1 Data understanding and preprocessing

1.1 Introduction

This is a study on the **Free Music Archive** (FMA), an open and easily accessible dataset suitable for evaluating several **MIR** tasks, a field concerned with browsing, searching, organizing and extracting information from large music collections. For our task we decide to analyze which kind of Album our tracks are, using the dataset `tracks.csv`. So with this task in mind we start by understanding how our big dataset is structured. We have 52 features of which 5 are categorical, 7 are date-time, 2 are float, 15 are integer and 23 are objects; and there's a total of 106574 records.

1.2 Coherence

We found two types of missing values. First, *nan*: this appears in our instances randomly, so we assumed these to be normal missing values due to mistakes or omissions in data entry and we decided to treat them in a following part.
The second missing value we found is *-1*. We did a first coherence pass with integrity checks for the non negativity of variables like *favorites*, *listens* and *comments*, which would not make sense to be negative. We found that there are many instances of these being -1, which we figured to be **missing values** based on the fact that there's a **very high correlation** of -1 appearing in one field and -1 appearing in multiple fields.
This second kind of missing value, *-1*, was replaced with *no info*. We want to treat as data point this kind of absence of values, since we think there's a high chance of these missing values indicating things like those "-1" fields not applying in those specific songs cases (e.g.: a field of artist comments in a song uploaded in a website where it was not possible to add comments to the artist). Also, we chose this approach because this second type of recurring missing value appears in many thousands of our instances.

1.3 Discretization

For some classification methods we will use decision boundaries must be drawn, and where that happens, it makes sense to limit the complexity of those boundaries by discretizing continuous variables, like *listens*. We're not really interested in if a song had 81 or 84 listens; we might be interested instead if listens were, compared to their distribution in this dataset, *very low*, *low*, *medium* or *high*.
Therefore, in an effort of getting the best possible **generalized performance** and minimizing possible sources of overfitting, we chose to discretize several variables. We did two types of discretization:

- discretization of features that we didn't want to keep track of their value but just considering **if there was a value** or **not** (extracting dummy variables);
- discretization of features according their **distribution**.

Using the first method we discretized: *(album, comments)*, *(artist, comments)* and *(track, comments)* in "*no info*" if there was -1 (like previously explained), "*no comments*" if there was 0 and the rest of instances with value "*commented*".
According to their distribution we discretized: *(album, favourites)*, *(artist, favorites)*, *(track, favorites)*, *(album, listens)*, *(track, listens)* in: "*no info*" if there were a -1 value and the rest of instances in "*low*", "*medium*", "*high*", "*higher*" and "*super*".
We also initially discretized *(track, duration)* in "*1min*" if the track is shorter than 60 seconds, "*2min*" if the track is shorter than 120 seconds and "*3min+*" if the song is longer than 120 seconds, but later on we decided to exclude this in the classification task because most classifiers didn't see their performance improved by using it.
For the date variable we decides to only take the year, again, to avoid being overly specific in our data and avoid overfitting.
Of course, this discretization was only applied when useful and it will be specified in each section (e.g.in knn methods, it wasn't used).

1.3.1 Very low coverage attributes

There are a few attributes with very low coverage ($<$10 %) that are very interesting. These are:

- *(artist, active year end)*
- *(artist, wikipedia page)*
- *(track, composer)*
- *(track, date recorder)*
- *(track, information)*
- *(track, lyricist)*
- *(track, publisher)*

Of these, let's take *engineer*. It's not strictly relevant for us the specific name of the sound engineer. We're not going to extract information out of the names, like experience of the professional or their hourly rate. However, the presence of an engineer in the song metadata (which we hope is as complete as song metadata can be) might indicate a recording **made with higher quality** or with a **higher budget** than one without a sound engineer. Therefore we figured that all we cared about, instead of people names, were the **presence or not** of an engineer. Therefore we made dummy variables out of attributes such as this one.
Dummies have value 1 when the attribute was present, 0 when it was missing.

1.4 Other missing values

We made an assumption where the language of title of a certain tracks is the same of the lyrics, so we replaced all missing values of variable *(track, language code)* using a language detector applied on the *(track, title)*, that has full coverage of values instead.

For the rest of variables we decided to simply remove all the rows with missing values since they were relatively very few. Features involved in this process are:

- *(artist, date created)* with 103045 non-null values
- *(track, license)* with 106487 non-null values
- *(album, date created)* with 103045 non-null

We did the same thing for the *(album, type)* variable, even if it has a greater number of missing values (*100066 non-null values*), but we wanted to keep our target variable as accurate as possible.

1.5 Special variables study

1.5.1 Album Type

We decide to set as *target class* **Album Type**, to predict which kind of album type the tracks belong to. *(album, type)* is a categorical feature of 5 values:

- **Album**
- **Live Performance**
- **Radio Program**
- **Single Tracks**
- **Contest**

Looking at the class distribution we saw that *contest* has only 14 records in the entire dataset. So we decided to delete them. Below we can see the distribution of the variable and how we encode that for the classification task.

Album-type value	Number of istance	Encode as
Album	86944	0
Live Performance	5005	1
Radio Program	6524	2
Single Tracks	917	3
Contest	14	Deleted

Figure 1: Album Type Enconding

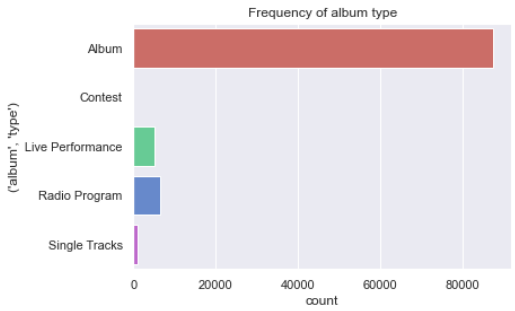

Figure 2: Album Type Distribution

1.5.2 Genres

We noticed that the feature **genre top** had *49598 missing values*, for this reason we decided to fill them with a set of dummy variables. For each genre we added a **dummy** variable that is a binary variable (coded as 1 or 0) to reflect the presence or absence of a particular genre of first level inside the total set of genre per track that we found in the variable *genre all*. Our first level genre variables created are:

- Rock
- Electronic
- Experimental
- Hip-Hop
- Folk
- Instrumental
- Pop
- International
- Classical
- Old-Time / Historic
- Jazz
- Country
- Soul-RnB
- Spoken
- Blues
- Easy Listening

Since some tracks had no first level genre we added a new dummy called **top genre missing** to the set. After the dummy creation we deleted all three features about genres: *genre*, *genre top* and *genre all*.

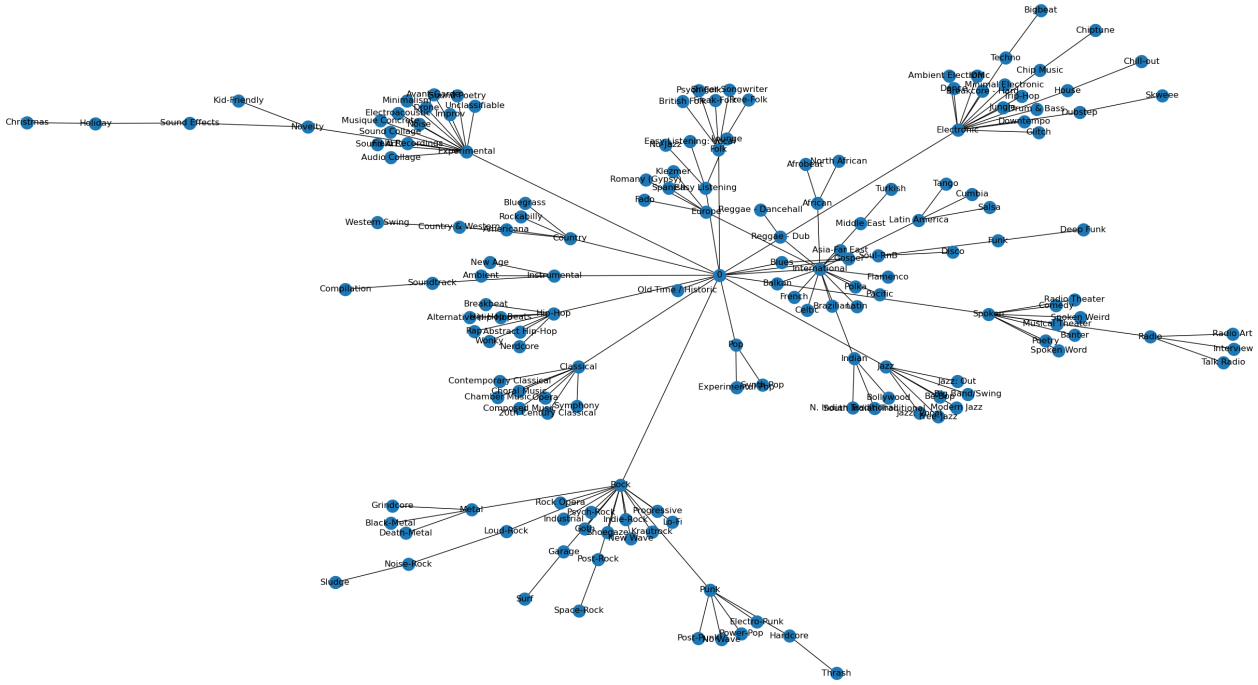

Figure 3: Genres tree

1.5.3 Correlation Matrix

To guide our work, we analyzed and plotted the correlations between attributes with a **Pearson's correlation matrix**. We have as final preprocessed dataset 44 features and 99319 records.

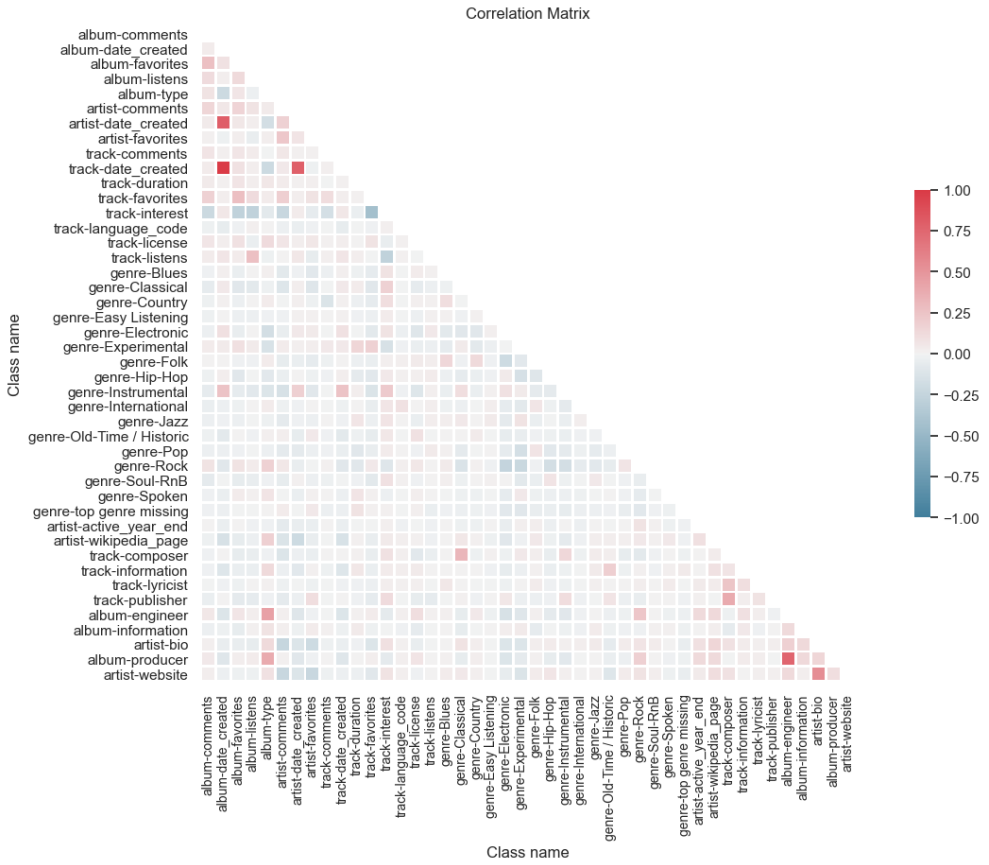

Figure 4: Pearson's Correlation Matrix

2 Starting classification

2.1 Feature selection

We decided to exclude some features in order to avoid any possible undue effects on the classification. For instance, the track number might highly correlate to the track being a single (if ==1) or part of an album (if >1). Moreover, it doesn't make much sense to include the title of the songs or albums in our classification models. We excluded these variables:

- (album, title)
- (album, id)
- (track, title)
- (artist, id)
- (artist, name)
- (track, number).

2.2 Decision Tree Classifier

We started by building a decision tree. We built two different decision trees with same parameters, a first one using discretized features described in paragraph **1.3** and the second one using the continuous features. For the building phase we used 43 features.
After the feature selection our dataset is split using the variable *(set, split)* in our raw data set. A splitting of training and test of 80-20 percent is generated. For the parameters tuning we used a random search with a wide set of different values.

Once found the best set of parameters we run the model. We obtained the following results:

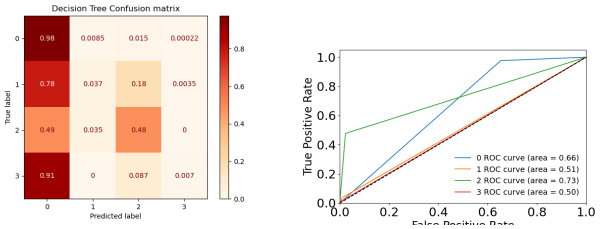

Figure 5: DecisionTree Discrete Features, Confusion Matrix and ROC

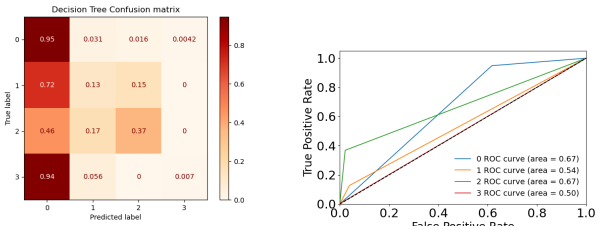

Figure 6: DecisionTree Continuous Features, Confusion Matrix and ROC

As we can see from the results, the decision tree with discretized features is slightly better. As we expected our naive trees tend to classify records as the majority class, artificially increasing the accuracy, which is around **88**%.