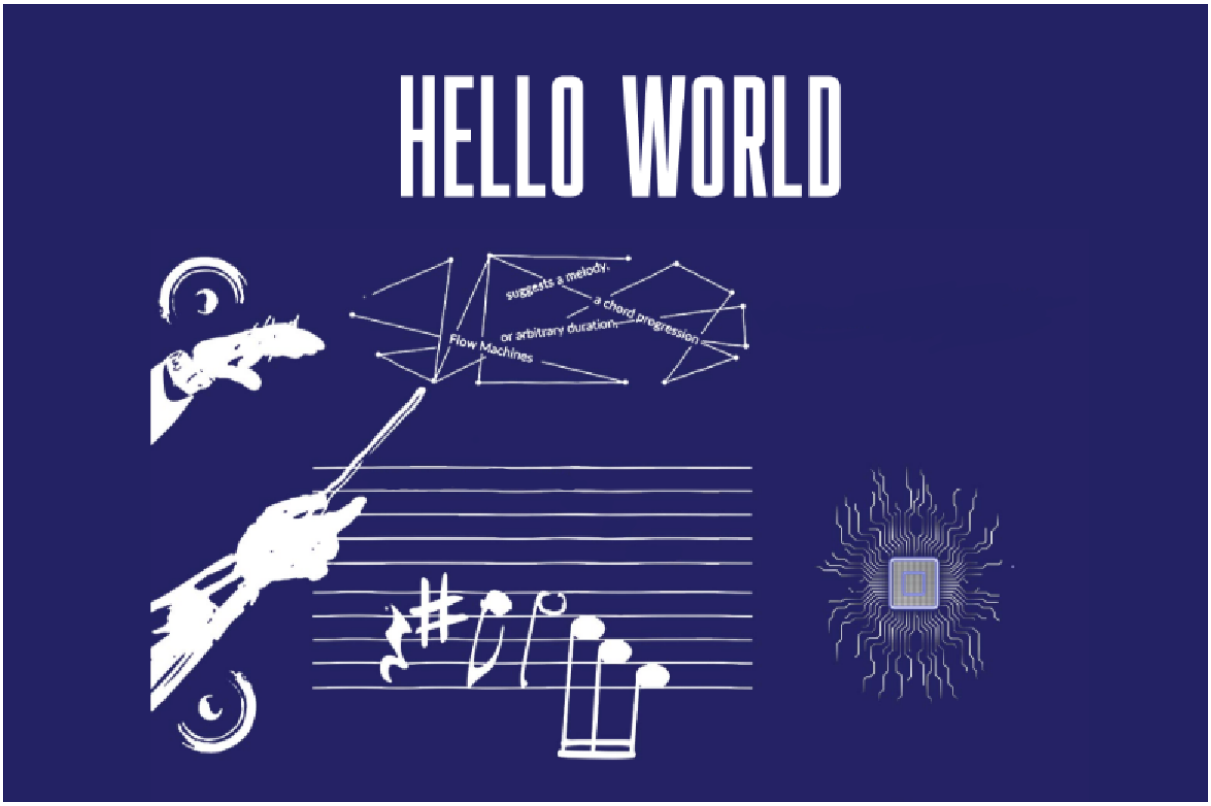
SONG GENRE CLASSIFICATION

APA Machine Learning Project



Pere Ginebra, Joan Domingo
APA Q1 2021 – 2022
FIB – UPC

TABLE OF CONTENTS

1.Introduction

1.1. Dataset description

The dataset chosen for this practical work is a set of 50.000 audio features provided by Spotify. It has 18 variables, such as popularity, acousticness or energy, including the attribute we are going to predict: the music genre of each song.

More deeply, our original dataset has 11 numerical variables:

- instance_id: unique ID for each song,
- popularity: how popular is this song, between 0 and 100,
- acousticness: acousticness level, between 0 and 1,
- danceability: danceability level, between 0 and 1,
- duration_ms: duration of the song in ms,
- energy: energy level between, 0 and 1,
- instrumentalness:instrumentalness level, between 0 and 1,
- liveness: liveness level, between 0 and 1,
- loudness: loudness level, between -47 and 3.74,
- speechiness: number of words in the song, between 0 and 1,
- valence: musical positiveness, between 0 and 1,

6 categorical:

- artist_name: song's composer name,
- track_name: name of the song,
- key: song key (A, A#, B, C, C#, D, D#, E, F, F#, G or G#),
- mode: song mode (Major or Minor),
- tempo: song tempo,
- music_genre: song genre, our target variable to predict (Electronic, Anime, Jazz, Alternative, Country, Rap, Blues, Rock, Classical or Hip-Hop),

and 1 time related variable (obtained_date).

Dataset source: https://www.kaggle.com/vicsuperman/prediction-of-music-genre

1.2. Initial problems encountered

Since the variable to predict is categorical, it's obvious that we will be facing a classification problem. One of the first issues we can find by just looking at the unprocessed dataset is that there are some variables that don't give information about the song genre, such as the track or the artist name, the obtained date of the song or it's ID. We can also think that attributes such as valence or danceability are quantified by an algorism, since they are very subjective. The last initial problem we encountered is that some songs may be classified as different genres, since music fusion is more popular every day. For example, the category Rap and Hip-Hop can be hardly distinguishable, and there are a lot of songs which can be classified as Alternative-Rock or Jazz-Rap. The Alternative music genre seems a bit difficult to classify as well, since the label "alternative" means that the songs are produced by performers who are outside the musical mainstream, so it seems that is not a musical genre per se. With all these issues in mind, we will start the preprocessing of the chosen dataset.

2. Preprocessing

2.1. Feature Selection

The first step is to load the dataset, which will be stored in a python variable named music_data. As we have said in section 1.2, we will drop the instance_id, artist_name, track_name and obtained_date columns, and then we will take a look at the dataset description and first rows.

	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	valence
count	50000.000000	50000.000000	50000.000000	5.000000e+04	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000
mean	44.220420	0.306383	0.558241	2.212526e+05	0.599755	0.181601	0.193896	-9.133761	0.093586	0.456264
std	15.542008	0.341340	0.178632	1.286720e+05	0.264559	0.325409	0.161637	6.162990	0.101373	0.247119
min	0.000000	0.000000	0.059600	-1.000000e+00	0.000792	0.000000	0.009670	-47.046000	0.022300	0.000000
25%	34.000000	0.020000	0.442000	1.748000e+05	0.433000	0.000000	0.096900	-10.860000	0.036100	0.257000
50%	45.000000	0.144000	0.568000	2.192810e+05	0.643000	0.000158	0.126000	-7.276500	0.048900	0.448000
75%	56.000000	0.552000	0.687000	2.686122e+05	0.815000	0.155000	0.244000	-5.173000	0.098525	0.648000
max	99.000000	0.996000	0.986000	4.830606e+06	0.999000	0.996000	1.000000	3.744000	0.942000	0.992000

2.2. Missing values

We found that some descriptive variables have missing values, or equivalent values such as '?', ' ' or '-1'. Converting them to NaN, we see that there are 5 rows with all missing values and, since they give no information about the music genre and can't be predicted, we will directly remove those rows, leaving the dataset with 49.995 rows to work on.

```
Missing Values
 popularity          5
acousticness         5
danceability         5
duration_ms       4944
energy               5
instrumentalness     5
key                  5
liveness             5
loudness             5
mode                 5
speechiness          5
tempo             4985
valence              5
music_genre          5
```

The variables duration_ms and tempo are a bit conflictive, since they have about 5.000 missing values each one. Luckily, the missing values are seemingly equally distributed through all 10 categories of our target variable music_genre, with approximately 500 'NaN's each.

After the treating of the categorical variables and of the outliers we will try to impute those missing values using a KNN-Imputer.

2.3. Treat categorical attributes

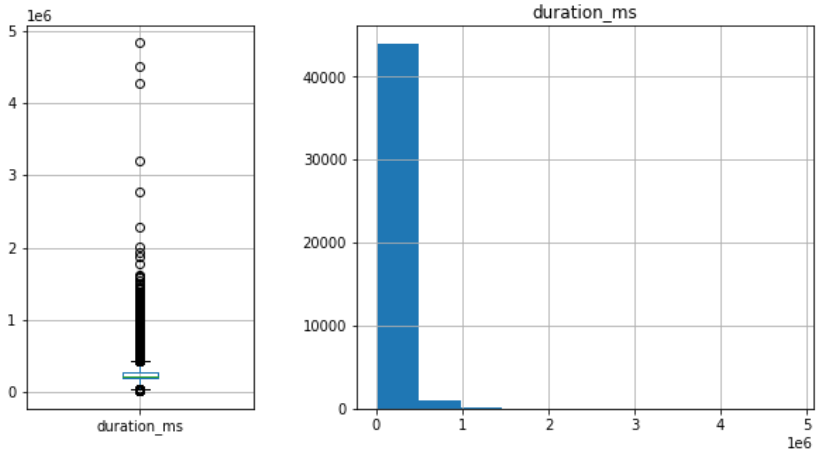
The next step of our preprocessing is to treat the categorical variables, converting them into numerical variables. Since tempo values are real numbers, we convert the variable from its 'object' type to a numerical (float) one. We convert the variable mode to a numerical binary variable (transforming Minor/Major to 0/1) and the variable key to 12 binary variables that describe the key. This is done so we don't get any trouble with specific models that can't handle strings/objects.

2.4. Outliers

Most attributes in our dataset are calculated and range from 0 to 1 or from 0 to 100, so they generally do not need special outlier treatment. Tempo, loudness and duration are the only ones that might need treatment. From an initial analysis over their ranges we can extract the following hypotheses:

- Tempo: ranges from 34 to 220 bpm, which is a reasonable range for music.
- Loudness: ranges from -47 to +3 dB, has a mean of -9 and std of 6, so the lower values could be treated as outliers (or strange cases of very quiet music).
- Duration: ranges from 155s (2.6min) to 4497s (74.95 min) which also seems somewhat reasonable, but the upper threshold could also be treated as outliers.

From a further study with boxplots, histograms and distributions we can see that, as predicted, tempo has a pretty good distribution and not many outlier candidates. Loudness, on the other hand, seems to have some extreme cases, but it can be justified as very quiet songs of the Classical category. If we check the songs at this extreme we notice that these values still carry a great deal of information as most extremely quiet songs belong to the classical genre.



Analysing song durations is where we find the first clear cases of outliers. These are probably correct values, but they look to be special cases that do not represent their genre correctly and might affect our models, so we will drop them.

2.5. Impute missing values

Using the *KNNImputer* from *sklearn*, we will fit and transform a KNN model with k=5 to replace the duration and tempo missing values for predicted ones.

2.6. Train-test split

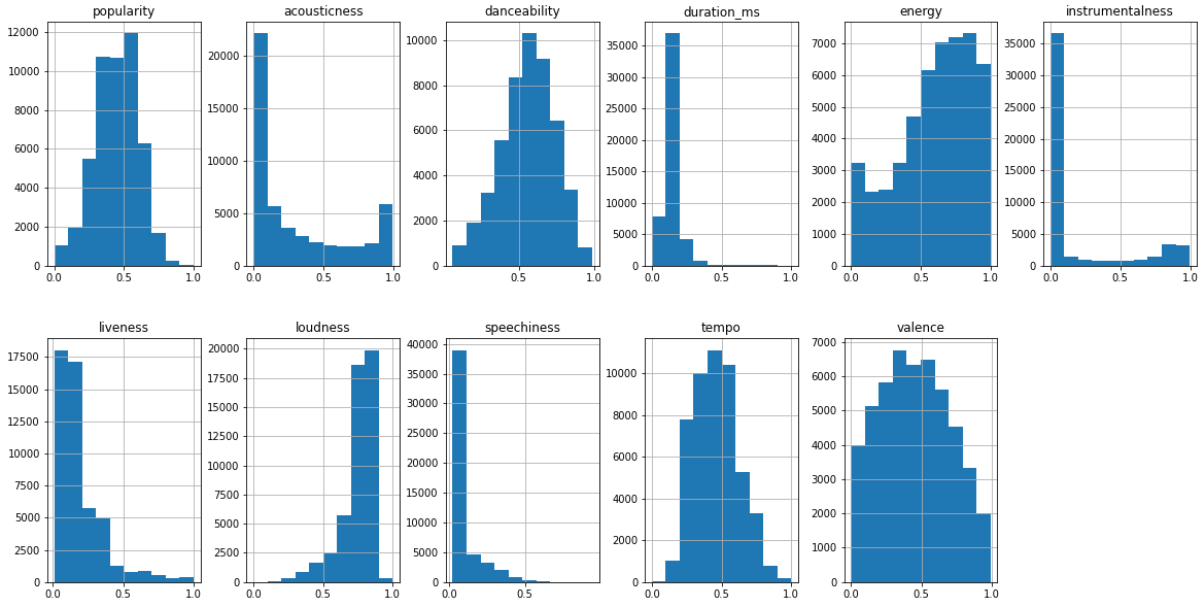
As we approach the time to start creating classification models, we need to split the data into train and test partitions using *scikit-learn*'s *train_test_split()* function. We chose to do a 75-25 split, so the test data represents 25% of all the processed dataset rows.

2.7. Normalize variables

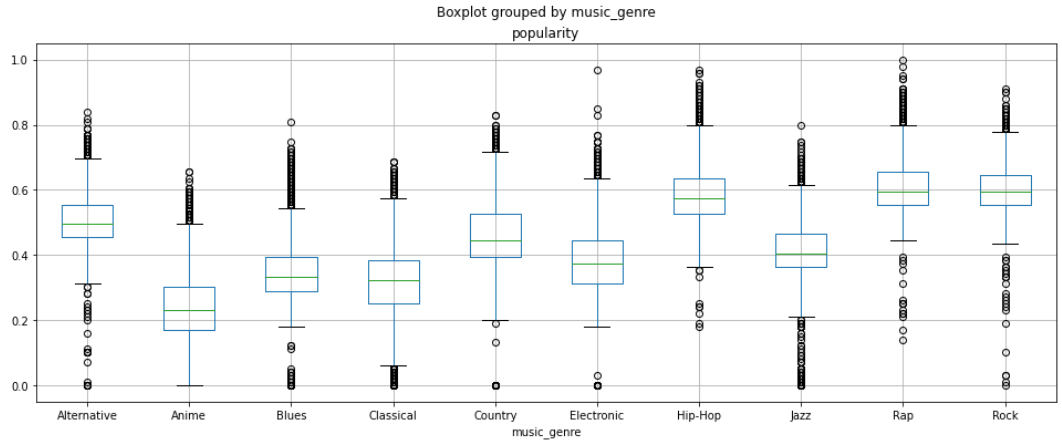
Since most of our variables are already normalized to the range [0,1] we will only have to manually normalize popularity, duration_ms, loudness and tempo. These will be normalized using a min-max scaler computed from the train-split data.

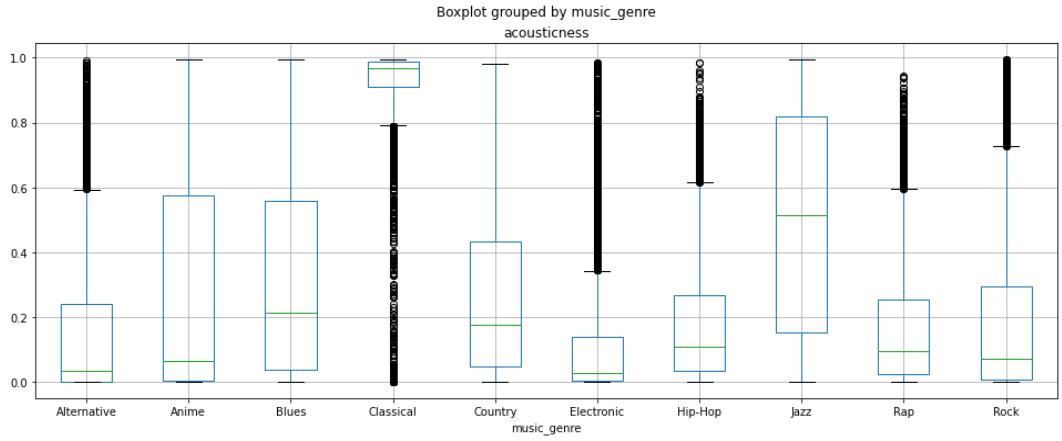
2.8. Final analysis of our preprocessed dataset

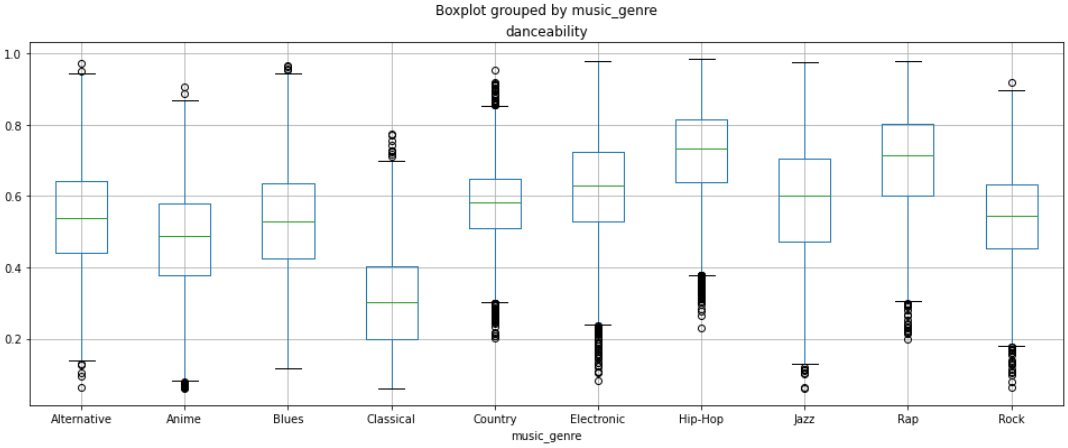
After we have finished preprocessing our data we can take a final look at our dataset before we start modeling and predicting. First we can look at the distribution of each variable independently from the genre.

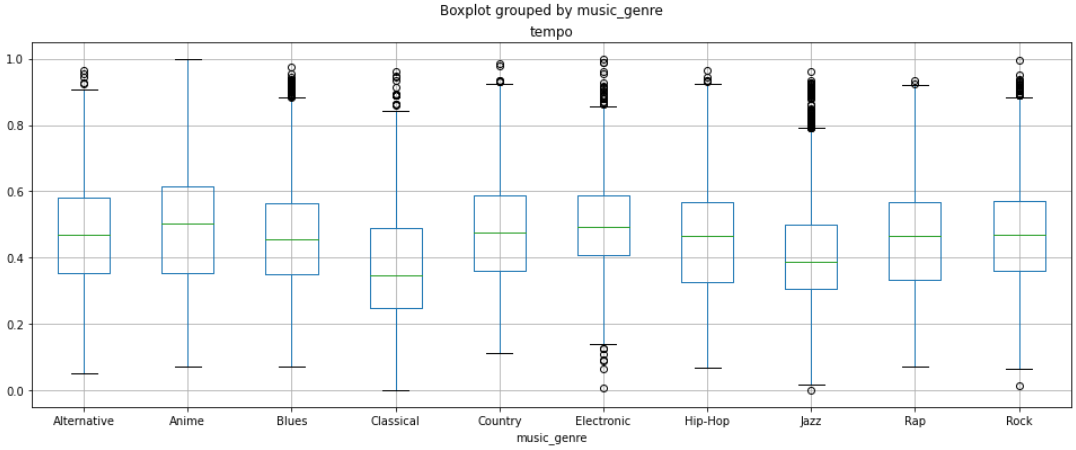


Now we can see the boxplot distribution of some of these attributes depending on a song's genre:

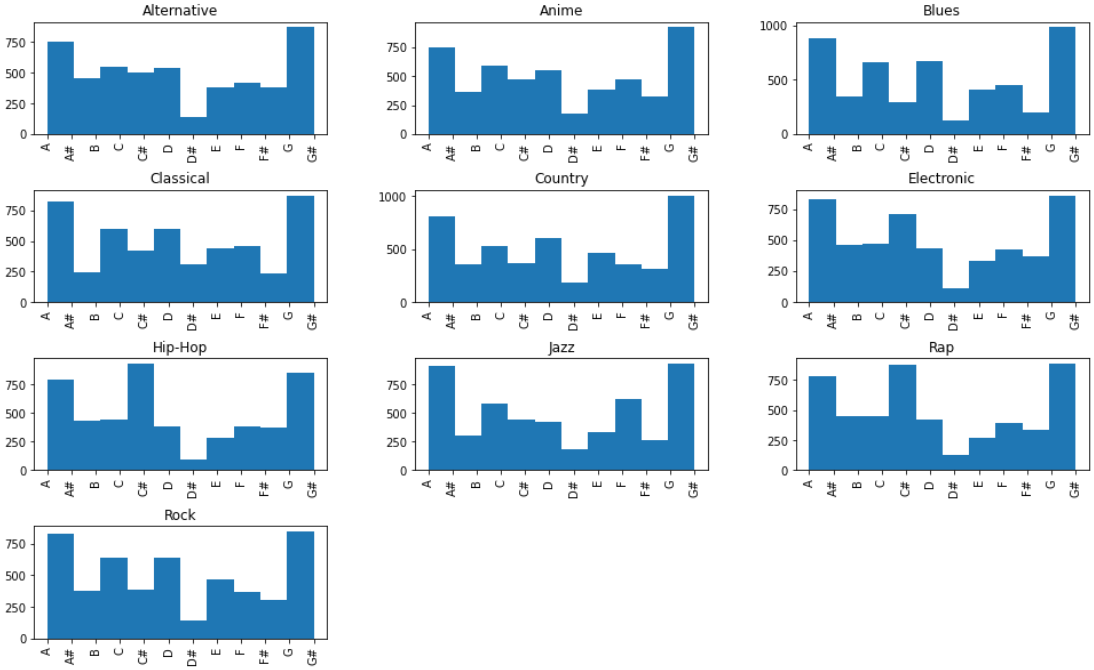

Boxplot grouped by music_genre
popularity
1.0
0.8
0.6
0.4
0.2
0.0
Alternative
Anime
Blues
Classical
Country
Electronic
Hip-Hop
Jazz
Rap
Rock
music_genre


Boxplot grouped by music_genre
acousticness
1.0
0.8
0.6
0.4
0.2
0.0
Alternative
Anime
Blues
Classical
Country
Electronic
Hip-Hop
Jazz
Rap
Rock
music_genre

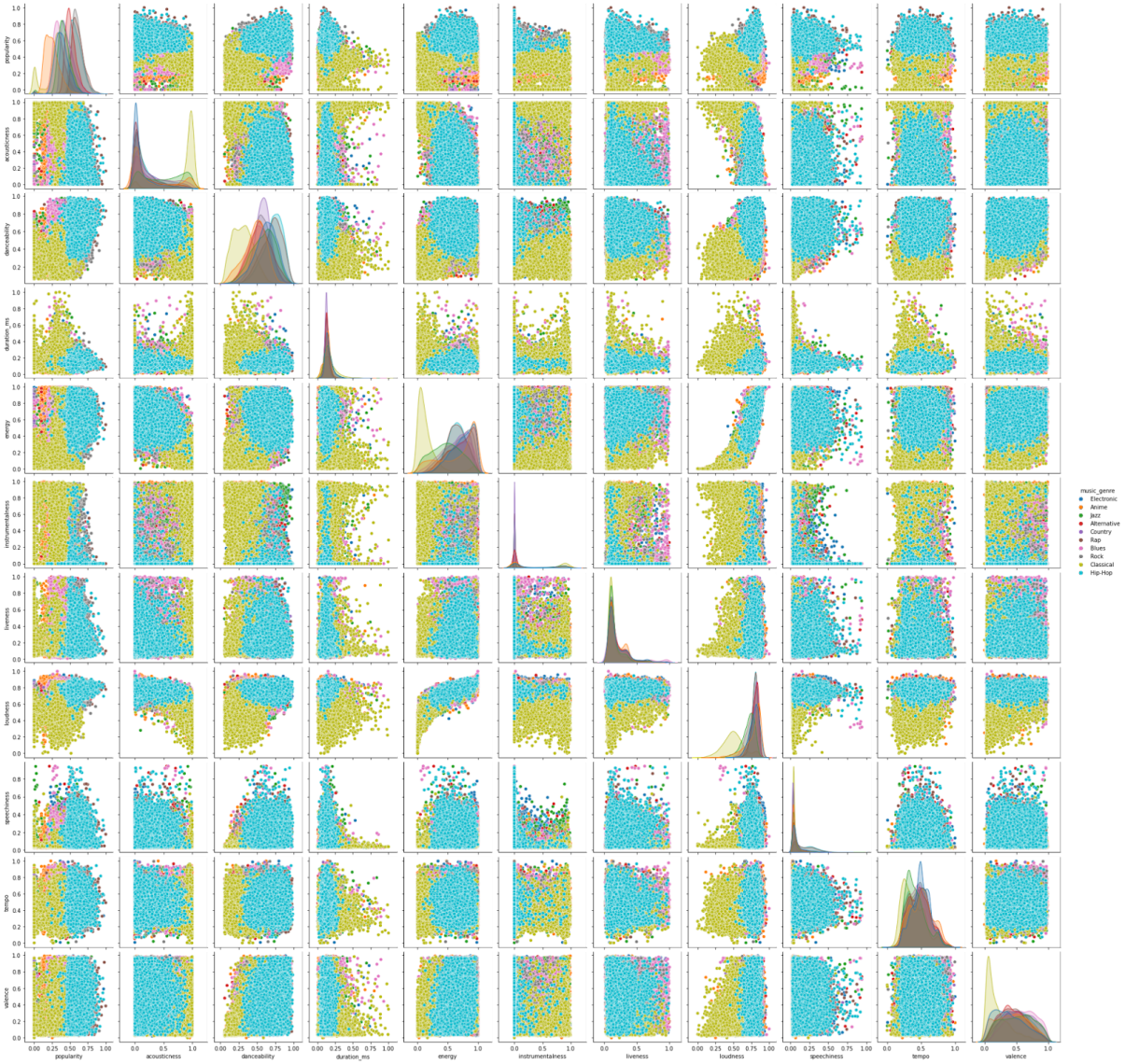

Boxplot grouped by music_genre
danceability
1.0
0.8
0.6
0.4
0.2
Alternative
Anime
Blues
Classical
Country
Electronic
Hip-Hop
Jazz
Rap
Rock
music_genre


Boxplot grouped by music_genre
tempo
1.0
0.8
0.6
0.4
0.2
0.0
Alternative
Anime
Blues
Classical
Country
Electronic
Hip-Hop
Jazz
Rap
Rock
music_genre

And also the histogram distribution of key of each genre:

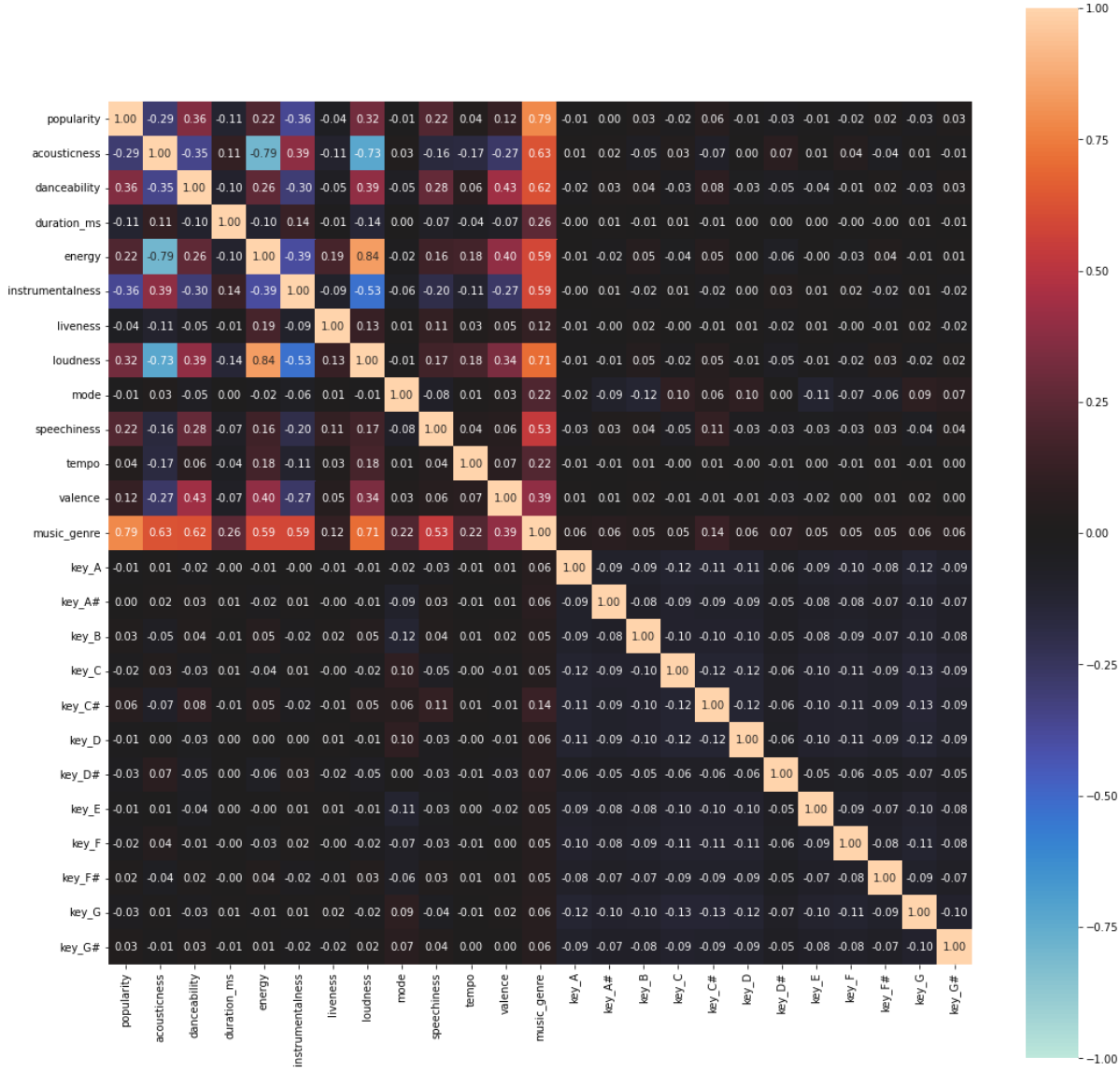


With these we can start to see which attributes are more relevant to each genre and which don't vary much at all. This will help us understand our dataset a bit better, give us a first insight into what variables might be the most helpful and show us what genres might end up being the most troublesome.



Even though the image is too big to read well here, we can see that some pairs of variables slightly describe the difference between some genres, but we won't be able to accurately classify them from any of these pairs alone.

Before starting to modelize classifiers for our preprocessed dataset, we can check the correlation between all our variables. This will give us an initial point of view to understand how some models must be built and to check if the model's weights make enough sense.



The result table shows some interesting results. The key variables are not very correlated to any variable in a strong way, so they will not be too important in our future models.

Apart from this, all other variables seem to have some relation to music_genre. In particular, the most correlated variables are popularity, loudness, acousticness and danceability. As opposed to those variables, the least related variable is liveness.

3. Testing Linear/quadratic Classifiers

To analyze and compare results between different models, apart from using a train-test split, we will also use a 5-fold Cross Validation as the resampling protocol when calculating the cross validation scores with *scikit-learn*'s *cross_val_score* and *GridSearchCV*. This same resampling protocol will also be used in the non-linear classifiers.

Additionally, we will use the test partition of the dataset to test all our classifiers to correctly represent the generalization accuracy of each and avoid false results due to overfitting and generalization errors.

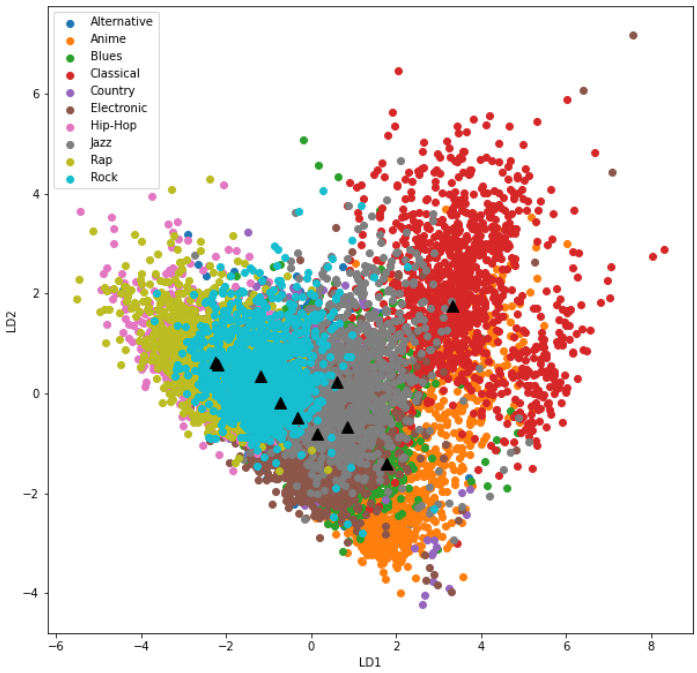
3.1. LDA and QDA

The first model approach we are going to use for our predictions are the Linear Discriminant Analysis and the Quadratic Discriminant Analysis models. Those are simple models with no hyperparameters to explore that will give us a basic model score and accuracy that we are going to use to compare to the other and more complex models.

After fitting the model with the train data, we can check the priors, means and weights of our first linear model. As we predicted in the previous section, the attributes popularity and loudness have some high weights since they are the most correlated variables to our target, while liveness and all the key variables have values near to 0.
Trying to predict the music genre with the test data we get some interesting results. First, doing cross validation we get an average score of around 0.52.

	precision	recall	f1-score	support
Alternative	0.36	0.37	0.37	1266
Anime	0.63	0.58	0.61	1276
Blues	0.49	0.49	0.49	1286
Classical	0.77	0.84	0.80	1198
Country	0.44	0.52	0.48	1227
Electronic	0.61	0.56	0.58	1274
Hip-Hop	0.43	0.51	0.47	1250
Jazz	0.50	0.38	0.43	1211
Rap	0.45	0.31	0.37	1268
Rock	0.50	0.64	0.56	1243
accuracy			0.52	12499
macro avg	0.52	0.52	0.52	12499
weighted avg	0.52	0.52	0.52	12499

Thanks to the classification report and the test LDA transformation, we can see that our model certainly groups together sets of songs, but from their scores and how close they are we can tell that it doesn't do a great job of it. This being said, it does a slightly better job of classifying and telling apart classical music (and anime to a certain extent) from the rest. This is probably due to them being the most different genres of the bunch.



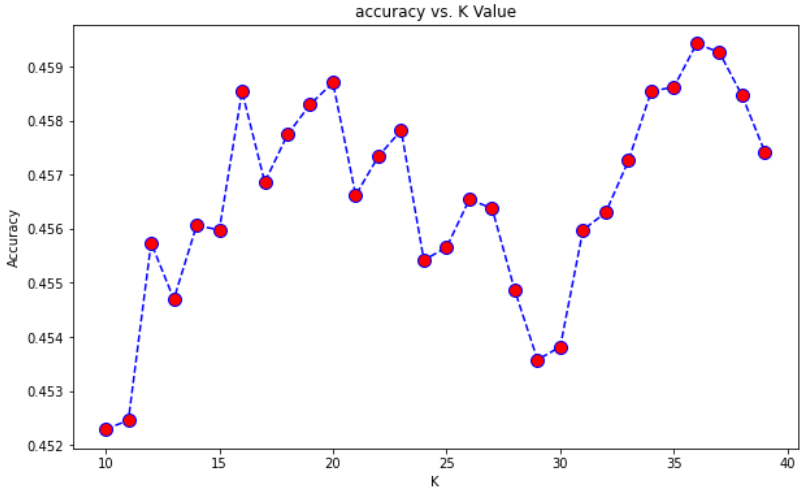
Alter fitting the Quadratic Discriminant model we see similar priors and means, and a similar weight distribution. Even though our LDA model didn't give us a great score or accuracy, it seems that our quadratic model gave us bad results as well. We can see that our score is even worse when using QDA (0.44 approximately), so using a fancier version of discriminant analysis will not work for our use case.

	precision	recall	f1-score	support
Alternative	0.38	0.18	0.24	1266
Anime	0.60	0.39	0.47	1276
Blues	0.40	0.41	0.41	1286
Classical	0.69	0.83	0.75	1198
Country	0.34	0.62	0.44	1227
Electronic	0.48	0.51	0.49	1274
Hip-Hop	0.35	0.38	0.36	1250
Jazz	0.43	0.26	0.32	1211
Rap	0.35	0.46	0.40	1268
Rock	0.46	0.38	0.42	1243
accuracy			0.44	12499
macro avg	0.45	0.44	0.43	12499
weighted avg	0.45	0.44	0.43	12499

3.2. KNN

We try to find a better model to classify the music genres using a K Nearest Neighbors model. For our hyperparameter exploration we will use *GridSearchCV*, which lets us do an exhaustive search over specified parameter values for an estimator, and we will try to find the best K (number of neighbors to take in account) and metric (distance metric to use) parameters.

Before the full exploration of the hyperparameters we will try to reduce the number of possible neighbors values plotting the accuracy vs the k-value with a simple loop.



The results show that we obtain the best accuracy with high k values, between 35 and 40, so we will try to find the best model with k = [35, 36, 37, 38, 39] and with euclidean, manhattan, mahalanobis and minkowski metrics.

After the hyperparameter exploration, the best KNN Classification model has k = 39 and metric = 'manhattan', but it doesn't give a better score than the initial models we checked before. In fact, it seems that using a Linear Discriminant model is still the best model to use to solve our classification problem, since KNN's score is 0.48 using the best parameters we found (while LDA was 0.52).

	precision	recall	f1-score	support
Alternative	0.34	0.36	0.35	1264
Anime	0.67	0.46	0.54	1274
Blues	0.57	0.34	0.43	1270
Classical	0.72	0.85	0.78	1238
Country	0.34	0.64	0.45	1254
Electronic	0.59	0.50	0.54	1274
Hip-Hop	0.39	0.44	0.41	1241
Jazz	0.49	0.40	0.44	1214
Rap	0.39	0.33	0.36	1243
Rock	0.45	0.43	0.44	1222
accuracy			0.48	12494
macro avg	0.50	0.48	0.47	12494
weighted avg	0.50	0.48	0.47	12494

Comparing the results with the LDA and QDA, we can see that this model has less precision trying to classify all music genres, even though the classification on classical music and anime are still the higher ones. Looking at the confusion matrix, we can confirm that our model classifies alternative songs as different genres, such as electronic or rock, as we

predicted in the introduction. Hip-hop and Rap have a similar problem, since the confusion matrix shows a horrid result when trying to classify songs from these two genres.

Predicted / Actual	Alternative	Anime	Blues	Classical	Country	Electronic	Hip-Hop	Jazz	Rap	Rock
Alternative	460	21	21	2	309	41	131	37	84	158
Anime	67	583	76	210	148	118	1	58	2	11
Blues	94	130	435	19	316	59	13	154	7	43
Classical	40	17	20	1047	33	21	0	59	0	1
Country	101	21	52	1	808	9	40	32	28	162
Electronic	149	86	44	10	126	633	66	102	32	26
Hip-Hop	63	1	4	0	91	15	547	13	409	98
Jazz	49	11	93	150	185	148	46	484	22	26
Rap	93	0	4	1	72	4	518	18	410	123
Rock	222	1	10	15	268	24	56	38	58	530

3.3. Linear SVM

The next linear model we are going to use is a Linear Support Vector Machine model, which only depends on the regularization parameter (C in the sklearn function). Using GridSearchCV again we get that the best regularization parameter is 100 between the one we tried, this means that the classifier chooses a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly, as opposed to a small C parameter, that would look for a larger-margin separating hyperplane, even if that hyperplane misclassified more points.

	precision	recall	f1-score	support
Alternative	0.42	0.36	0.39	1264
Anime	0.63	0.63	0.63	1274
Blues	0.55	0.45	0.49	1270
Classical	0.78	0.82	0.80	1238
Country	0.47	0.55	0.51	1254
Electronic	0.63	0.58	0.60	1274
Hip-Hop	0.45	0.52	0.49	1241
Jazz	0.53	0.47	0.50	1214
Rap	0.47	0.36	0.41	1243
Rock	0.51	0.70	0.59	1222
accuracy			0.54	12494
macro avg	0.54	0.55	0.54	12494
weighted avg	0.54	0.54	0.54	12494

This model is the first one that gives us a better score than the LDA model, even though it's not an amazing improvement (0.54 in comparison with LDA's 0.52).

Predicted Actual	Alternative	Anime	Blues	Classical	Country	Electronic	Hip-Hop	Jazz	Rap	Rock
Alternative	452	7	33	3	251	74	131	60	35	218
Anime	35	806	108	135	71	78	0	33	0	8
Blues	47	210	574	20	150	60	2	150	0	57
Classical	23	72	23	1020	15	28	0	54	0	3
Country	96	20	106	1	693	31	23	43	11	230
Electronic	82	106	76	10	57	742	33	120	27	21
Hip-Hop	84	0	0	0	22	17	651	13	370	84
Jazz	42	59	127	108	88	142	33	567	2	46
Rap	76	1	1	0	25	5	530	8	449	148
Rock	137	1	3	5	92	7	34	26	63	854

Looking at the confusion matrix and the classification report we can see that it classifies, as we saw in the previous models, the classical music genre quite well, with an accuracy of 78%, and the anime and electronic music genre slightly better than the other models (accuracies of 63%). On the other hand, the model still misclassifies the Hip-Hop and Rap genres, and it doesn't know what to classify when dealing with songs labeled as Alternative Music. These three last music genres have the worst accuracy of our model.

4. Testing non-linear Classifiers

4.1. MLP

Our first non-linear model will be a multilayer perceptron or neural network. This is a more complex model as it has far more hyperparameters such as the activation function, optimizer or, more importantly, the architecture itself. To find the best combination of these we have manually tested many until we found no more significant improvements in consequent modifications.

To find good architectures we built increasingly bigger (layer-wise) networks until we got to a point of diminishing returns and to set the size of each layer we started randomly selecting values and slowly found better patterns for our specific case. One pattern we found worked well was having all hidden layers use a number of neurons between the input and the output layer size, so have the biggest layer be the first/input one and the smallest one be the last/output (which has 10 neurons as we have to classify between 10 different music genres). We tested all these architectures with different activation functions and found that the *relu* function gave us the best results. The final architecture we picked has 7 layers with the following number of nodes each: 200, 75, 100, 25, 40, 50, 15.

Since this is a classification problem, the activation function on our last layer will be *softmax* as it will output the probability of each song belonging to a genre. This function could also be helpful in the event that we also needed to find subgenres or similar genres for a song.

Finally, to find the highest possible amount of epochs while avoiding overfitting we have used early stopping when test/validation dataset performance goes down with extra epochs

and does not recover within 10 of these. This value depends on each architecture, but it generally fluctuates between 15-25 on the best ones we have found. The architecture we ended up picking gave us the best results with around 26 epochs.

	precision	recall	f1-score	support
Alternative	0.47	0.42	0.44	1264
Anime	0.77	0.68	0.73	1274
Blues	0.62	0.50	0.55	1270
Classical	0.82	0.84	0.83	1238
Country	0.54	0.55	0.54	1254
Electronic	0.71	0.57	0.63	1274
Hip-Hop	0.45	0.35	0.39	1241
Jazz	0.56	0.51	0.53	1214
Rap	0.47	0.54	0.50	1243
Rock	0.47	0.81	0.59	1222
accuracy			0.58	12494
macro avg	0.59	0.58	0.57	12494
weighted avg	0.59	0.58	0.57	12494

Even with the complexities and learning power of this model, the best accuracy score we could get was 0.600. But looking at the confusion matrix of the result we can see that similarly to other models our biggest problem lies in the similarity between some genres, especially between rap and hip-hop, but also rock, alternative, country and some others. This being said, if we ignore these easily confused genres in the confusion matrix we are getting good enough results.

Predicted Actual	Alternative	Anime	Blues	Classical	Country	Electronic	Hip-Hop	Jazz	Rap	Rock
Alternative	532	9	22	1	165	24	100	42	59	310
Anime	50	872	94	97	63	56	0	27	0	15
Blues	58	120	640	28	151	49	1	146	2	75
Classical	32	32	22	1046	8	19	0	73	0	6
Country	102	13	60	1	687	11	21	42	12	305
Electronic	106	63	92	8	45	723	19	137	27	54
Hip-Hop	63	0	1	0	12	8	429	6	607	115
Jazz	76	16	102	99	81	125	31	614	3	67
Rap	42	0	1	0	6	3	327	6	671	187
Rock	83	2	3	2	52	2	17	12	61	988

4.2. Random Forest

The last model we will try is a Random Forest model. This model fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

It also has a lot of hyperparameters to explore. In particular, we will try to get the best random forest model depending on the number of trees in the forest, the maximum depth of the trees, the minimum number of samples required to split an internal node and the

minimum number of samples required to be at a leaf node. After trying with both types of criterion, we decided that 'gini' is the best function to measure the quality of a split. Finally, we chose to test the following hyperparameters: for max_depth, we will try 5, 10, 15, 30, 50 and none (default). For min_samples_leaf and min_samples_split 2, 4 and 6. And, for n_estimators 100, 150, 200 and 500.

This time, to avoid a higher execution time, we will use RandomizedSearchCV for the exploration of the hyperparameters explained before. The result shows that the best parameters are maximum depth = 15, minimum sample split = 2, minimum sample leaf = 2 and number of trees = 100

	precision	recall	f1-score	support
Alternative	0.48	0.33	0.39	1266
Anime	0.81	0.74	0.77	1276
Blues	0.64	0.55	0.59	1286
Classical	0.83	0.87	0.85	1198
Country	0.58	0.60	0.59	1227
Electronic	0.67	0.64	0.65	1274
Hip-Hop	0.36	0.41	0.39	1250
Jazz	0.53	0.53	0.53	1211
Rap	0.35	0.31	0.33	1268
Rock	0.49	0.73	0.58	1243
accuracy			0.57	12499
macro avg	0.57	0.57	0.57	12499
weighted avg	0.57	0.57	0.57	12499

Using a random forest classifier doesn't seem to have an improvement in the score of our predictions, as it is a bit lower than the MLP model. Looking at the classification report we see that some categories have a better precision, though, like Classical or Anime, that are the best classified categories in all our models, but also in Alternative, which is a conflictive category as we have seen before. In opposition, the classification of Rap and Hip-Hop is even worse, since it classified the majority of Rap songs as Hip-Hop ones and vice versa.

Predicted Actual	Alternative	Anime	Blues	Classical	Country	Electronic	Hip-Hop	Jazz	Rap	Rock
Alternative	450	12	25	2	176	80	127	56	63	273
Anime	35	892	85	115	56	54	0	24	0	13
Blues	46	127	666	17	123	65	4	150	3	69
Classical	26	28	25	1069	8	17	0	59	0	6
Country	86	20	57	0	726	9	20	41	27	268
Electronic	69	63	92	4	34	793	23	148	18	30
Hip-Hop	36	1	1	0	16	9	482	13	619	64
Jazz	33	16	130	97	72	138	24	657	8	39
Rap	45	0	3	0	8	2	647	7	401	130
Rock	109	3	5	7	81	12	46	21	64	874

5. Final Model

Having seen the results of all our models, it is clear that our MLP model has been the most accurate (0.600 max accuracy), and therefore is our first choice as the final model to classify our songs. Since we have used the test partition along with cross validation for all our analysis, we know that generalization errors are not causing us to get 'false results'. This being said, if we use our MLP classifier with the train partition we don't gain much accuracy, so our generalization error isn't very big in this case anyway.

	precision	recall	f1-score	support
Alternative	0.50	0.37	0.43	3736
Anime	0.81	0.73	0.77	3726
Blues	0.63	0.54	0.58	3729
Classical	0.82	0.89	0.85	3757
Country	0.60	0.54	0.57	3746
Electronic	0.64	0.67	0.65	3722
Hip-Hop	0.48	0.36	0.41	3744
Jazz	0.58	0.54	0.56	3786
Rap	0.47	0.59	0.52	3757
Rock	0.51	0.77	0.61	3778
accuracy			0.60	37481
macro avg	0.60	0.60	0.60	37481
weighted avg	0.60	0.60	0.60	37481

6. Conclusions

6.1. About the dataset

We have quickly found that song classification with this dataset is harder than we anticipated due to many different factors:

- Firstly, even though we have many features, music is very hard to define numerically and there will probably always be missed information that can't be accurately portrayed in a small enough dataset (lyrical intonation, synth sound description, singer accent, local popularity…)
- Genres have very broad definitions/borders and most songs don't fit a single one, so subgenres and proximity to other genres should be taken into account when trying to fit a song's genre.
- Some metrics used to define music are more subjective than others. For example, we can easily define a song's length in seconds, or tempo in beats per minute, but trying to describe a song's danceability or liveness with a number is harder.

6.2. About our limitations

We have obviously been limited by a lot of factors, and many could be eliminated/fixed in future studies, which is part of what projects of this kind are useful for. The main problems we have found are:

- Limited computational resources to be able to test more complex models and explore bigger hyperparameter combination spaces. *Google's colaboratory* has greatly helped us in this regard, specially when training our MLP model using *tensorflow*'s GPU power.
- Time, again to be able to explore more possibilities and better exploit the methods used. This is somewhat tied to the previous point, as long execution times have limited our experimentation.
- The dataset itself, as it has a limited number of attributes to describe songs and only uses one genre for each, as we discussed in 5.3. We believe that this is one of the biggest factors that could be fixed in a future study on this subject, and that using a dataset with multiple genres per song would greatly increase accuracy.

6.3. About models/classifiers

Our first takeaway from analyzing so many different models on a single dataset is that more complex models don't always give better results and when they do it's not always by much, like we expected. The first sign we got of this was after testing all our linear/quadratic classifiers and saw that the first and one of the simpler models (LDA) gave us one of the best results of the bunch while also being the fastest to train.

We also quickly noticed the importance (and computational/time complexity) of finding the best set of hyperparameters in most models, specially the ones where making an educated guess (and sometimes not so educated) was the most we could do without testing. Models like MLP which have been proven to be more trial-and-error based (when testing architectures), as they greatly depend on the dataset at play, are very hard to perfectly optimize.

The final observation we have made is that no matter how complex and good at finding patterns a model is, sometimes it just isn't enough to solve the problem at hand. As we have found in the rest of our conclusions, sometimes spending more time on curating the perfect dataset is better than trying to optimize a fancy classifier and fit it with less useful data, and sometimes a problem may just be too subjective to be easily and accurately classified by a machine.

7. Documentation / References

Python external library documentations referenced:

- Scikit-learn: https://scikit-learn.org/stable/modules/classes.html
- Matplotlib: https://matplotlib.org/stable/api/index
- Tensorflow: https://www.tensorflow.org/api_docs/python/tf

Articles used:

- Early stopping to avoid overfitting in a tensorflow MLP: https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/
- On resampling protocols: https://machinelearningmastery.com/statistical-sampling-and-resampling/
- On MLP optimizer functions: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6
- On multi-label classification: https://machinelearningmastery.com/multi-label-classification-with-deep-learning/
- On finding the best architecture for an MLP: https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/
- On finding the best parameters for an SVM: https://www.vebuso.com/2020/03/svm-hyperparameter-tuning-using-gridsearchcv/
- On plotting KNN accuracy: https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb
- On using RandomSearchCV and GridSearchCV: https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/

Forum discussions considered:

- On activation functions for the output layer: https://stats.stackexchange.com/questions/218542/which-activation-function-for-output-layer
- On designing an MLP architecture: https://stats.stackexchange.com/questions/232393/is-there-any-method-for-choosing-the-number-of-layers-and-neurons
- On MLP layer sizes: https://www.quora.com/In-an-MLP-if-the-number-of-nodes-in-the-input-layer-is-fewer-than-the-hidden-layer-what-would-happen
- On using SVM or RLS: https://cvstuff.wordpress.com/2014/11/29/latex-l_1-versus-latex-l_2-loss-a-svm-example/
- On influence of C parameter in SVM: https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel