Music Genre Classification using Machine Learning with Python

Valentine Brunet
UPC - ETSETB

Ana Manzano
UPC - ETSETB

***Abstract*—This paper aims to obtain a prediction of the musical genre of a set of song samples as accurately as possible. 14 characteristics define each sample (duration_ms, key, mode, loudness, acousticness, danceability, energy, instrumentalness, liveness, speechiness, tempo, valence, release_date, popularity and genre) and up to 17 different classifiers are used to find the one that provides the best performance. The selected model Random Forest has been able to achieve a prediction in the test set of 24% and in the validation of 32%.**

CONTENTS

I. INTRODUCTION

Music genre is a difficult thing to define, even among people, opinions may vary and they may disagree on the genre of a piece[4]. Spotify[3], a digital music, podcast, and video service, has a database that includes information on each song characteristic. The aim of this paper is to develop a predictive system for 15 music genres for music tracks based on the analysis of each one of them. We will use some of the samples from the Spotify database and we will explore the best way to combine features and classifiers to obtain the highest possible accuracy.

II. QUICK VISUALIZATION OF THE DATA

A. Data Sources

The dataset is taken from Kaggle [2], consists of two CSV files. The first one is used for training, it contains 8363 different samples of music pieces with the following characteristics for each of them: duration_ms, key, mode, loudness, acousticness, danceability, energy, instrumentalness, liveness, speechiness, tempo, valence, release_date, popularity and genre. Each of the pieces is identified with an Id. The second CSV file is used to test our model, it has 3586 different songs with the same characteristics as the previous file but without genre, which is the one we predict.

A validation set with 1362 is created, so this samples will be extracted from train set. Therefore, the samples are distributed as follows:

	Samples
Train set	7001
Validation set	1362
Test set	3586

TABLE I
NUMBER OF SAMPLES FOR EACH SET

B. First visualization of the dataset

Let's see the first 5 rows of the train_dataset:

Id	duration_ms	key	mode	loudness	acousticness	danceability	energy	instrumentalness	liveness	speechiness	tempo	valence	release_date	popularity	genre
9238	251594	7	1	-9.652	0.442	0.635	0.27300	0.8770	0.1590	0.0368	84.039	0.2130	2021-05-14	39	5
7394	294960	1	1	-33.366	0.993	0.335	0.00501	0.9190	0.0618	0.0451	132.085	0.0383	2018-07-27	65	6
5575	206583	1	0	-5.428	0.761	0.672	0.71400	0.0000	0.1010	0.2530	172.061	0.4090	2017-09-05	43	10
7002	258280	5	0	-5.749	0.409	0.820	0.63500	0.0000	0.2440	0.0772	101.024	0.5280	2020-02-29	68	12
6699	223173	5	1	-7.462	0.202	0.830	0.71100	0.0448	0.1370	0.0448	114.646	0.9010	1978-07-17	55	3

Fig. 1. Head(5) of train_dataset

So we can see that we have a lot of difference of range between all values, so we need to scale or normalize the data.

C. Checking types of columns

Let's check now the types of all columns, because it's really important that all value is a "float" or "integer" (number) to be used in ML models.

```
 duration_ms          int64
key                   int64
mode                  int64
loudness            float64
acousticness        float64
danceability        float64
energy              float64
instrumentalness    float64
liveness            float64
speechiness         float64
tempo               float64
valence             float64
release_date         object
popularity            int64
genre                 int64
```

Fig. 2. Types columns of the dataset

As you can see, there is only one column : ”release_date” that is ”object” type (string) and so we have to change it. (Done in Section 4.1)

D. Checking some null values on the dataset

```
Nul values on train_dataset :
 duration_ms         0
key                  0
mode                 0
loudness             0
acousticness         0
danceability         0
energy               0
instrumentalness     0
liveness             0
speechiness          0
tempo                0
valence              0
release_date         0
popularity           0
genre                0
```

Fig. 3. Null values on the train_dataset

So we can see that this dataset is very clean and complete.

E. How many features per genre?

Let’s sum the genre for all rows to check repartition of the output of our dataset:

```
[   0,  407]
[   1,  449]
[   2,  645]
[   3,  617]
[   4,  320]
[   5,  353]
[   6,  337]
[   7,  442]
[   8,  930]
[   9,  526]
[  10,  320]
[  11,  388]
[  12,  899]
[  13, 1034]
[  14,  696]
```

Fig. 4. Number of features(right) to the according genre(left)

Being blues: 0, classical: 1, country: 2, disco: 3, electronic: 4, hip-hop: 5, indie: 6, jazz: 7, pop: 8, punk: 9, r-n-b: 10, reggae: 11, reggaeton: 12, rock: 13 and soul: 14.

F. Scatter Matrix

Let’s analyse the scatter matrix:

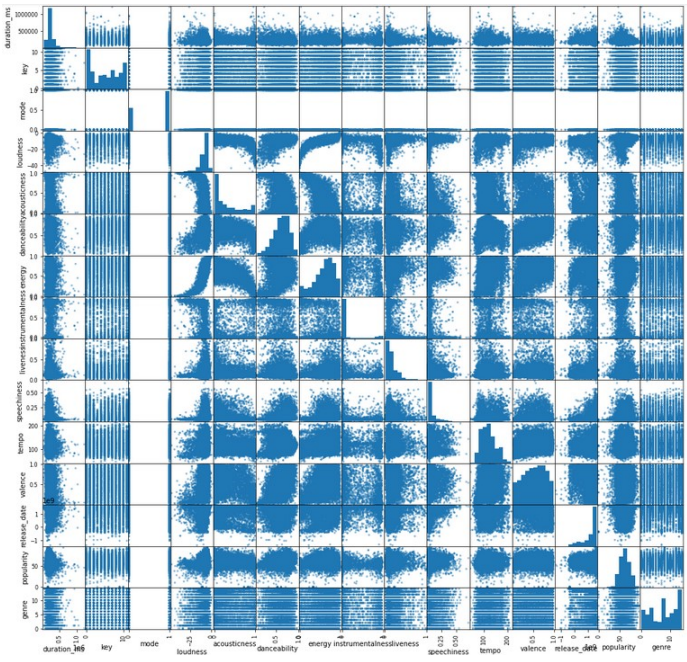

Fig. 5. Number of features (left) to the according genre (right)

On the one hand, variables present big variation on their own. Some features seems to follow Gaussian law, others are binary, etc... On the other hand, we can see that the output genre seems to very dispersed on with respect to all others features, so this problem promises to be complicated.

G. Correlation and dependance of the features

This is a heatmap showing correlation (Pearson) between all features and output:

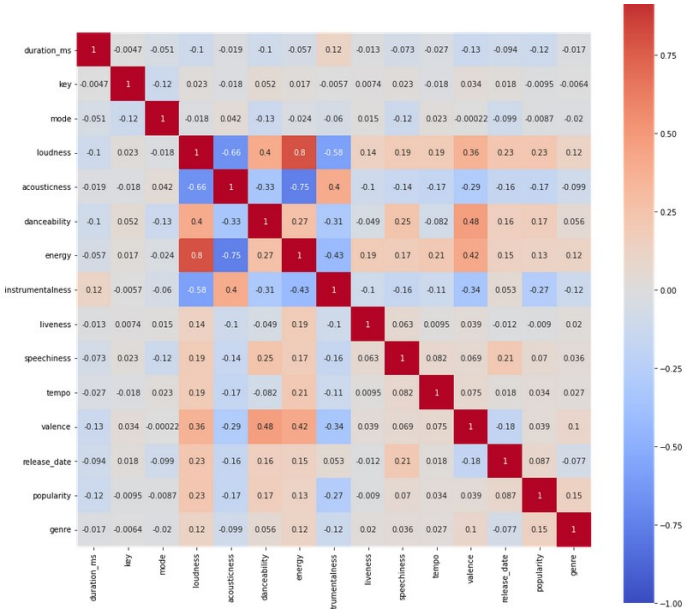

Fig. 6. Heatmap of correlation

So the first obvious observation we can make is the dependance of 5 features (in the middle of the figure) : ”loudness”, ”acousticness”, ”danceability”, ”energy” and ”instrumentalness”. This is not surprising at all that loudness is correlated with energy. That’s why a PCA/MDA is mandatory to choose the accurate features and reduce inter-dependance.

On the other hand, we can see that there is only few features that seems to influences the output ”genre”: ”loudness”, ”energy”, ”valence”, ”instrumentalness” and ”popularity”.

We have now make a small visual analysis of the data, let’s keep forward.

III. CLEAN THE DATA AND PREPROCESSING IT

A. Change the format of the column ”release_date”

As has been said before, the column ”release_date” is on a non-interpretable type for ML Model, we just have to convert into a number. But how can we convert a date to a number? Using Timestamp! This is the number of second since 1st January 1970. So with different conversion in python : object ->datetime, datetime ->timestamp, we succeed to change the type of this column.

```
release_date          float64
```

Fig. 7. Type of column ”release_date”

B. Remove outliers

Outliers are the values in dataset which standouts from the rest of the data, in our case it could be strange music that can be hard to classified. Is defined as a value that is more than 3 standard deviations from the mean. We used the Z score measurement.

Z score is an important measurement or score that tells how many Standard deviation above or below a number is from the mean of the dataset. It is calculates by subtracting each value with the mean of data and dividing it by standard deviation.

So after doing it in the original train_dataset, we loose around 1000 inputs values, this is absolutely enormous! So we decided to use 4 standard deviation to cut only 300 inputs values.

C. Let’s preprocess the data

Many machine learning algorithms perform better when numerical input variables are scaled to a standard range. We choose standardization.

Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1.

IV. REDUCTION OF DIMENSION

The main purposes of a principal component analysis are the analysis of data to identify patterns and finding patterns to reduce the dimensions of the dataset with minimal loss of information.

We used PCA and MDA for this purpose. Depends on each model PCA and MDA had closer results on prediction so we used them according to the model.

Finally we saw that we have the best score using a dimension reduction of 6 to 8 features.

V. OUR METHODOLOGY

Let’s present our methodology when running the models. To do so, we create 4 main functions we used everywhere :

1) Fitting the model: We create one function that is going to make a research of best hyperparameters using cross-validation using Kfold. And after fit the model.
2) Result function: This function is going to compute accuracy and error score on train and validation dataset. Also display correspondive confusion matrix.
3) Prediction function: This one make the prediction on test dataset, and will call the submission function to create the CSV we will upload in Kaggle.

VI. MODELS USED AND IMPROVEMENTS

We use many different classifier model because we know that there is no better model for every problem, just one that is appropriate of our problem:

1) LDA
2) QDA
3) Non-linear SVM
4) Parzen Windows
5) KNN
6) Decision Tree
7) Random Forest
8) MLP
9) ExtraTrees Classifier

10) Hard Voting
11) Soft Voting
12) AdaBoost
13) XGBoost
14) Catboost

A. *Imbalanced Classes*

As we said in II.E, the 15 genres that will be predicted are: blues, classical, country, disco, electronic, hip-hop, indie, jazz, pop, punk, r-n-b, reggae, reggaeton, rock and soul tracks. We can see how the samples of train are distributed in the different genres in the following graphic:

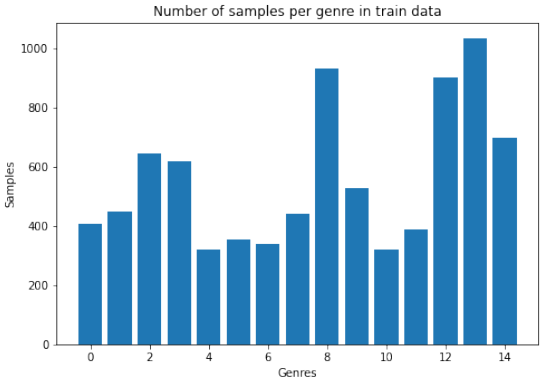

Fig. 8. Number of samples per genre in train data

As we can see in Fig. 8, there is quite a difference between the number of samples in the different classes, which could lead to a poor prediction in the test set. To solve this problem, we have chosen to test both oversampling and undersampling[1], with the last one mentioned giving better results. In oversampling, we duplicate examples from the minority class and on undersampling, we delete examples from the majority class. What we end up with is a training where for each class we have the same number of samples.

B. *Results*

After a long process of data processing, we were able to evaluate which of the different classifiers gave the best results. From the very beginning and with all of them, the presidency in the training set has been very high (93%-98%) but not so in the validation set (25-32%). By balancing the data and reducing the dimension we managed to obtain 32% accuracy in the validation.

At the end our best model is Random Forest, with 24% test score:

```
Best hyperparameters {'max_leaf_nodes': 300, 'min_samples_split': 3}
Estimator train score: 0.568091
Train_error : 0.4319085487077535
Estimator validation score: 0.327038
Validation_error : 0.672962226640159
```

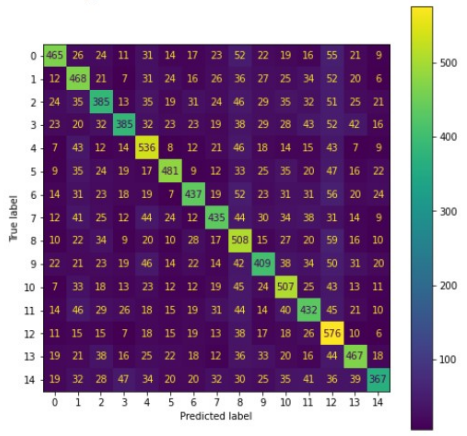

Fig. 9. Scores and train confusion matrix

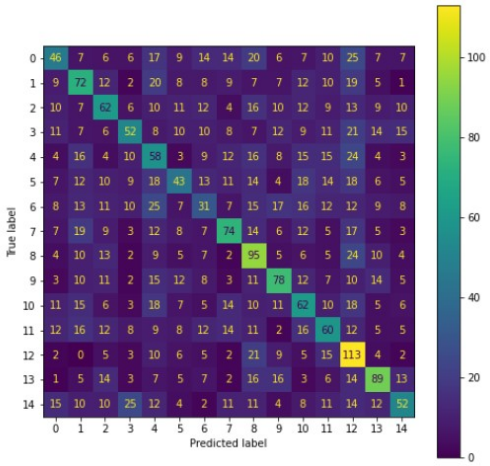

Fig. 10. Test confusion matrix

It's the best model because the random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm. And DT algorithm forces the consideration of all possible outcomes of a decision and traces each path to a conclusion, so that is pretty accurate with music genring.

On those confusion matrix, we can't really observe a pattern of misclassification, so it would not help us find where the problem comes from.

VII. CONCLUSION

We could not understand why we have this bad score in validation and test predictions. Having a high train score and bad test score is synonym of overfitting, but none of the techniques to solve this problem have been able to significantly reduce validation and test error. We think that the cause may be a discernible difference between the data in the sets.

Even so, we believe we have done a good job, in which we have learned a lot and we have been able to work with a real problem. As a future work, we propose to improve significantly the performance in the sets that we have not been able to obtain a score higher than 93%, focusing on the type of information we have.

REFERENCES

[1] *10 Techniques to deal with Imbalanced Classes in Machine Learning*. URL: https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/.

[2] *Kaggle*. URL: https://www.kaggle.com/ (visited on Jan. 16, 2022).

[3] *Spotify*. URL: https://www.spotify.com/ (visited on Jan. 10, 2022).

[4] Tyler Dammann and Kevin Haugh. *Genre Classification of Spotify Songs using Lyrics, Audio Previews, and Album Artwork*. Dec. 29, 2021. URL: http://cs229.stanford.edu/proj2017/final-reports/5242682.pdf.