Python for Data Analysis

Project S7

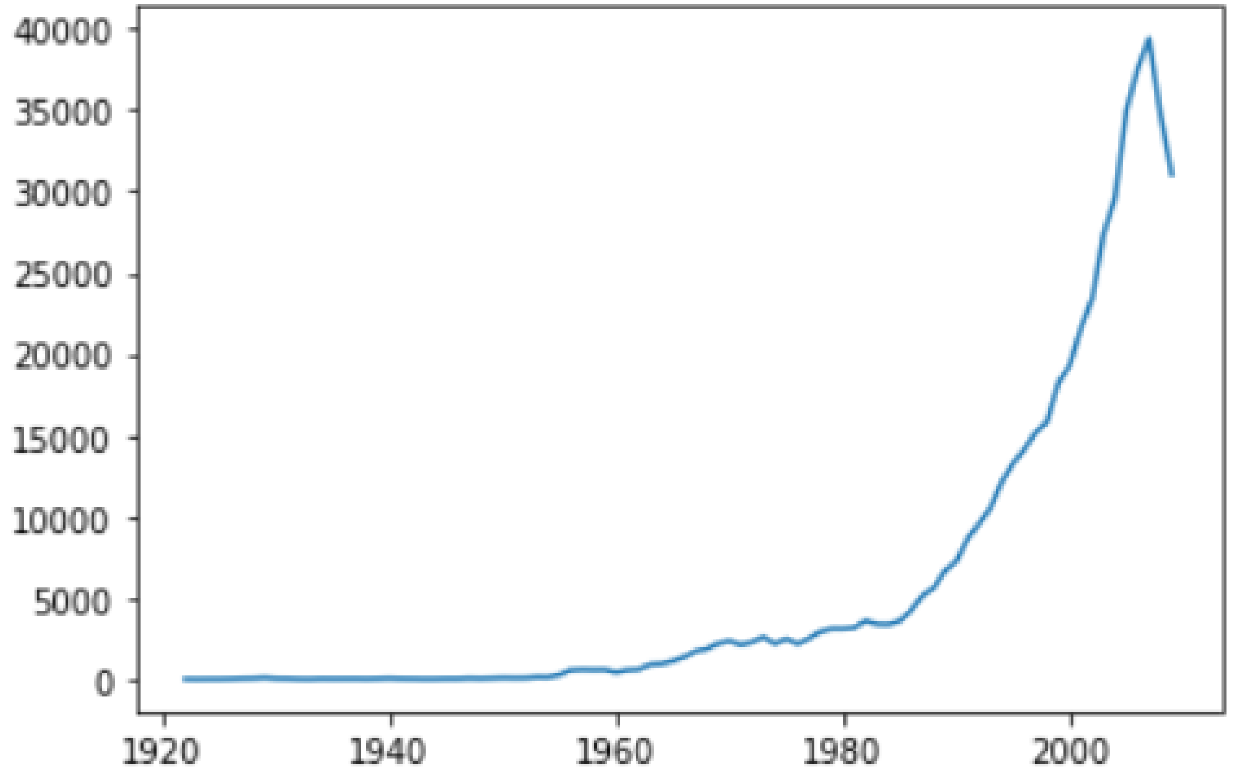
Paul JOUËT - Aladin HOMSY

Introduction

- The goal of the project is to study a dataset and apply learning methods on it, using Python and packages such as sklearn, pandas and numpy. In our case, we studied the YearPredictionMSD Data Set. The elements of the dataset are 515345 songs, with 90 attributes representing timbre variances and covariances, and the target is the release year of the song. We will call all the variances and covariances of the timbres 'timbre or feature [1 to 90]' from now on.
- The dataset contains mostly western and pop music, and the audio features are described as real numbers. We will try modelling a regression for estimating the release year of a song. We might also try another approach using ranges of years (70s, 80s... for example) if the results are not satisfying.

Data visualization

- Is the data evenly spread ?



Is the data evenly spread ?

We see that the data is **unevenly spreaded**, if we look at the number of occurences of a year in the dataset, relative to the year, the obtained graph seems to represent an **exponential** growth.

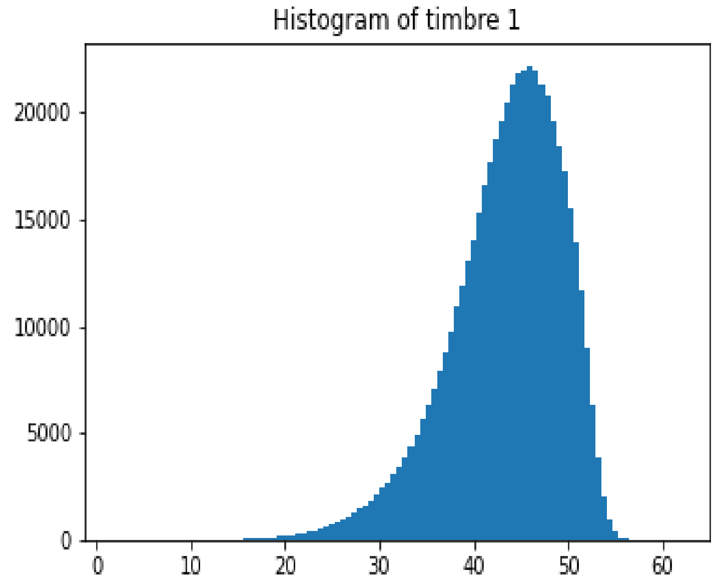
We thought about pre-processing methods in order to solve this problem, for example, we could duplicate values in the 1920 to 1980 range or ignore most values in the 1980-2010 range to obtain a more evenly distributed dataset.

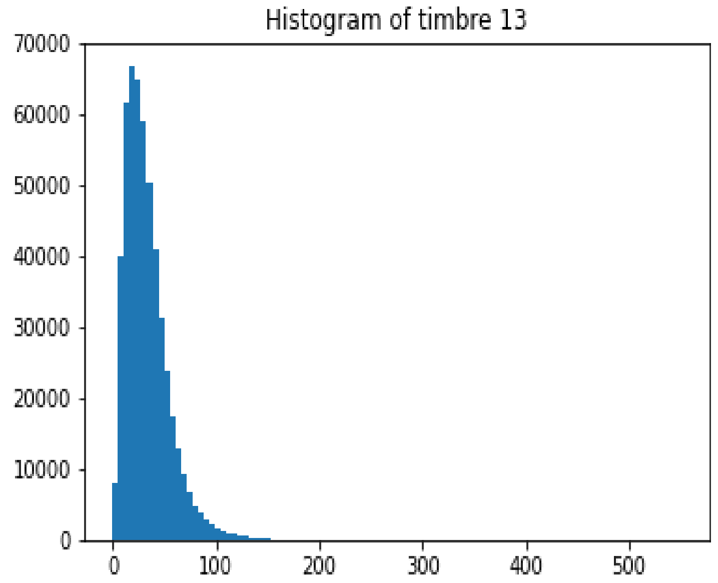
The problem with duplicating will be the *computational time* (5 million entries may be way too long to process for our little computers) and ignoring data could lead to *poor results* if there is not enough correlation between the features and the target...
According to *Pre-processing Protocol for Nonlinear Regression of Uneven Spaced-Data* from P. Panja et al. 2020, it could be possible to use only 5-10% of the data and still get good results if the right method is used, however we are not sure that the methods presented are applicable in our case.

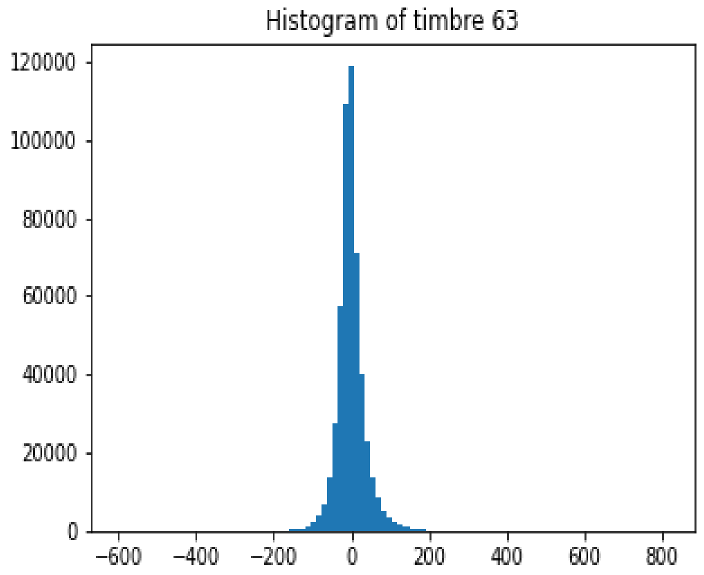
We could do both to mitigate the drowbacks from each method, by **duplicating** some values and **ignoring** others.

Understanding the other features

- Distribution







After re-reading the description of the dataset, we thought something was a little bit unclear, what do the variables REALLY represent ? We knew they represented timbres **variances** and **covariances**, but the number of timbres was unclear. Since the dataset is not annoted we just told ourselves that this was just not really interpretable by humans, after all we dont know about a timbre, how it was computed, nor in which order they were put into the dataset (because there is not much information). But hey, if we have *12 variances*, we should have *66 covariances* : because the binomial coefficient C(2,12) = 66.

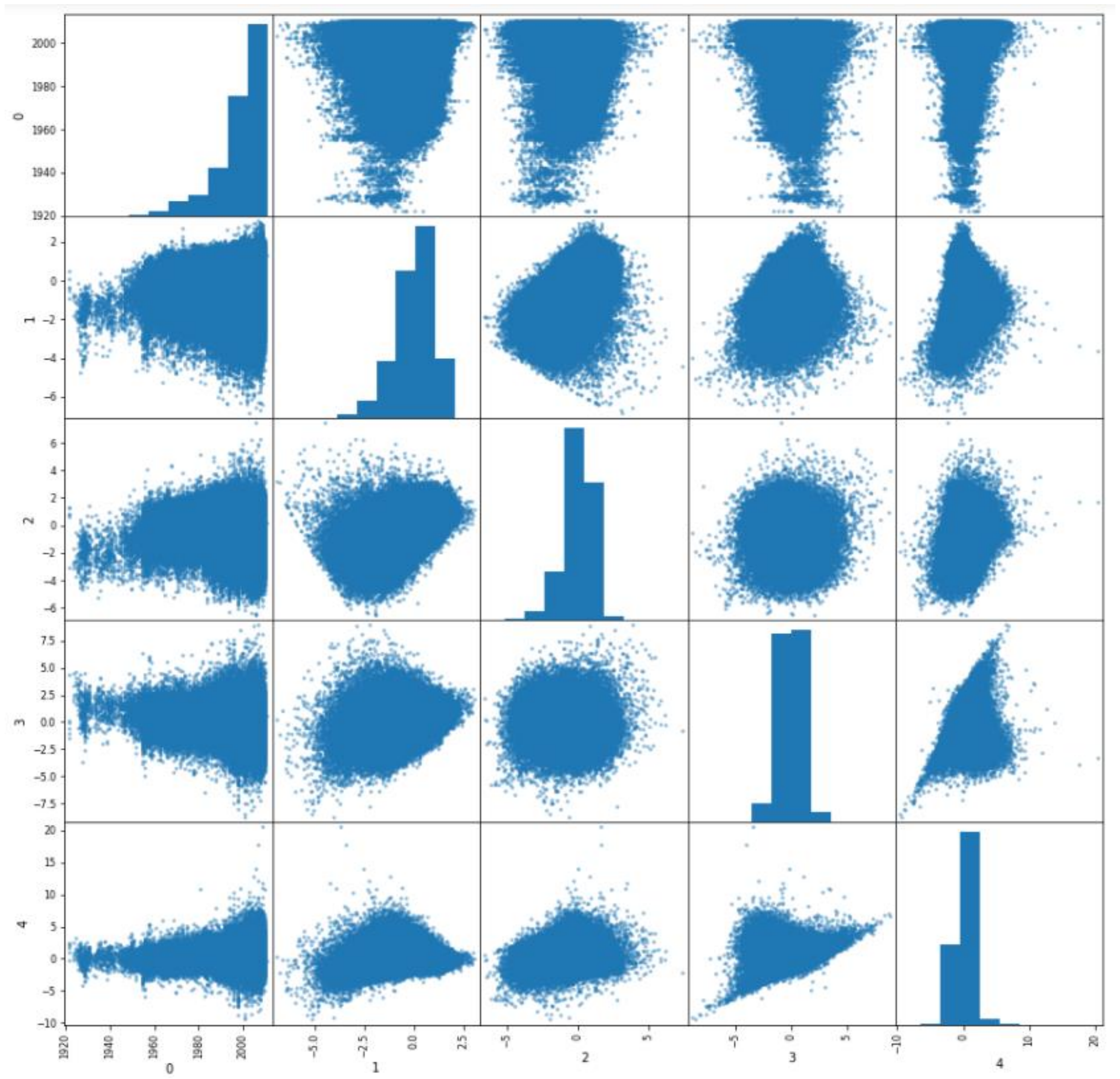
But the dataset description mentions *78 covariances* and *12 variances*. What we actually think is that the first 12 features represent the timbres ! Or maybe their mean value or something related. That would make sense because when we look at the distribution of the variables, We can cluster them into the first 12, the next 12, and the 66 remainings. The remaining 66 should represent covariances, because they have the same normal distribution, with big outliers, and can take on values ranging from -10000 to 10000 approximately. The features 13 to 24 should represent variances, because they only have positive values and kind of a χ^2 distribution. The first 12 features can then only represent the timbres themselves.

That would mean we actually have 12 timbres values (probably the mean over the song), with their variance and covariances.

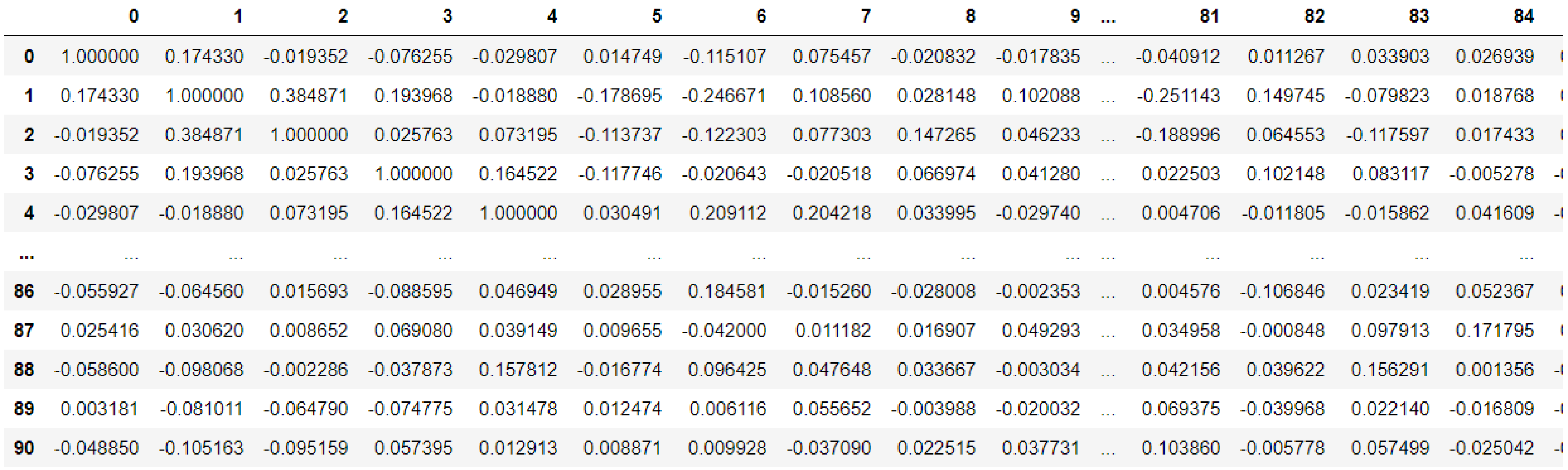
- Scatter plot

When we compare the target to the features (first row and column), we see that the data becomes more and more *spread* when the year increases : this can be explained by the distribution of the target as seen previously.

There are no indications (at least from the above) that it is useful to make any transformation to better suit the target (no logarithmic or exponential shape etc) we still could try computing the relationship between the target and all features just to be sure.



- Correlation matrix



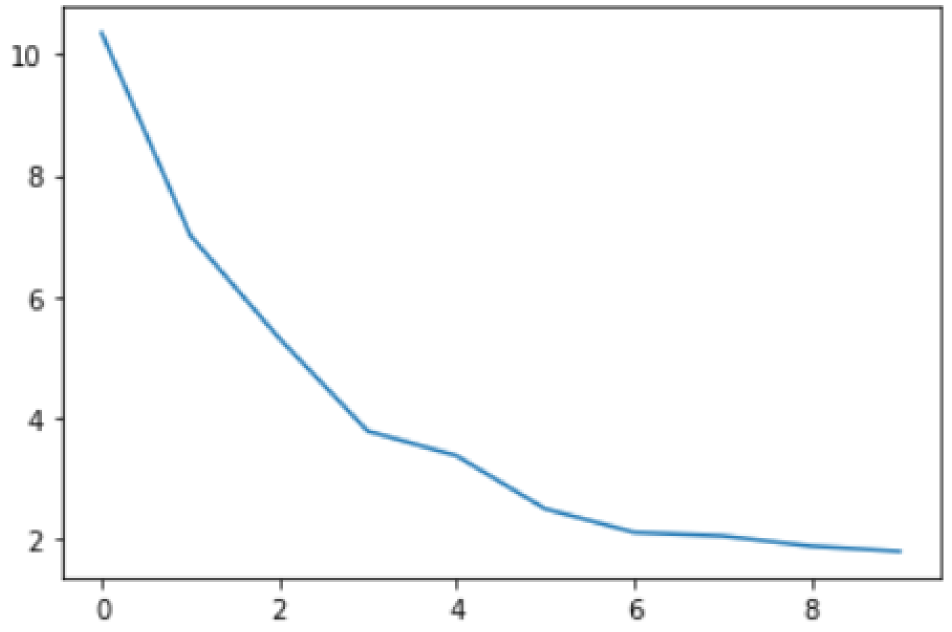
	0	1	2	3	4	5	6	7	8	9	...	81	82	83	84
0	1.000000	0.174330	-0.019352	-0.076255	-0.029807	0.014749	-0.115107	0.075457	-0.020832	-0.017835	...	-0.040912	0.011267	0.033903	0.026939
1	0.174330	1.000000	0.384871	0.193968	-0.018880	-0.178695	-0.246671	0.108560	0.028148	0.102088	...	-0.251143	0.149745	-0.079823	0.018768
2	-0.019352	0.384871	1.000000	0.025763	0.073195	-0.113737	-0.122303	0.077303	0.147265	0.046233	...	-0.188996	0.064553	-0.117597	0.017433
3	-0.076255	0.193968	0.025763	1.000000	0.164522	-0.117746	-0.020643	-0.020518	0.066974	0.041280	...	0.022503	0.102148	0.083117	-0.005278
4	-0.029807	-0.018880	0.073195	0.164522	1.000000	0.030491	0.209112	0.204218	0.033995	-0.029740	...	0.004706	-0.011805	-0.015862	0.041609
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
86	-0.055927	-0.064560	0.015693	-0.088595	0.046949	0.028955	0.184581	-0.015260	-0.028008	-0.002353	...	0.004576	-0.106846	0.023419	0.052367
87	0.025416	0.030620	0.008652	0.069080	0.039149	0.009655	-0.042000	0.011182	0.016907	0.049293	...	0.034958	-0.000848	0.097913	0.171795
88	-0.058600	-0.098068	-0.002286	-0.037873	0.157812	-0.016774	0.096425	0.047648	0.033667	-0.003034	...	0.042156	0.039622	0.156291	0.001356
89	0.003181	-0.081011	-0.064790	-0.074775	0.031478	0.012474	0.006116	0.055652	-0.003988	-0.020032	...	0.069375	-0.039968	0.022140	-0.016809
90	-0.048850	-0.105163	-0.095159	0.057395	0.012913	0.008871	0.009928	-0.037090	0.022515	0.037731	...	0.103860	-0.005778	0.057499	-0.025042

For most variables, the correlation is very **low** (<0.1) however we see some values over 0.2 which can be interesting, and also we are directly interested in the highest correlations between the target and the features as we would like to fit a first linear regression model.

There are 3 features which have a correlation with the target higher than 0.1 : feature 1, 6 and 63.

PCA

- After normalizing and splitting the data, we performed a PCA to simplify it. We know that according to the correlations, this cannot be very efficient, but we thought it would maybe make things clearer



Explained variance with 10 principal components

How to fit a model ?

- The difficulty of this dataset is the lack of correlation between the different features. It is also difficult to come up with any transformation, even after having determined what they mean.
- Also there are a lot of them, and we cannot really choose some among others because none seem to have more weight in predicting the release year. One thing we could do however is fitting model using separately the variance, covariance and mean of the timbres

How to fit a model ?

- Because there is no correlation between the features, standard machine learning models are not very efficient. Especially multiple linear regression.
- The way the features are spread are also a difficulty on top of all this, because we would like to fit a model which is accurate on all the dataset range, and not only years from 1990 to 2010.

Our conclusion

- After implementing some classic ML models the results we obtained were so bad that we decided to try another approach using deep learning
- The large quantity of learning data allows us to create such model, and there seems to be no other alternative
- After trying just one model with almost randomly chosen parameters we obtained the best results so far, we expect it is a matter of tuning and varying the input variables to get an efficient model

Conclusion

- We fitted efficient models using keras, and implemented an interface
- All the details are in the Jupyter Notebook