2	0.1	0.69
3	0.1	0.69
4	0.2	0.71
5	0.25	0.73
6	0.35	0.72
7	0.7	0.7

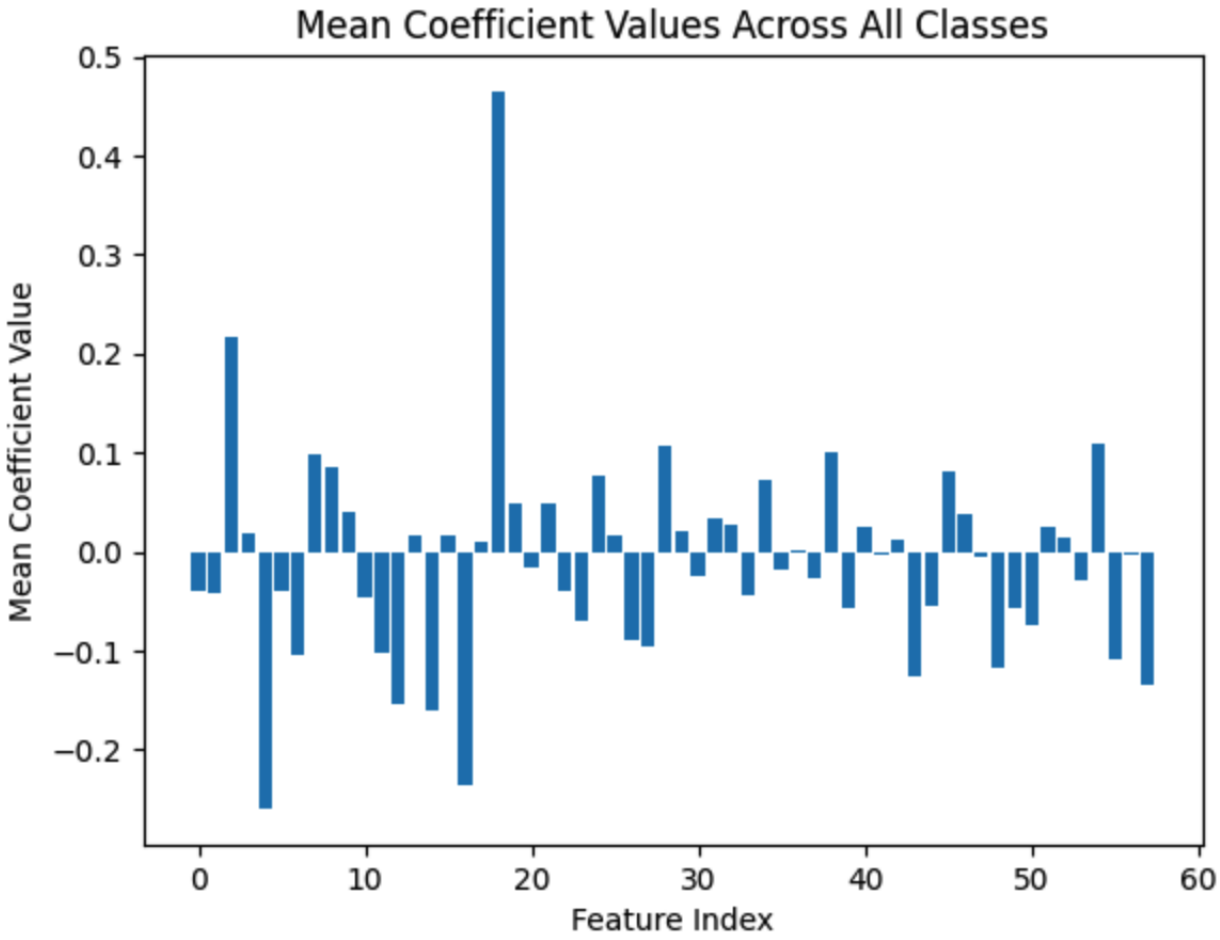
We clearly observe that the L2 regularization does not help our model. The reason could be that our model is too complex: L2 regularization may not be able to effectively control the weights of

complex models, and other techniques such as dropout or batch normalization may be more appropriate.



The class with the highest AUC value is Classical music and it has a curve that is closest to the top-left corner of the plot, with AUC value 1, indicating that the model is able to correctly classify positive examples of Classical songs 100% of the time on unseen data which is extremely impressive. Blues, Hip-hop, Metal, and Pop also have high AUC values, indicating that the model is able to correctly classify sounds of these classes with high TPR and low FPR. Country, Jazz, and Reggae have slightly lower AUC values, indicating that the model may have more difficulty distinguishing members of these classes from the negative class (i.e., examples that do not belong to any of these classes). Disco and Rock have the lowest AUC values, indicating that the model has the most difficulty correctly classifying examples belonging to these classes.

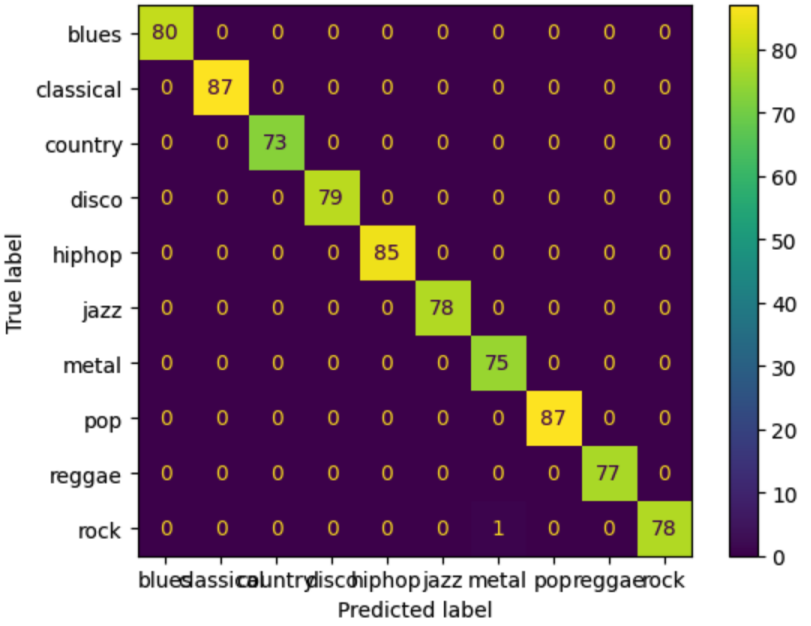
Let’s look at the weights of our features in our logistic regression model:



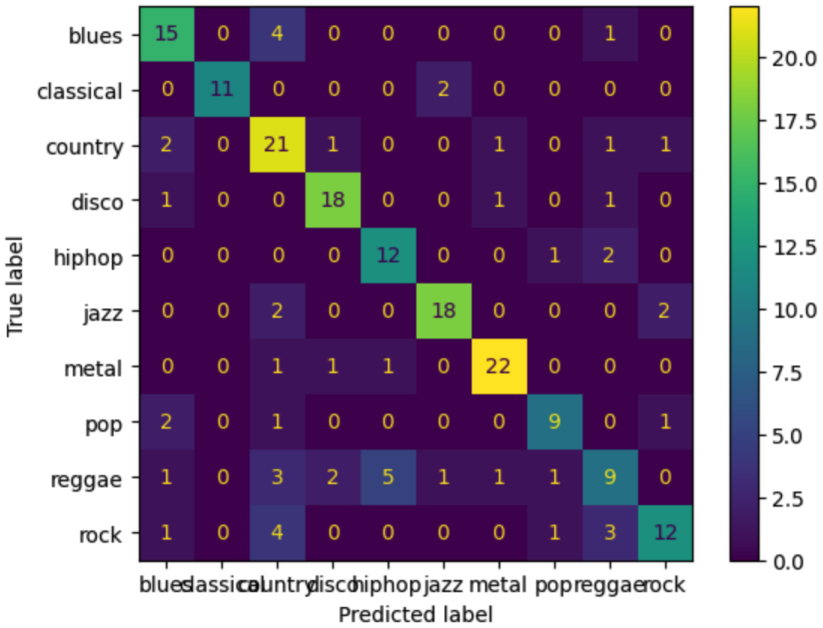
The average weights can be used to interpret the relative importance of the different features in the model. A feature with a higher average weight indicates a stronger influence on the predicted outcome, while a feature with a lower average weight has less influence. The feature that has the highest mean coefficient value across all iterations of the model was harmony variance., while some features had almost no effect on the classification. Overall, expect some outliers, we can see most of the features played a part in the classification.

Confusion Matrices for Logistic Regression with Degree 3 Transformation:

TRAIN DATA

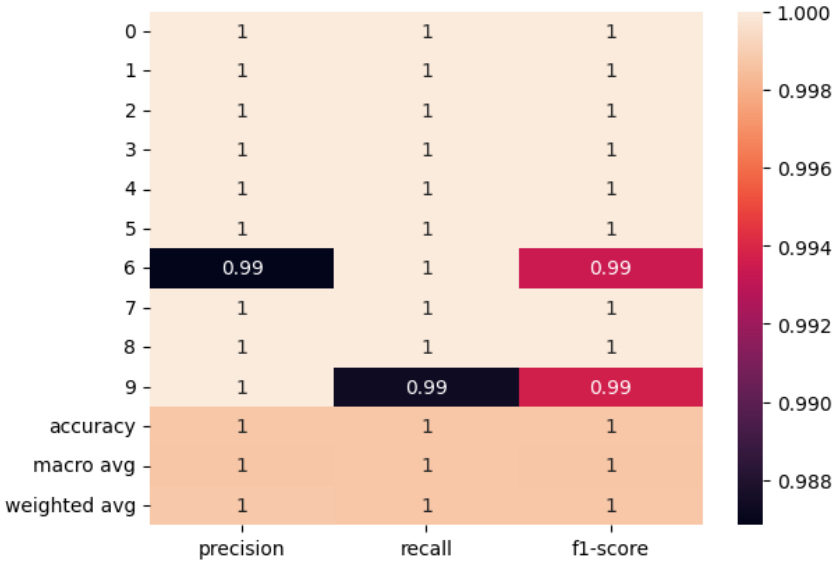


TEST DATA



Classification Report for Logistic Regression with Degree 3 Transformation:

Train Data



Test Data

We can see the test data had an extremely high accuracy and precision, this could be because logistic regression models are prone to overfitting when the model is too complex. In our case, the algorithm could have learned to fit the noise in the training data. However, looking at the results based on the test data, we see that the model did not have trouble generalizing on the unseen data, as the model had a 0.73 accuracy on test data.

To conclude, the logistic regression model that gave us the best results was the model with degree 3 and no regularization. The model had the stats:

Best Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Deg 3, no reg	0.73	0.75	0.74	0.74

3.3 SVM

The last model we used was Support Vector Machines. The SVM algorithm works by transforming the input data into a high-dimensional space, where the classes can be linearly separated by a hyperplane. We tried running the SVM model for 4 different kernel's: linear, polynomial, radial basis function, and sigmoid. We utilized the sklearn GridSearchCV library to run through multiple hyperparameters to search multiple parameter spaces, hyperparameters such as C, gamma, coefficient 0 etc.

We used the k-fold cross-validation technique to tune the hyperparameters of the SVM algorithm, such as the regularization parameter C and the respective kernel function parameters, by testing different combinations of hyperparameters on different folds of the data. This helps to choose the best combination of hyperparameters that generalizes well to new data and improves the model's performance.

To start off, let's look at the linear and polynomial kernels we used to optimize our model:

Polynomial Kernels

Kernel	**LINEAR**	**Degree 2**	**Degree 3**	**Degree 4**	**Degree 5**	**Degree 6**
F-1 Score	0.70	0.67	0.74	0.59	0.62	0.50
Recall	0.71	0.67	0.75	0.58	0.61	0.50
Precision	0.71	0.68	0.76	0.64	0.70	0.60
Accuracy	0.69	0.68	0.74	0.59	0.60	0.51

We can see that the accuracy of our model peaks at degree 3 kernel and decreases further as we increase the degree of the polynomial. This is quite interesting, as our logistic regression model also performed best in a Degree 3 feature transformation. This could mean that the classification pattern between the features and the data could be a complex cubic relationship.

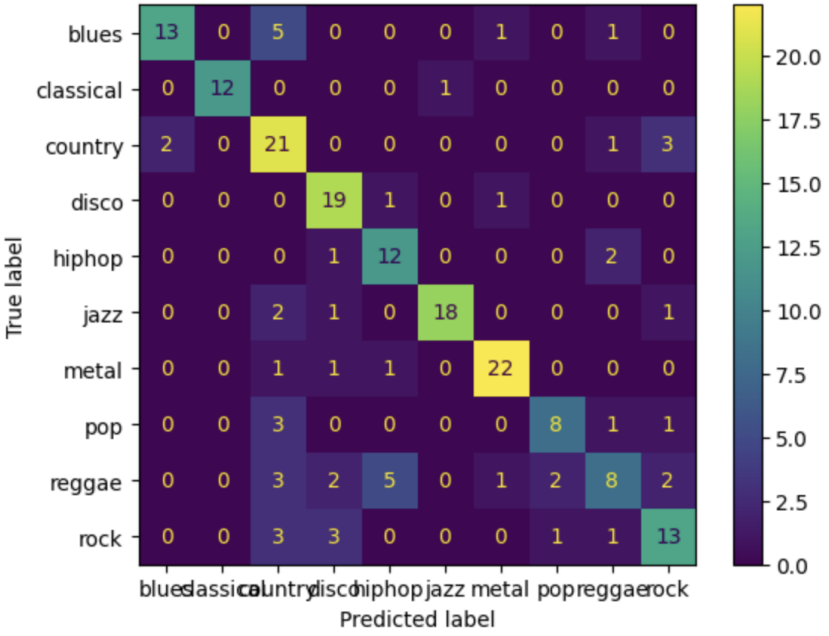
Now let's look at the RBF and sigmoid kernels for our SVM model:

Other Kernels

Kernel	**Radial Basis Function**	**Sigmoid**
F-1 Score	**0.73**	**0.73**
Recall	**0.74**	**0.74**
Precision	**0.73**	**0.73**
Accuracy	**0.73**	**0.73**

Other Kernel's did perform fairly well, but they did not perform better than the degree 3 Polynomial Kernel. We used cross-validation techniques to test for different parameters in these kernels.

Classification Report and Confusion Matrix on Test Data for SVM



4. Conclusion

Support Vector Machines and Logistic Regression models performed similarly. Using a 3rd degree polynomial transformation, the classifier reached as high as 75%. Comparatively, neural networks, despite being our goto for classification examples, didn't perform as well, only at 70% accuracy. Neural networks are used to represent complex relationships between features stored in the complex interconnected hidden layers. After visualizing the data in the beginning, we noticed there was little variation in between either label, making us believe there were complex layers needed to represent the model. However, the exploration into neural networks led to more overfitting. The training accuracy was significantly higher than the testing accuracy, modeling the noise and variance too closely. An improvement to this model could be limiting the epochs in the training to a lower number.

Support Vector Machines improved the highest classification accuracy but not by a significant amount. The data is non-linearly separable from the initial assessment of the data. When we transformed the dataset into third degree space, the accuracy significantly improved from 0.70 at linear models to 0.74 on degree 3. At the moment, we are unsure about improvements to the model. From our research, gradient boosting could potentially improve the accuracy through minimizing a cost function by adding weak models that focus on the misclassified data points. But this isn't implemented in our project

We were surprised by the accuracy of the logistic regression model achieving similar results to the SVM. Logistic regression tries to fit a probability distribution to the data and is sensitive to outliers. Since this model performed as well as others, we concluded that there were no significant outliers in the data. By looking at the weights for each feature, we noticed that rms_mean, spectral_centroid_mean, and tempo were the most influential to the model. Looking at these values empirically, rms_mean represents energy in the sound which would make a significant difference in classifying between a sample with rock and a sample with classical music. We, however, believe that the logistic regression model is also overfitting the data. With such a high training accuracy but relatively low testing accuracy, we speculate that the data is simply too complex to be classified with this regression model.

5. Notes

When we first started this project, we speculated how to classify differences between a rock song and a pop song. However, by understanding how each sample was recorded and analyzing the discrete data points that represent the sample such as the MFCC and RMS, we acknowledged the capability of using machine learning to categorize them.

From a purely observational perspective, it's not easy to tell the difference between hip-hop and pop as influences from either genre motivate the musical scene in the other. Rather, it was astonishing seeing as high of an accuracy as 74% for an SVM to predict the genre correctly

compared to the guessing average of 10%. An improvement we could make to the project would likely be working with a lower number of labels. Having two music genres that are distinct enough where even humans could find the difference in tonal quality (eg. classical and rock music) would likely yield higher accuracy. However, in order for that to classify more labels, more models would have to be created to reach the same conclusion.

Another method the project could hope to tackle would be more feature transformations. Due to technical limitations, changing the degree of the logistic classifier is enough to exponentially increase the time it takes to fit the model. Some of the motivating examples include transforming the model to a sinusoidal space as such audio samples would be continuously represented as such.

An extension of this project would be to utilize the Librosa Python library to generate new music samples from more updated song tracks.

Works Cited

GTZAN Dataset - Music Genre Classification

https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification

Scikit-Learn GridSearchCV

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Scikit-Learn SVC (implementation of libsvm)

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Scikit-Learn MLPClassifier

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html