Multi-Class Classification with Neural Networks

Project Goals

I wanted to create as multi-class classification model for this dataset with the following goals:

1. **Great Generalizability** (Low Bias, Low Variance)
2. **High Test AUROC**
3. **Non-Complicated Feature Set** (Low Dimensionality)
4. **Checks on Data Leakage** to Ensure True Performance
5. **Compact and Fast Training Models**

Model Choice

I experimented with all models, across all feature sets, and settled on Neural Networks for the task. Deep NN's were favored over tree-based ensemble models – Random Forests, AdaBoost, and XGBoost - due to my need for precise tunability. For this dataset, while I FAFO'd, the tree-based models tended to exhibit high variance and poor generalizability. The NN based models, however, could be precisely hyperparameter-tuned to not overfit. I did this by testing different number of layers, neurons per layers, learning rates, and epochs. Tuning these hyperparameters and tracking performance across epochs enabled me to get the model just right – with high AUROC, excellent generalizability (minimal deviations between train and test performance), and faster train-times compared to tree-based ensemble models.

Project Outline

1. Data Cleaning and Preprocessing

Cleaning - The dataset had inconsistencies. There were NaN rows, values of duration were negative across 10% of the data, values for tempo were '?' across 10% of the data – these were imputed. 2489 artist names were empty, these were not dealt with as the main models do not use these features. I decided to drop 5 rows that were entirely NaN. The obtained_date feature had very limited value counts and held no relevance. Thus, the final feature set included 13 features – popularity, acousticness, danceability, duration, energy, instrumentalness, key, liveness, loudness, mode, speechiness, tempo, and valence.

Imputation - For negative values of duration, I decided to impute them with the median of the artist's positive duration times. In cases where the artists had no positive duration times, the duration was imputed with the dataset's global duration median. **The median was chosen over the mean as the duration feature had significant outliers**. I did not address these outliers, since after inspecting, a lot of songs had extremely high durations, but these were legitimate songs (not podcasts), hence important information for the models. For imputing tempo, **since it may be critical to classifying genre later**, I tested regression models for more accurate imputing. Following the theme of this project, **I chose a deep NN, after trying linear regression and tree-based models, since it gave the best balance between bias and variance.** For tuning this NN, a train and validation set was created to assess performance. All models were poor performers at R^2 of 0.1-0.14, but the NN performed the best. This model was used to impute missing values for duration. While the R^2 is low, I felt like it was a more structured approach than just imputing with the mean/medians – which could bias downstream models.

Encoding and Standardization - Categorical variables including Key, Mode, and Genre, were ordinally encoded. While mode turned into a binary feature, key and genre had multiple labels. All continuous features were standardized. In models that required PCA (PCA & t-SNE), I was forced to not include Key, this was because otherwise, due to its magnitude, it would be assigned the highest eigenvalue. Standardization was not an option as it would make the categorical data meaningless.

<u>Preventing Data Leakage</u> - This was a key concern with all the models I was running – for classification or imputation – and all the dimensionality reduction methods I was using. The data frame was shuffled before splitting. The data was split into train and test-sets, as prescribed by the spec sheet, just after cleaning, imputing, and encoding. The train and sets were shuffled again after splitting too, as the test-set was built using iteration. This was to prevent any sequential relationships that the NN models could adapt to. Standard Scaler was fit only on the train set and this fit was used to transform the test features. For the NN Imputer model, leakage is not a concern as it is a regressor and did not see any labels. For all dimensionality reduction methods, I fit to the train set and applied that fit on the test feature set. The PCA did not see the labels, the LDA was trained only on the train labels, for t-SNE, the entire predictor set needed to be passed, I did not however, pass the labels into t-SNE. These measures ensure that my eventual classification models reflect true performance. The only potential source of leakage I see is the limited number of artists. There are 6863 artists in total, thus there is no way to create a train and set with no overlaps of artists. **This will only turn into an issue when we use artist name as a feature.**

2. Exploratory Data Analysis and Feature Selection

I explored the data using histograms and correlation plots. These enabled me to see how features distributed, observe anomalies (outliers in duration), and select features for my models. Most of the features were not normally distributed, but since we are using Neural Networks, we don't make any assumptions about the data. Using correlation heatmaps (both Pearson and Spearman based on feature), I observed features that best correlated with tempo, and used these for my imputer - unfortunately, the performance with a limited feature set was poor. Conclusively, this also highlights that some features may not be sufficiently modeled by linear relationships – making the case stronger to use NNs.

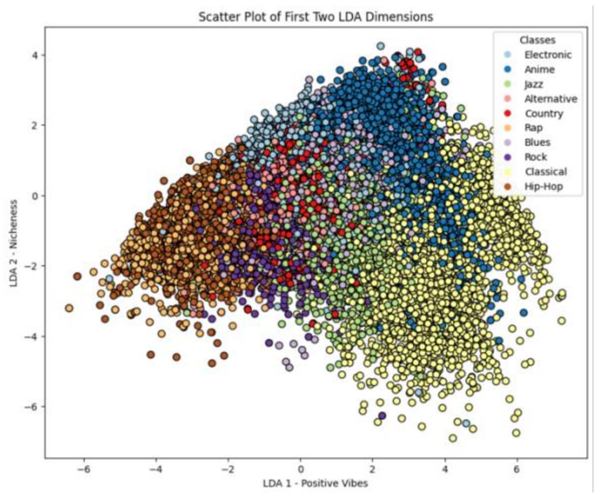
3. Feature Extraction via LDA (Dimensionality Reduction) and Clustering

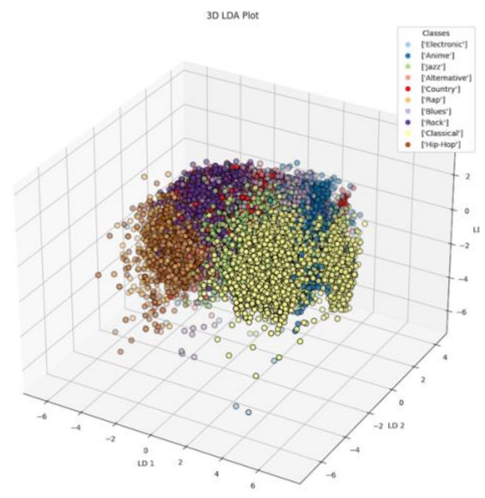
<u>Dimensionality Reduction</u>

Both linear and non-linear dimensionality reduction methods were used. For Linear methods, I experimented with both PCA and LDA. Since we had labels, applying LDA was a better choice since it considered the genres and maximized separability between genres, while minimizing separation within genres. This was reflected in the plots too, **LDA gave a better separation than PCA – both in the 2D and 3D space.**

I chose the top 3 LDA features since together they explained 92% of the separation across genres. Based on the loadings, the first feature, which explains 64% of the class separation, relies heavily on danceability, instrumentalness, and energy. I interpret this as **"Positive Vibe".** This shows that genres can be determined based on whether a song has a positive vibe or not. The second component, which explains 17% of the separation, weighs very heavily against popularity. This suggests that some genres, in general, might have less popular songs. I interpret this as **"Nicheness"**. The third feature, explains 11% of the separation, and is weighted by acousticness, instrumentalness, mode, and valence. I interpret this as **'Rhythm"**, and this suggests that genres can be separated across rhythm.

These interpretations make sense when we look at the axes of the 2D plot to see that Anime is on the higher end of the Niche Scale while Classical is on the lower end. Moreover, Rap is on the negative side of Positive Vibes and Jazz is on the positive axis.

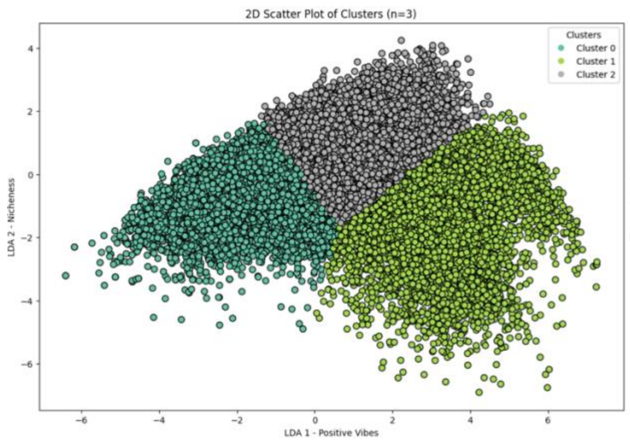


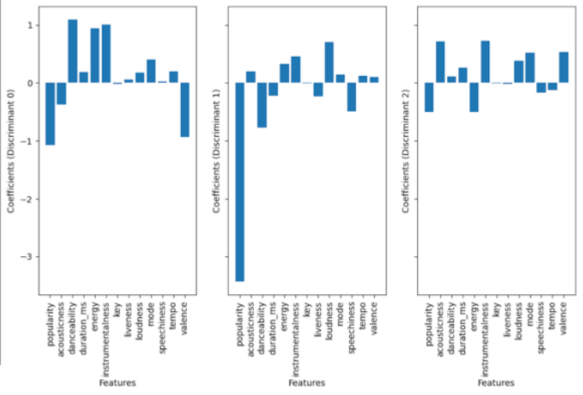


For non-linear methods, I applied t-SNE as MDS and UMAP were not feasible given the dataset size and my compute resources. To preserve the global structure of the data, I applied PCA before t-SNE – this was to make the features more interpretable. I ran both 2D and 3D t-SNE, at perplexity 30 and 50, however but there was significant overlap between the genres and and t-SNE not yield the visual separability outputted by LDA. **They also took a lot of time to train, conclusively, LDA was my final choice for dimensionality reduction.**

Clustering

I then applied k-Means to the LDA features to observe potential clusters in the 2D and 3D space. In both cases, 3 clusters were optimal with a moderate Silhouette Score of 0.35 (3D) and 0.44 (2D). These cluster labels can be used as additional features for the models, but I avoided that for the sake of a reduced feature-set and wanting the models to discover natural relationships themselves. The three clusters in the 2D space are very interpretable (1) – Non-Positive and Non-Niche Songs (Rap, Rock, Hip Hop) (Dark Green), Moderately Positive but Niche Songs (Anime, Electronic) (Gray), and Very Positive Songs (Light Green) (Classical, Jazz).





Top 3 LDA Feature Loadings for Interpretation

4. Model Building and Validation

I built 3 final NN models, and I distinguish them by model complexity. I publish 3 models instead of 1, as all models have good performance, and hence, they can be used according to use case and compute resources. The first is a model built with all features and the second uses the top 3 LDA features, the third used the top 2 LDA features. (As a bonus, in the end, I also publish a model with word features).

Hyperparameter Tuning

Loss Function: nn.CrossEntropyLoss was used for all models.

Optimizer: Adam was chosen since it can have different learning rates across different features for precise steps.

Learning Rates: Most models converged at either 0.1 or 0.01. In some cases, I had very precise increments to balance train-time and limit variance.

Activation Functions – ReLU was used across due to faster train times and convergence. nn.CrossEntropyLoss applies Softmax natively, so the models do not have Softmax activations.

Layers: I scaled the layers based on input features. Smaller models have 1 hidden layer, larger ones have 2.

Neurons per Layer: Varies model to model to model and depends on input features. Usually kept low to minimize training time.

Dropout/Regularization: I did not need dropout layers. Based on train-test loss plots, I observed that regularization was not needed and ended the training at epochs where the test-losses started to increase over the train-losses.

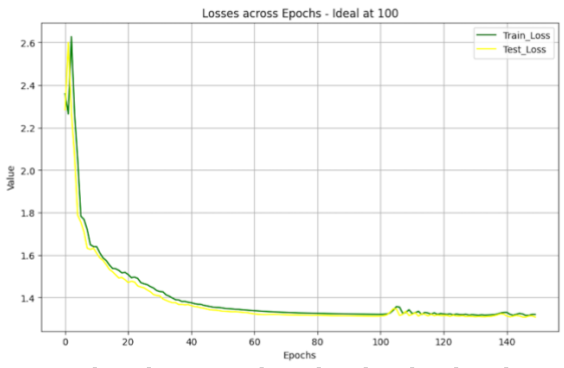
Epochs: They were a function of learning rate and number of features. Most models were trained between 100-150 epochs. For final models, I limit the epochs to where the train and test loss curves cross (train loss falls below test loss) as it was an indicator of the model starting to overfit.

Model Validation (All Reported Scores are Test Scores)

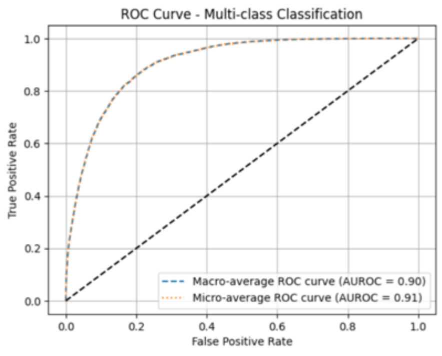
Model	AUROC (Macro)	Test Accuracy	Input Features	Hidden Layers	Total Parameters	Optimal Epochs	Train Time
All Features	**0.93**	**58%**	**13**	**2**	**755**	**125**	**34 secs**
LDA 3D	**0.91**	**50%**	**3**	**1**	**220**	**100**	**24 secs**
LDA 2D	**0.88**	**42%**	**2**	**1**	**205**	**150**	**17 secs**
Gradient Boost Forest on LDA 3D (For Benchmark)	**0.89**	**49%**	**3**	**N/A**	**100 Estimators**	**N/A**	**41 secs**

The 3D LDA model, according to me, is the best balance between performance and complexity. It uses only 3 features, has only one hidden layer (therefore, is not even a deep model), trains twice as fast as the gradient boost forest, has minimal deviations between train and test performance highlighting generalizability - and gives a high test AUROC of 0.91. **This model meets the goals of the project that I initially mentioned.**

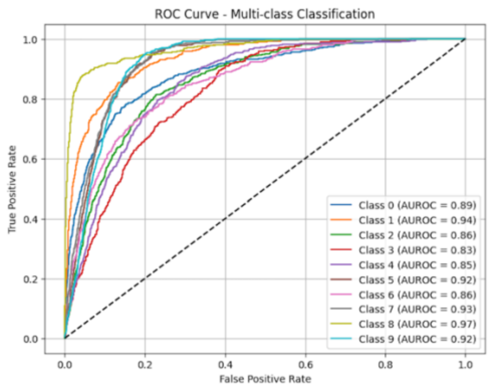
For the features that impact this model's AUROC the most, after iteratively removing features, I found that the first LDA feature has the biggest impact, in fact, just running the model with the first LDA feature gave an AUROC of 0.82. We can infer our LDA loadings to conclude that for this model's AUROC - danceability, energy and instrumentalness (positivity) will have the largest impact, in order. Conclusively, I believe reducing the dimensionality of the dataset, with LDA, which takes into account labels, enabled a high AUROC despite a very reduced feature set.



Epochs and LRs were chosen based on these loss plots



Micro and Macro AUC Curves



Genre-wise AUROC Curves

Extra Credit - NLP

Pre-Processing

Word Features- Two Features were unused across all the previous models – the track and artist name. I wanted to use them and see if they added any value to the classification model. To achieve this, I researched some NLP methods.

Vectorizing Words – I cleaned all word features and then I used SpaCY's pretrained model to lemmatize, tokenize, and vectorize the features of artist and track names. Then I standardized these vectors to ensure that their magnitude is consistent with the features used in our original models. The artist and track name were both converted into 300 dimensional vectors. So, in total, I had 600 word features.

Model Building for Classification

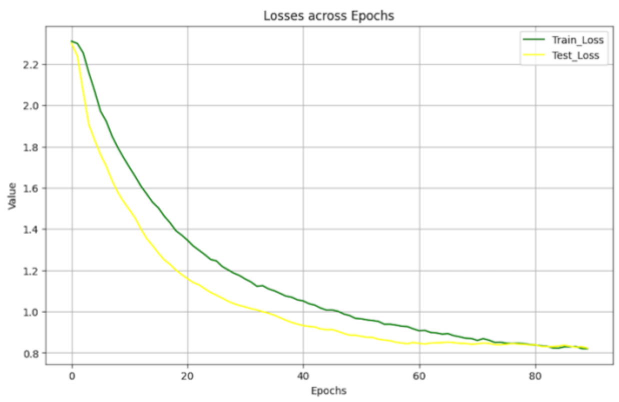
Neural Networks were again used due to the number of features. I ran two final models, one with only word vectors and the other with word vectors and the original features.

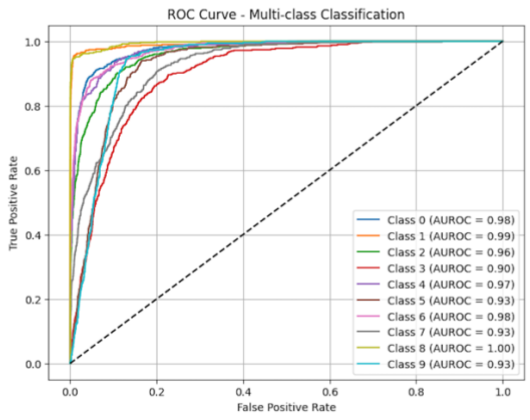
Addition of Dropout – Since these were deeper models with more layers and more neurons, they tended to overfit very quickly. The train AUROC's for the model with all features went up to 1 – highlighting significant overfit. For regularization, I added dropout layers after every activation. These significantly reduced variance and enhanced the model's generalizability.

Model Validation

Model	AUROC	Test Accuracy	Input Features	Hidden Layers	Total Parameters	Optimal Epochs	Train Time
Word Features	0.88	60%	600	4	146885	40	21 secs
All Features	0.96	70%	613	4	149485	90	72secs

Based on these findings, we can conclude that artist name and track name do help us model genre better.





Concerns

While this mode's performance here is very strong, and as the graphs suggest, variance is low (test loss is below train loss till about 90 epochs), this model is very complex. We have a lot of parameters, the train-time, if we include time to vectorize, is over an hour. Another thing to note is that our model with original features gave an AUROC of 0.93 and the model with only word features gave an AUROC of 0.91 – combining these feature sets though, only helped incrementally. I believe the addition of the 600-dimension word vectors diminishes the impact the original features have on classification- hence it is more than likely that the words are overpowering our original features. **There is also potential leakage. Since we have about 6863 artists in total, there are definite overlaps between the train and test set for artist names that might explain these high AUROCs. Conclusively, this is not my ideal model or approach.**

Going Forward

All models presented here – with or without the word features - have additional performance that can be harnessed via hyperparameter tuning. Apart from tuning them to perfection, one thing that I want to test out is reducing the dimensionality of the 300 dimensional vectors for the words and artists, and then incorporating them with the existing 13 features. This will do two things – it will make the models with word features less complex and it will ensure equal influence of all features on final classification performance. For this approach, I could look into linear or non-linear dimensionality reduction for the word vectors. Or I could use SpaCy's smaller size English models to have lower dimensional word vectors in the first-place.