Master in Mechanical Engineering

Advanced Automation

Universidade de Lisboa - Instituto Superior Técnico

Databases
Spotify Track Popularity Prediction

GITHUB

João Rosado - 96409, joaopedrorosado@tecnico.ulisboa.pt

January 2024

Table of Contents

1 Introduction

The popularity of a song isn't easily explained by a set formula; it's more about emotion, relatability, and a pleasing sound. While catchy tunes and skillful instrumentation matter, the real magic often comes from a narrative that strikes a chord with many or an artist's genuine expression. Emotional connection goes beyond genres, and a distinct production style can make a song stand out.

Nevertheless, there can be certain correlations, both positive and negative, of some parameters to the general popularity of a given track, and these can definitely be looked at and studied.

Such examination encompasses the proposition of this report.

2 Dataset and Data Processing

The subject of study leans on a Spotify track dataset containing over 114.000 entries, which has been modified to better suit the different needs of the analysis. The main drawback of having such an extensive dataset is the large computational effort required to compute every machine learning methodology, as well as the time it would take to do so.

As a consequence, the original dataset was subject to some processing.

Firstly, it was reduced to a random sample of 15.000 instances. Then, columns with irrelevant data, such as track id, artist, album's name and track's name, were removed. Thirdly, the duration (which came in milliseconds) was converted to seconds and several dummy variables took the place of the track's genre. Of these genre dummy variables, ten were chosen to be kept in the dataset whereas the rest was removed, along with all the tracks of the correspondent excluded genres. Additionally, all rows with missing information were discarded.

The final form of the dataset is composed of 1305 entries.

A list of all predictors used to create the models, with their corresponding description, is shown in Table 1.

Table 1: List of Predictors

Column Name	**Description**
popularity	value based on the total number of plays (0-100)
duration	track length
explicit	explicit lyrics (True or False)
danceability	how suitable for dancing a track is (0-1)
energy	perceptual measure of intensity and activity (0-1)
key	key of the track
loudness	overall loudness (dB)
mode	minor=0 / major=1
speechiness	presence of spoken words (0-1)
acousticness	confidence measure of tracks' acousticness (0-1)
instrumentalness	whether the track contains no vocals (0-1)
liveness	presence of audience in the recording
valence	musical positiveness (0-1)
tempo	beats per minute (BPM)
time_signature	beats per bar
track_genre_(...)	the genre in which the track belongs

3 Methods and Results

3.1 Linear Regression

Linear regression is used for predicting a continuous outcome variable based on one or more predictor variables that are assumed to have a linear relationship with the outcome. The equation of a multi-variable linear regression is given by Equation 1.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n + \epsilon \tag{1}$$

where Y is the outcome variable, β_0 is the intercept, β_1, β_2, ..., β_n are the coefficients of the independent variables X_1, X_2, ..., X_n. In order to determine the coefficients, Ordinary Least Squares (OLS) was utilized. This method minimizes the sum of the squared differences between the predicted values and the actual values of the outcome variable.

Furthermore, interaction terms (Equation 2) and second-order polynomial terms (Equation 3) were incorporated to capture non-linear relationships in the data.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n + \beta_{n+1} X_1 X_2 + ... + beta_{2n} X_{n-1} X_n \epsilon \tag{2}$$

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n + \beta_{n+1} X_n^2 + \beta_{2n} X_n^2 + \epsilon \tag{3}$$

The selection of the X_n features was performed by plotting the correlation matrix (Figure 5) and choosing the independent variables with significant correlation to the outcome variable, *popularity*. In the end, eight predictors remained.

In Python, the model implementation was initialised by splitting the dataset into two partitions: one containing the predictors and the other comprising only the response variable. Using the *sklearn* package, it effortlessly crafted and trained the models on our data, subsequently obtaining each predictor's respective coefficient.

Table 2: Multi-variable Linear Regression Coefficients

Predictor	**Coefficient**
β_0	28.7539
duration	0.0251
energy	-3.9959
loudness	-0.0329
acousticness	3.8199
instrumentalness	-1.8873
track_genre_classical	-21.6496
track_genre_jazz	-19.5438
track_genre_pop	14.5196

Analysing Table 2 (and Table 7), it can be inferred that *acousticness* and *track_genre_pop* are the variables that more positively affect the popularity, while *track_genre_classical* and *track_genre_jazz* do so negatively. MLR with interaction terms and second-order polynomial terms are represented on Tables 8 and 9, respectively.

When examining the model errors in Table 3, it becomes evident that the linear regression incorporating quadratic terms exhibits superior performance across all parameters. This outcome is unsurprising, given that the inclusion of quadratic terms facilitates a more accurate fit to the dataset.

Table 3: Linear Regression Methods Errors

Errors	**Linear Regression**	**Linear Regression with Interactions**	**Linear Regression with Quadratic Terms**
R-squared	0.113	0,163	0,627
Adj. R-squared	0.108	0,141	0,615
AIC	12300	12280	12270
BIC	12350	12450	12470
F-statistic	20.68	7,488	55,89

3.2 Classification Methods

Classification methods are employed to forecast categorical outcomes based on a given set of input features. To facilitate the implementation, *high_popularity* was created, that equals 1 when the track's popularity is 75 or higher, and equals 0 otherwise.

3.2.1 Logistic Regression

Logistic regression is a type of linear model that is used for predicting the probability of a binary outcome. Lets consider only one feature ($\mathbf{X}$) and let $p(X) = Pr(Y = 1|X)$. The logistic regression is expressed in the following form:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \tag{4}$$

For each of the p predictors, a coefficient β_p must be determined. These, along with β_0, are computed using the maximum likelihood method and are presented in Table 10.

3.2.2 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) serves as a method applied for classification and dimensionality reduction. The main concept involves projecting the feature space into a lower-dimensional realm, aiming to maximize the separation between classes. This is accomplished by determining a linear combination of features that maximizes the ratio of between-class variance to within-class variance.

To ascertain the class of a sample, we rely on the discriminant score (Equation 5), which indicates the likelihood of an observation belonging to a specific class based on its feature values.

$$\delta_k(x) = x * \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \tag{5}$$

3.2.3 Naive Bayes

Naive Bayes is a classification algorithm that employs the Bayes theorem (Equation 6) to predict the class, C_i of a given sample. The term 'naive' in its name signifies the method's assumption that all features are independent of each other.

$$P(C_i|x) = \frac{P(x|C_i) * P(C_i)}{P(x)} \tag{6}$$

3.2.4 K-Nearest Neighbors

K-Nearest Neighbors (KNN) stands as a supervised machine learning algorithm employed for both classification and regression tasks. The core concept revolves around identifying the k training samples that exhibit the closest proximity to a new data point. Subsequently, the algorithm leverages the class labels of these k neighboring samples to formulate predictions. Unlike models that explicitly construct a representation, KNN refrains from building a formal model and instead retains the entire dataset. Predictions are then generated based on the historical data stored within the system.

3.2.5 Implementation and Evaluation of Outcomes

The Python implementation was consistent across all methods, except for KNN. Utilizing the *sklearn* package, we partitioned our dataset into a training set (80%) and a test set (20%).

Subsequently, models were constructed and fitted to the data employing various methods, once again utilizing the *sklearn* package. For KNN, the model was initialized as previously described, with the additional step of determining the optimal K-value (K = 12), illustrated in Figure 6, as it showed best AUC performance.

After prediction, each of the models created a confusion matrix, represented bellow:

Table 4: Confusion Matrices of Classification Methods

LR	Actual	
	< 75	≥ 75
Predicted < 75	252	9
Predicted ≥ 75	0	0

LDA	Actual	
	< 75	≥ 75
Predicted < 75	246	5
Predicted ≥ 75	6	4

Naive Bayes	Actual	
	< 75	≥ 75
Predicted < 75	147	0
Predicted ≥ 75	105	9

12NN	Actual	
	< 75	≥ 75
Predicted < 75	252	9
Predicted ≥ 75	0	0

With this approach, we can assess the performance of each model by examining their respective ROC curves (Figure 1) and the corresponding AUC values (presented in Table 5).

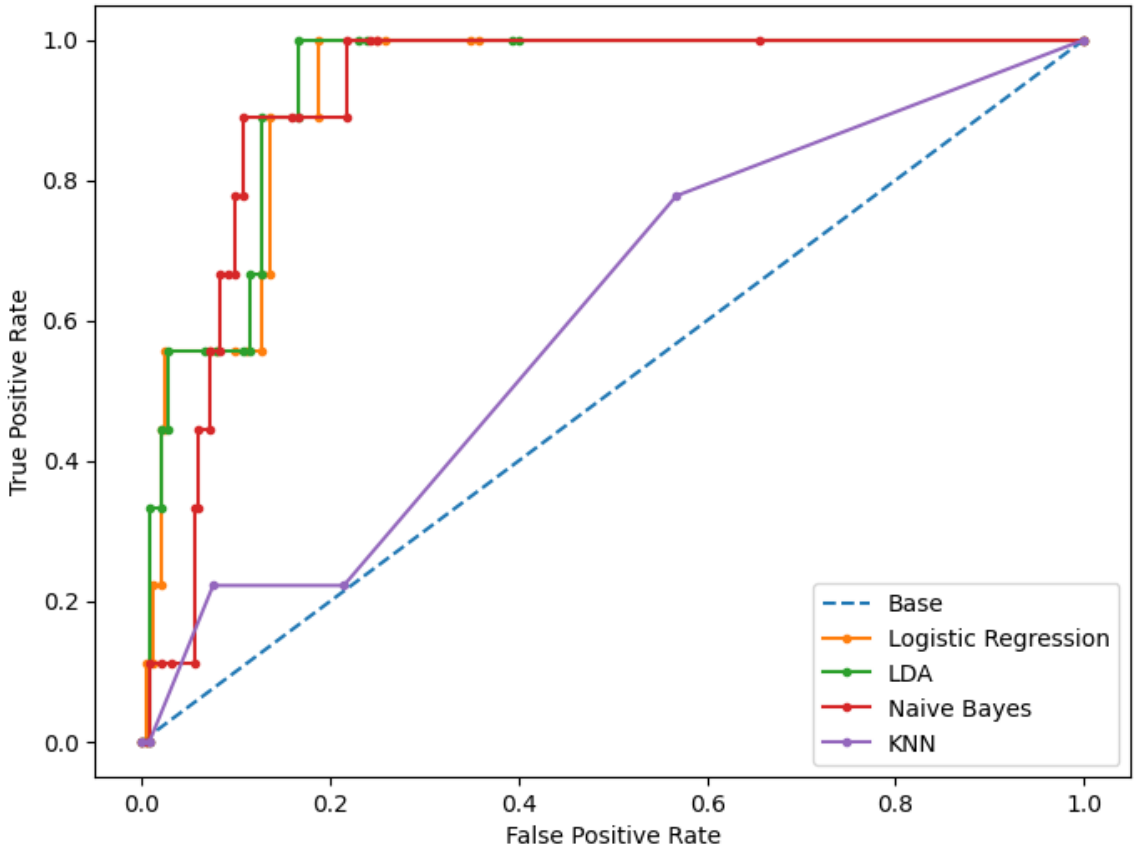

Figure 1: ROC Curves

Table 5: Model AUC and Error

Method	**AUC**	**Model Error (%)**
Logistic Regression	0,933	3,45
LDA	0,926	4,21
Naive Bayes	0,916	40,23
12NN	0,599	3,45

The K-Nearest Neighbors algorithm exhibits inferior performance, as anticipated, owing to its relative simplicity and susceptibility to errors. In contrast, the Logistic Regression algorithm outperforms other models.

Notably, the most recurrent mistake across models is the classification of false negatives, likely attributed to the set high threshold. The Naive Bayes algorithm also demonstrates a high model error, due to the assumption that all predictors are independent, which rarely happens in a dataset. After experimenting with different thresholds, maintaining the primary project goal of identifying characteristics that contribute to the popularity of a given track, it was determined that the selected threshold yields the most favorable results.

3.3 Tree-Based Methods

Tree-based methods represent a category of algorithms employed in prediction tasks, renowned for their interpretability. These methods build tree-like models that distinctly illustrate decision points and the pathway leading to the final prediction. In this structure, each internal node signifies a test on an attribute, each branch represents the outcome of the test, and each leaf node indicates a class label or a predicted value.

3.3.1 Decision Tree

The decision tree algorithm initiates with a single node, known as the root node, and iteratively partitions the data into subsets. Given the impracticality of computing every potential partition in the feature space, the method adopts a recursive binary splitting strategy. This approach is top-down, starting at the top and progressively moving down the tree, and is considered greedy, as the splitting decisions are made at each specific step without planning ahead.

At each iteration, the data is split based on the feature that yields the maximum information gain. The Gini index 7 serves as a metric for a node's purity, quantifying the probability of misclassification for a randomly selected sample drawn from the dataset.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{7}$$

3.3.2 Random Forest

The random forest algorithm enhances predictive performance by amalgamating multiple decision trees during the training phase. It produces the average prediction of the individual trees, thereby elevating the overall accuracy and resilience of the model. In this approach, when a split is required, only a random subset of m predictors is taken into account for that particular split, where m is less than p.

3.3.3 Gradient Boosting

Gradient boosting is an iterative method that constructs a series of decision trees, each refining its predictions based on the errors of the preceding iterations until reaching a desired level of accuracy. The residual errors are addressed by fitting a new tree to the negative gradient of the loss function. Subsequently, the weights of training examples are adjusted to minimize the overall loss.

Unlike aggressively fitting the data, which might lead to overfitting, this approach adopts a gradual learning process. In the end, it yields a composite model comprising multiple weak models, represented as:

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x) \tag{8}$$

3.3.4 Implementation and Evaluation of Outcomes

Each of the three tree-based methods in *sklearn* necessitates the provision of certain variable parameters: the Decision Tree requires the specification of the maximum depth of the tree; for the Random Forest, the number of estimators is crucial; and in the case of Gradient Boosting, both the number of estimators and the learning rate need to be supplied. To identify these optimal values, the *GridSearchCV* command was employed. This command conducts a thorough exploration for the best parameter combination within a given model by training and evaluating the model for all conceivable combinations of parameter values specified in a finite array. Its results are plotted in Figures 7, 8 and 9.

The evaluation of each model's performance involves conducting a 12-fold cross validation for each method. In each case, the process includes averaging the 12 resulting ROC curves and AUC values (Figures 2a, 2b and 2c; for greater detail, see Figures 10, 11 and 12).

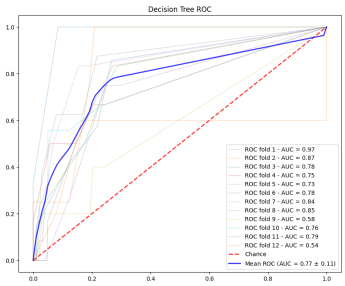

Figure 2a: Decision Tree ROC Curves

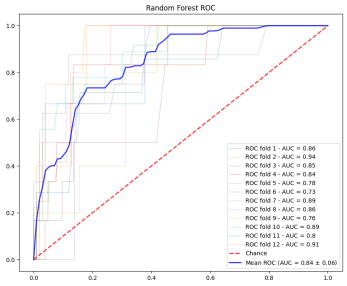

Figure 2b: Random Forest ROC Curves

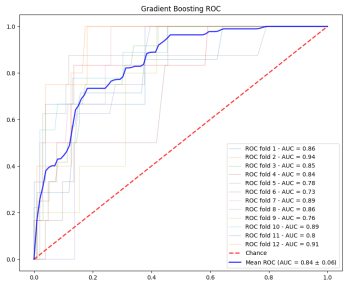

Figure 2c: Gradient Boosting ROC Curves

By visualization of the graphics above, it can be stated that the Random Forest and Gradient Boosting methods provide similar results, comparatively better than the Decision Tree. This can be explained due to the inability of capturing interactions between features and a high sensitivity to variations in the training data from the Decision Tree method.

A graph representation of the Decision Tree for Figure 2a is shown in figure 13.

3.4 Neural Networks

A neural network, a category of machine learning models, is crafted to emulate the structure and functionality of the human brain. Consisting of interconnected nodes known as neurons, these networks process and transmit information. Their primary objective is to learn non-linear functions that map input variables to output variables through training on a set of labeled examples.

For this task, a fully connected neural network, considered the simplest among feed-forward neural networks, has been chosen as the model. This selection is based on its perceived suitability for addressing the specific problem at hand.

Following the standardization of the data using MinMaxScaler, the subsequent phase in the neural network prediction involves training the model. The standardization process is employed to expedite model convergence and enhance performance. The neural network architecture incorporates two hidden layers, and experimentation will be conducted with

various sets of parameters, as outlined in Table 6. This iterative process aims to attain the highest level of accuracy for the model.

Table 6: Fully Connected NN Test Parameters

Set	1st Layer Neurons	2nd Layer Neurons	Learning Rate	Epochs	Batch Size
1	40	20	1e-4	500	32
2	200	100	1e-5	1500	16
3	10	5	1e-4	3000	32
4	100	50	1e-4	750	64

During the experimentation, a uniform selection of activation functions, specifically Rectified Linear Unit (ReLU) for the hidden layers and Sigmoid for the output neuron, was maintained. Additionally, the Adam optimizer served as the chosen optimization algorithm, utilizing the mean squared error (MSE) as the designated loss function.

The performance of four models was assessed by visualizing the progression of the loss across epochs for both the validation and test datasets. The validation loss serves as an indicator for potential overfitting, while the test loss assesses the model's generalization. Furthermore, comparing the loss values between the validation and test datasets aids in confirming the model's consistent performance.

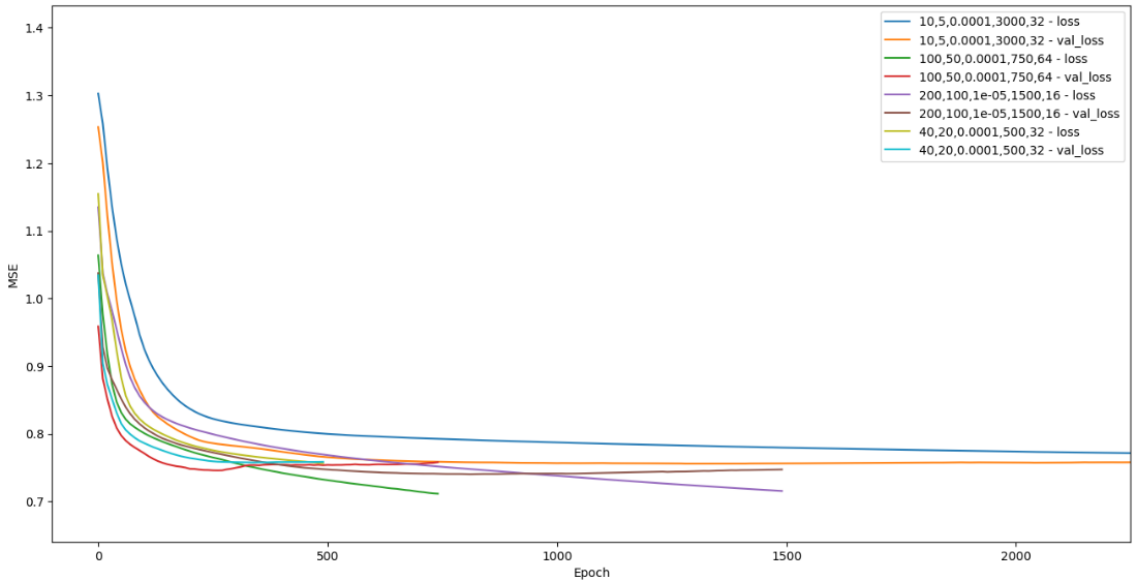

Figure 3: Evolution of Loss over the Number of Epochs

Upon analyzing the graph, it can be concluded that there are two most suitable models, depending on what is at stake: the one with the highest number of neurons per layer shows the highest decrease on losses at steady state, although with its lower learning rate it takes more epochs to be trained on. On the other hand, the one with the highest batch size decreases its loss more rapidly (it might be because the training data does not show much variance), in exchange for a comparably lower steady state loss.

Furthermore, the model with the lowest number of neurons, despite being trained for a

larger number of epochs, does not decrease its losses in steady state as much as the others do.

The result is a set of discrete variables. In Figure 4, these outcomes are depicted along with their corresponding residuals (y-axis). Despite a few outliers, we can observe similarity among the models.

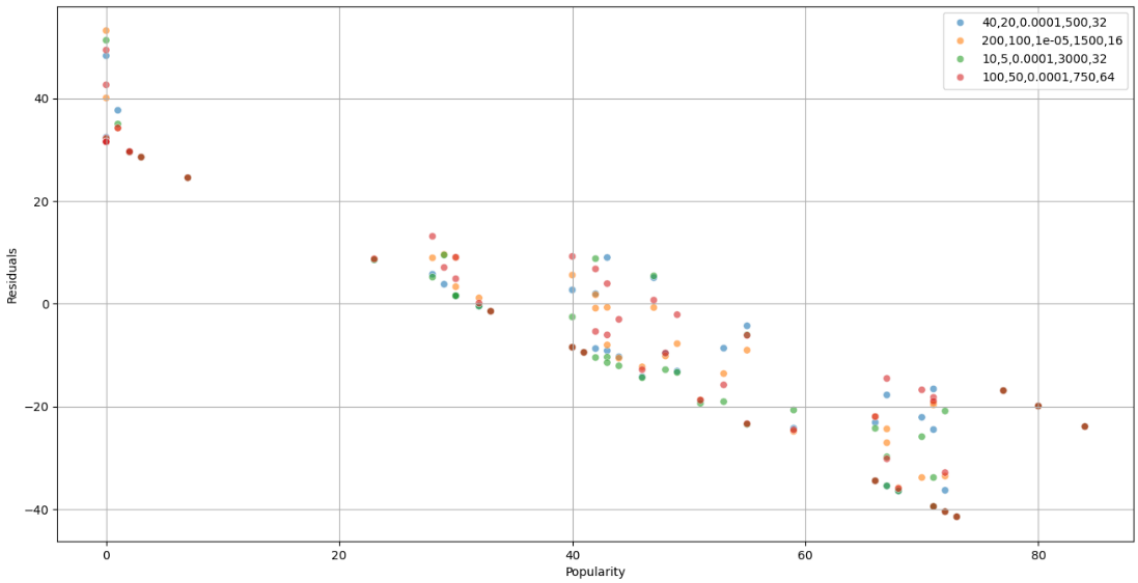

Figure 4: Predicted Outcome and Corresponding Residual

4 Conclusion

In summary, the examined methods yielded satisfactory outcomes. Linear regression, for instance, favored simplicity over precision, whereas tree-based techniques placed a premium on interpretability. More intricate methodologies like classification and neural networks demonstrated the potential for superior results. However, it is crucial to bear in mind that the selection of a method should also account for ease of implementation and the specific characteristics of the analyzed data.

Appendix

All the material can be found on the GitHub repository, by visiting https://github.com/Joao-Rosado/AA_Project.

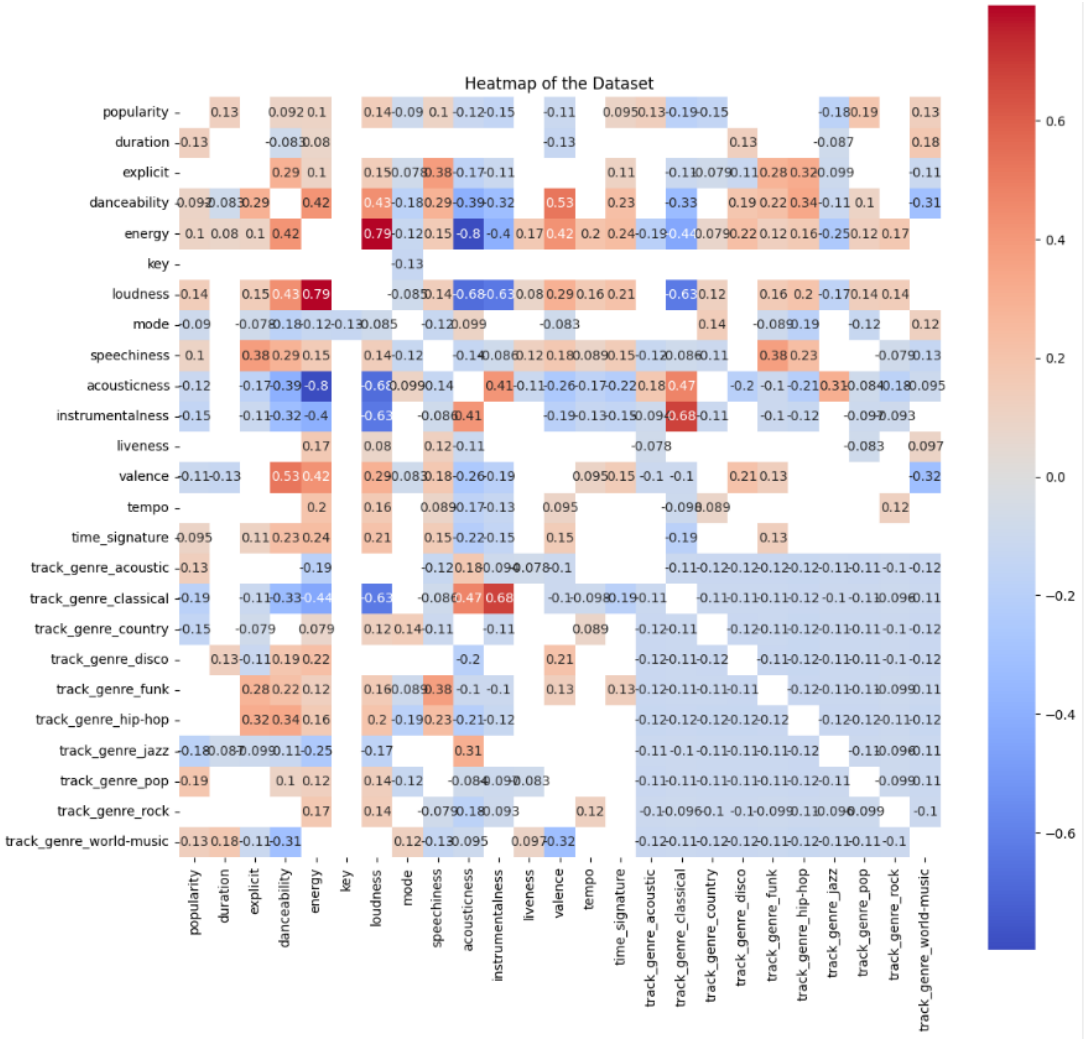

Figure 5: Correlation Matrix

Table 7: Coefficients and Statistics for the Multiple Linear Regression Model

Variable	**Coef**	**Std. Error**	**t**	$P > \|t\|$	**[0,025**	**0,975]**
const	28.7539	6.173	4.658	0.000	16.643	40.865
duration	0.0251	0.006	4.439	0.000	0.014	0.036
energy	-3.9959	6.348	-0.630	0.529	-16.449	8.457
loudness	-0.0329	0.292	-0.113	0.910	-0.606	0.540
acousticness	3.8199	3.809	1.003	0.316	-3.653	11.292
instrumentalness	-1.8873	4.527	-0.417	0.677	-10.769	6.995
track_genre_classical	-21.6496	3.927	-5.514	0.000	-29.353	-13.947
track_genre_jazz	-19.5438	2.851	-6.854	0.000	-25.138	-13.950
track_genre_pop	14.5196	2.555	5.684	0.000	9.508	19.531

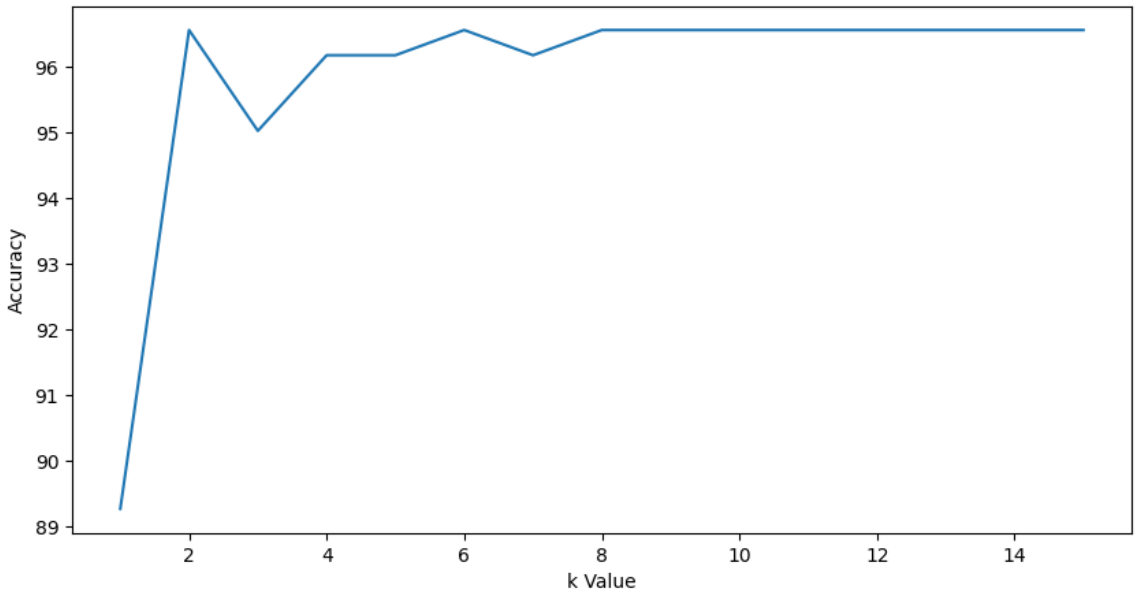

Figure 6: Accuracy for different K-values in KNN

Table 8: Coefficients and Statistics for the Multiple Linear Regression Model with Interactions

Variable	**Coef**	**Std. Error**	**t**	$P > \|t\|$	**[0,025**	**0,975]**
const	-33.2838	19.122	-1.741	0.082	-70.799	4.231
x1	0.2217	0.064	3.453	0.001	0.096	0.348
x2	53.5516	19.630	2.728	0.006	15.041	92.063
x3	-3.8251	1.474	-2.594	0.010	-6.718	-0.932
x4	51.3581	21.538	2.385	0.017	9.104	93.612
x5	23.5906	39.116	0.603	0.547	-53.148	100.329
x6	-60.9918	35.524	-1.717	0.086	-130.684	8.700
x7	-28.6907	31.031	-0.925	0.355	-89.568	32.187
x8	-34.0365	25.323	-1.344	0.179	-83.715	15.642
x9	-0.2076	0.061	-3.426	0.001	-0.327	-0.089
x10	0.0033	0.003	1.108	0.268	-0.003	0.009
x11	-0.0986	0.043	-2.269	0.023	-0.184	-0.013
x12	-0.0442	0.041	-1.076	0.282	-0.125	0.036
x13	0.0085	0.025	0.336	0.737	-0.041	0.058
x14	0.0899	0.037	2.448	0.014	0.018	0.162
x15	-0.0182	0.047	-0.384	0.701	-0.111	0.075
x16	2.5612	1.466	1.747	0.081	-0.315	5.438
x17	9.6234	19.207	0.501	0.616	-28.057	47.304
x18	-8.4172	38.850	-0.217	0.829	-84.634	67.800
x19	65.6577	34.260	1.916	0.056	-1.555	132.870
x20	-9.7874	30.805	-0.318	0.751	-70.222	50.647
x21	55.0611	24.469	2.250	0.025	7.056	103.066
x22	3.7784	1.497	2.524	0.012	0.841	6.715
x23	0.3153	0.985	0.320	0.749	-1.616	2.247
x24	-0.9848	1.142	-0.862	0.389	-3.226	1.256
x25	-0.9946	1.139	-0.874	0.382	-3.228	1.239
x26	-2.1071	1.543	-1.365	0.172	-5.135	0.920
x27	-0.5345	26.518	-0.020	0.984	-52.558	51.489
x28	9.9262	21.661	0.458	0.647	-32.569	52.422
x29	-21.0542	17.581	-1.198	0.231	-55.546	13.437
x30	16.4934	13.240	1.246	0.213	-9.482	42.469
x31	-0.1347	13.229	-0.010	0.992	-26.088	25.818
x32	-5.0404	14.127	-0.357	0.721	-32.756	22.675
x33	-107.3627	51.289	-2.093	0.037	-207.983	-6.743
x34	0	0	nan	nan	0	0
x35	0	0	nan	nan	0	0
x36	0	0	nan	nan	0	0

Table 9: Coefficients and Statistics for the Multiple Linear Regression Model - Quadratic Terms

Variable	**Coef**	**Std. Error**	**t**	$P > \|t\|$	**[0,025**	**0,975]**
const	0	0	nan	nan	0	0
x1	0.1423	0.061	2.333	0.020	0.023	0.262
x2	-13.1255	22.344	-0.587	0.557	-56.960	30.709
x3	-3.5650	1.557	-2.289	0.022	-6.620	-0.510
x4	12.2247	29.046	0.421	0.674	-44.760	69.209
x5	21.8508	44.136	0.495	0.621	-64.737	108.439
x6	-28.7571	18.368	-1.566	0.118	-64.792	7.278
x7	-8.4897	15.837	-0.536	0.592	-39.559	22.580
x8	-14.0511	12.694	-1.107	0.269	-38.956	10.853
x9	-2.774e-05	8.14e-06	-3.407	0.001	-4.37e-05	-1.18e-05
x10	-0.0496	0.066	-0.747	0.455	-0.180	0.081
x11	0.0012	0.003	0.419	0.675	-0.005	0.007
x12	-0.0828	0.043	-1.940	0.053	-0.167	0.001
x13	-0.0595	0.042	-1.411	0.159	-0.142	0.023
x14	0.0336	0.027	1.263	0.207	-0.019	0.086
x15	0.0722	0.037	1.972	0.049	0.000	0.144
x16	-0.0408	0.048	-0.852	0.394	-0.135	0.053
x17	16.8569	22.102	0.763	0.446	-26.504	60.217
x18	2.5148	1.718	1.464	0.144	-0.856	5.886
x19	40.9322	28.079	1.458	0.145	-14.155	96.019
x20	4.9446	42.919	0.115	0.908	-79.257	89.146
x21	48.9911	35.306	1.388	0.166	-20.274	118.257
x22	-17.4406	31.328	-0.557	0.578	-78.902	44.021
x23	54.7045	24.525	2.231	0.026	6.590	102.819
x24	-0.0347	0.054	-0.637	0.524	-0.142	0.072
x25	3.1412	1.363	2.304	0.021	0.467	5.816
x26	-0.6998	1.297	-0.539	0.590	-3.245	1.845
x27	-0.9453	1.182	-0.800	0.424	-3.264	1.373
x28	-0.9640	1.141	-0.845	0.398	-3.202	1.274
x29	-2.0827	1.542	-1.350	0.177	-5.109	0.943
x30	17.7341	17.809	0.996	0.320	-17.205	52.673
x31	5.4197	27.020	0.201	0.841	-47.589	58.428
x32	5.3434	22.846	0.234	0.815	-39.476	50.163
x33	-29.0245	18.375	-1.580	0.114	-65.074	7.025
x34	14.8522	13.235	1.122	0.262	-11.113	40.818
x35	-20.9804	26.485	-0.792	0.428	-72.939	30.979
x36	-4.1224	13.988	-0.295	0.768	-31.564	23.319
x37	-6.2045	14.325	-0.433	0.665	-34.307	21.898
x38	-112.8315	52.322	-2.156	0.031	-215.478	-10.185
x39	-28.7571	18.368	-1.566	0.118	-64.792	7.278
x40	0	0	nan	nan	0	0
x41	-8.4897	15.837	-0.536	0.592	-39.559	22.580
x42	0	0	nan	nan	0	0
x43	-14.0511	12.694	-1.107	0.269	-38.956	10.853

Table 10: Logistic Regression Coefficients

Coef	Value	Coef	Value
β_0	-1.34342771	β_{13}	-9.01160548e-03
β_1	9.61571551e-04	β_{14}	1.07358001e-01
β_2	1.27641446e+00	β_{15}	-1.17325297e+00
β_3	1.31960090e-01	β_{16}	-2.63051748e-01
β_4	-4.95769403e-01	β_{17}	4.31580202e-01
β_5	-3.54466615e-02	β_{18}	-7.08724875e-01
β_6	6.88172685e-02	β_{19}	-3.56308923e-01
β_7	8.17742314e-02	β_{20}	5.79501775e-01
β_8	1.20610619e-01	β_{21}	-8.91940081e-01
β_9	-8.30871143e-01	β_{22}	1.39064651e+00
β_{10}	7.59560875e-02	β_{23}	2.13382881e+00
β_{11}	-7.16995391e-01	β_{24}	-1.23230827e+00
β_{12}	-6.79986106e-01		

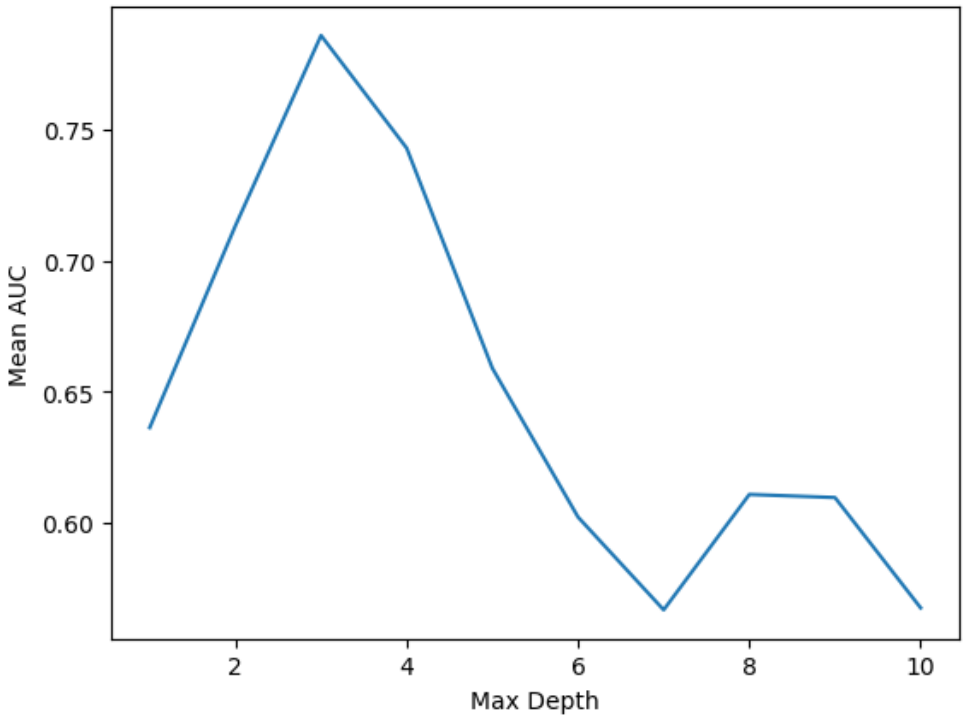

Figure 7: Decision Tree Grid Search

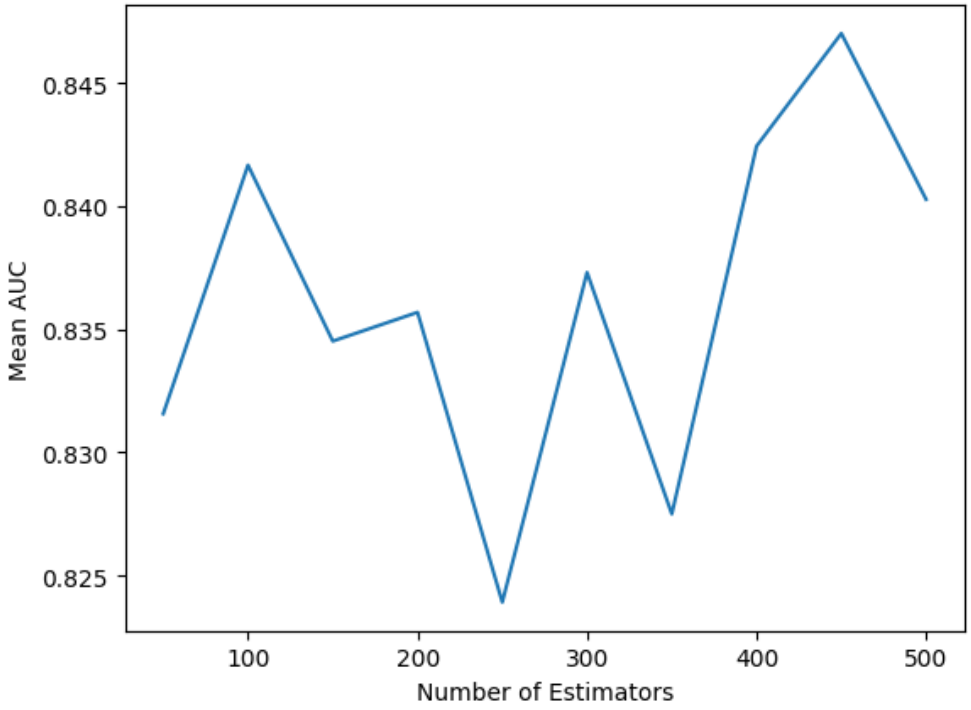

Figure 8: Random Forest Grid Search

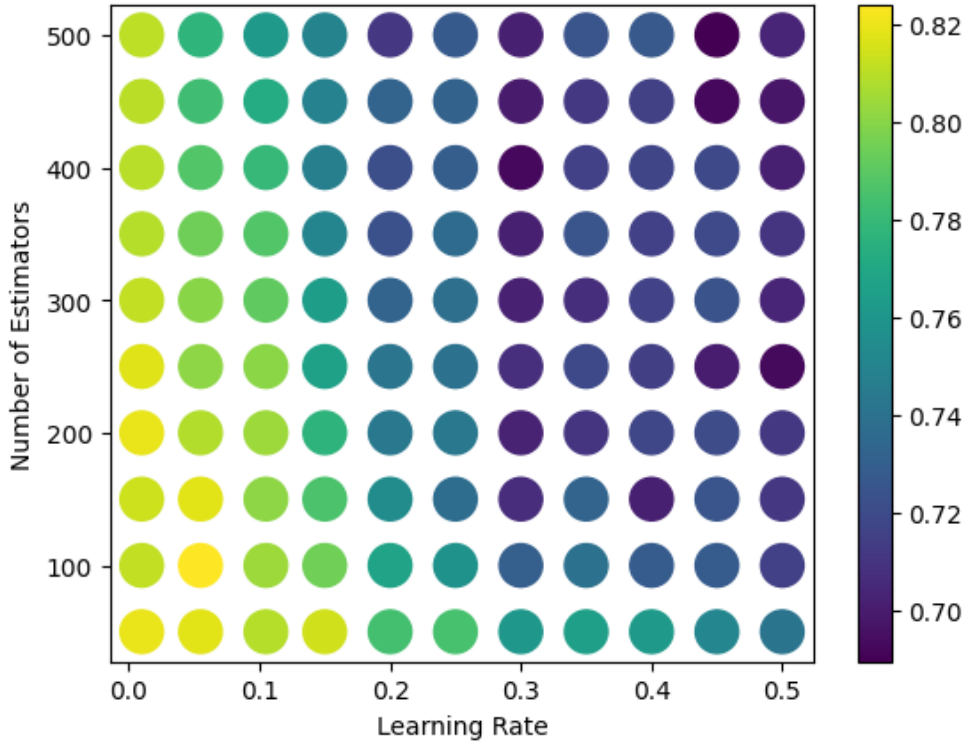

Figure 9: Gradient Boosting Grid Search (Colour Gradient is AUC)

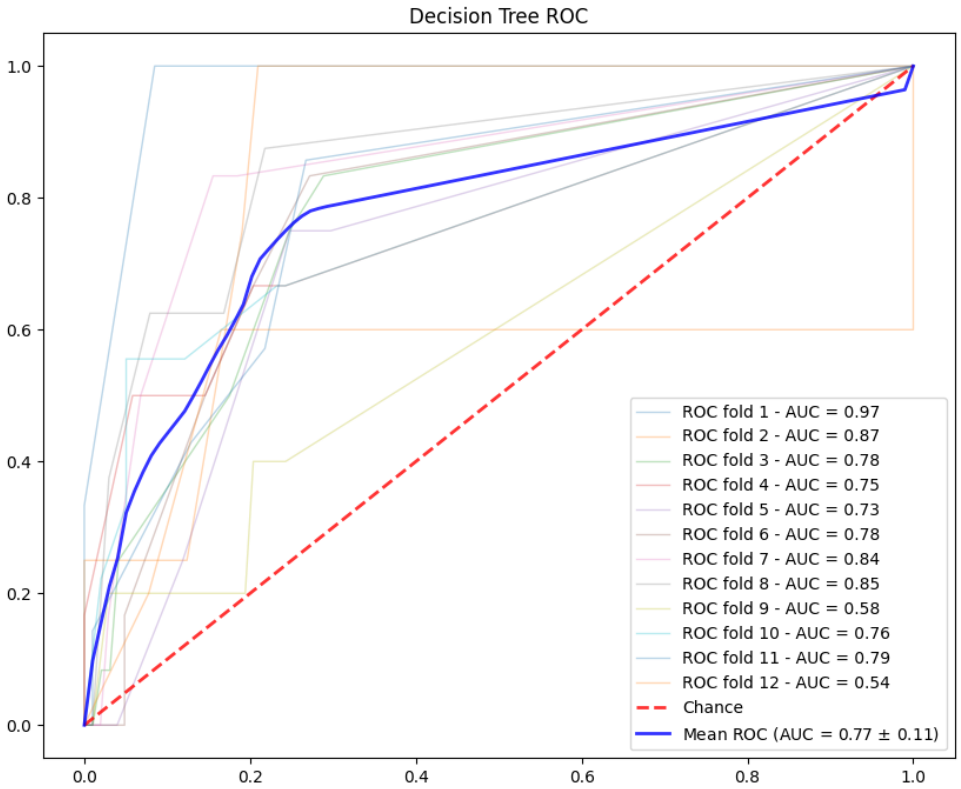

Figure 10: Decision Tree ROC Curves - Amplified

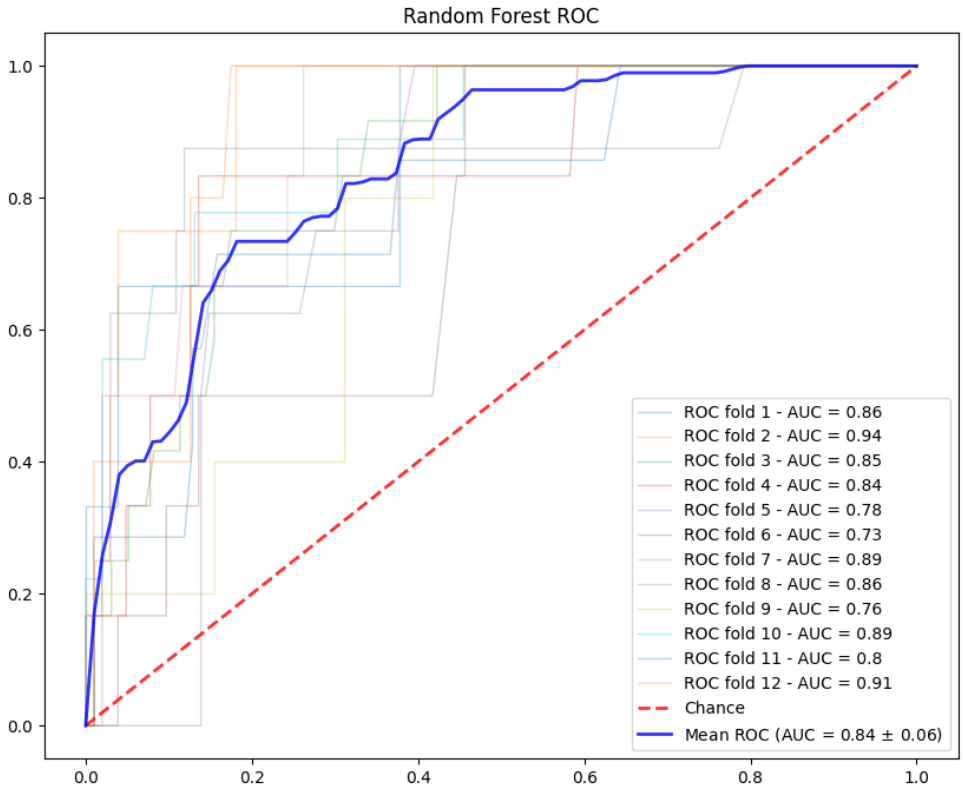

Figure 11: Random Forest ROC Curves - Amplified

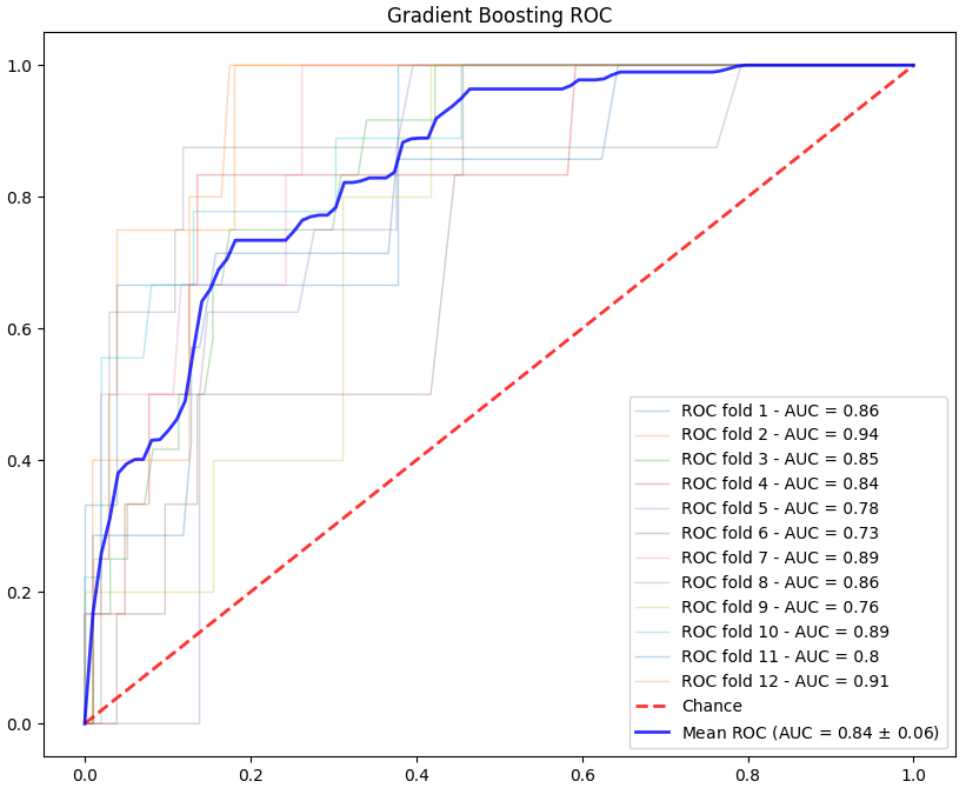

Figure 12: Gradient Boosting ROC Curves - Amplified

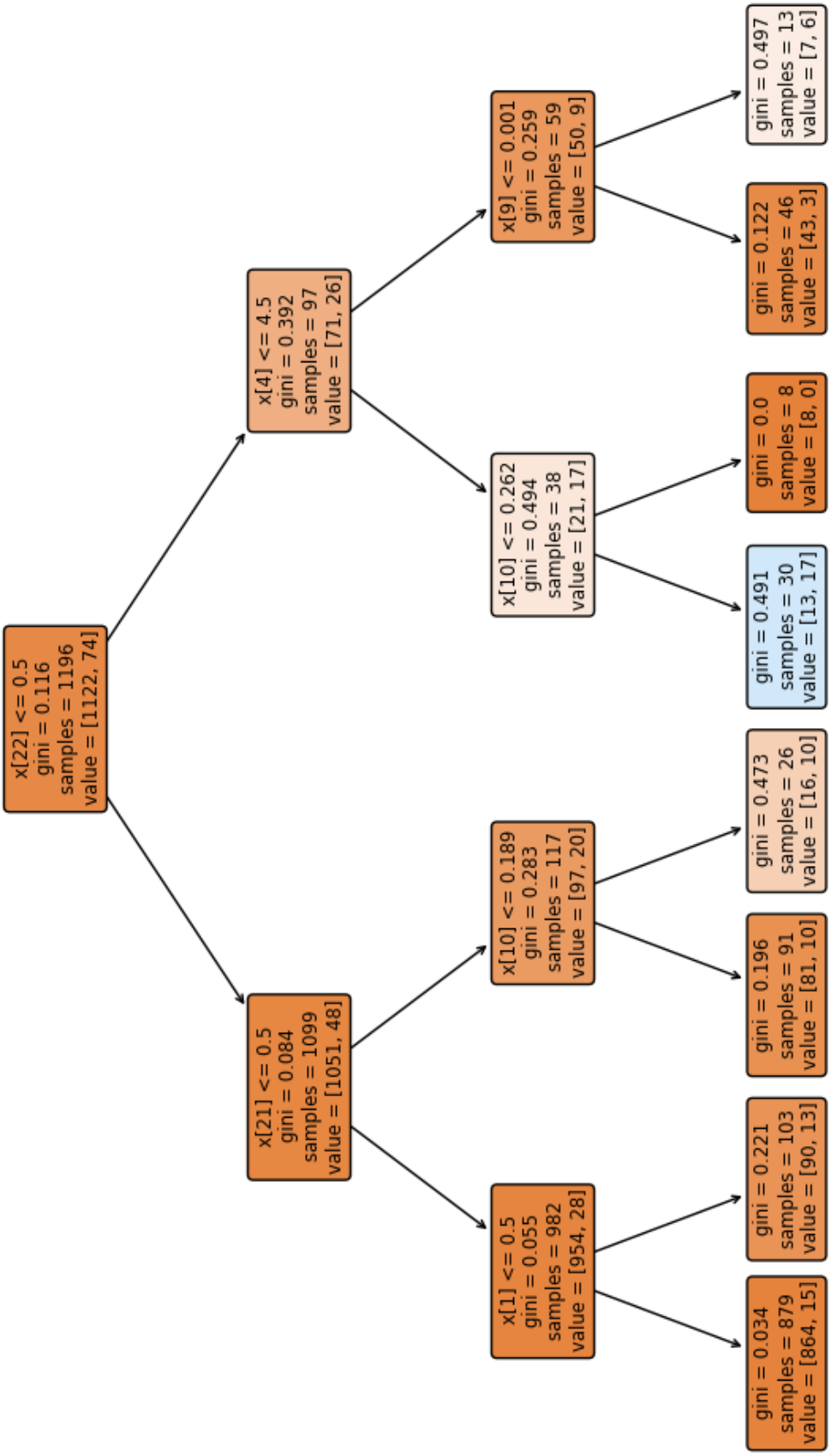

Figure 13: Decision Tree