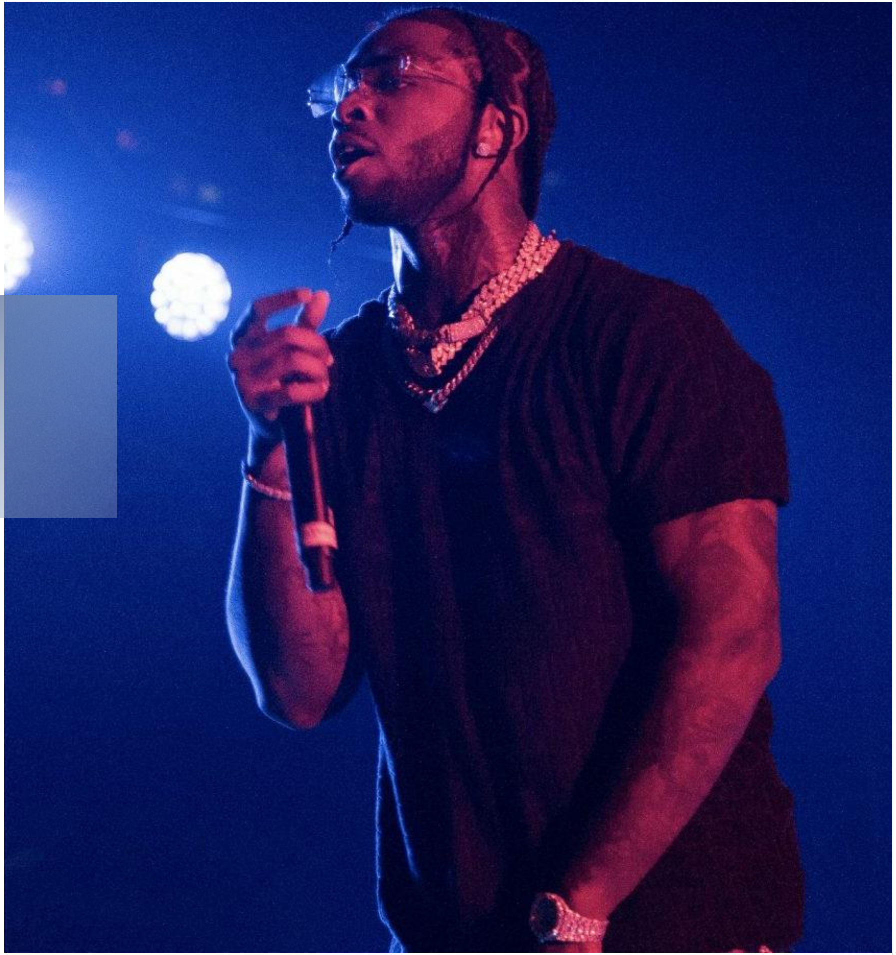
DECADE PREDICTION OF A SONG

Python Data Science – Django API

Valentin TASSEL – Ariel Tedgui

ESILV
ENGINEERING SCHOOL DE VINCI PARIS

TABLE OF CONTENTS



ABOUT THE PROJECT

The Million Song Dataset (MSD) is a freely-available collection of audio features and metadata for a million contemporary popular music tracks.

We have a dataset containing a subset of the MSD and contains audio features of songs with the year of the song.

The purpose being to predict the release year of a song from audio features.

Existing notebooks on the subject:

Some Kaggle notebooks that try to make a decade classification or just a data exploratory. Most popular one use SVC Classifier and obtains 48% of accuracy with 47% of macro avg precision.
Reference: https://www.kaggle.com/anshuljdhingra/predict-release-timeframe-from-audio-features

Databricks notebook using Linear Regression to predict the precise release year of the song.
Reference: https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3175648861028866/3427136292952836/657465297935335/latest.html

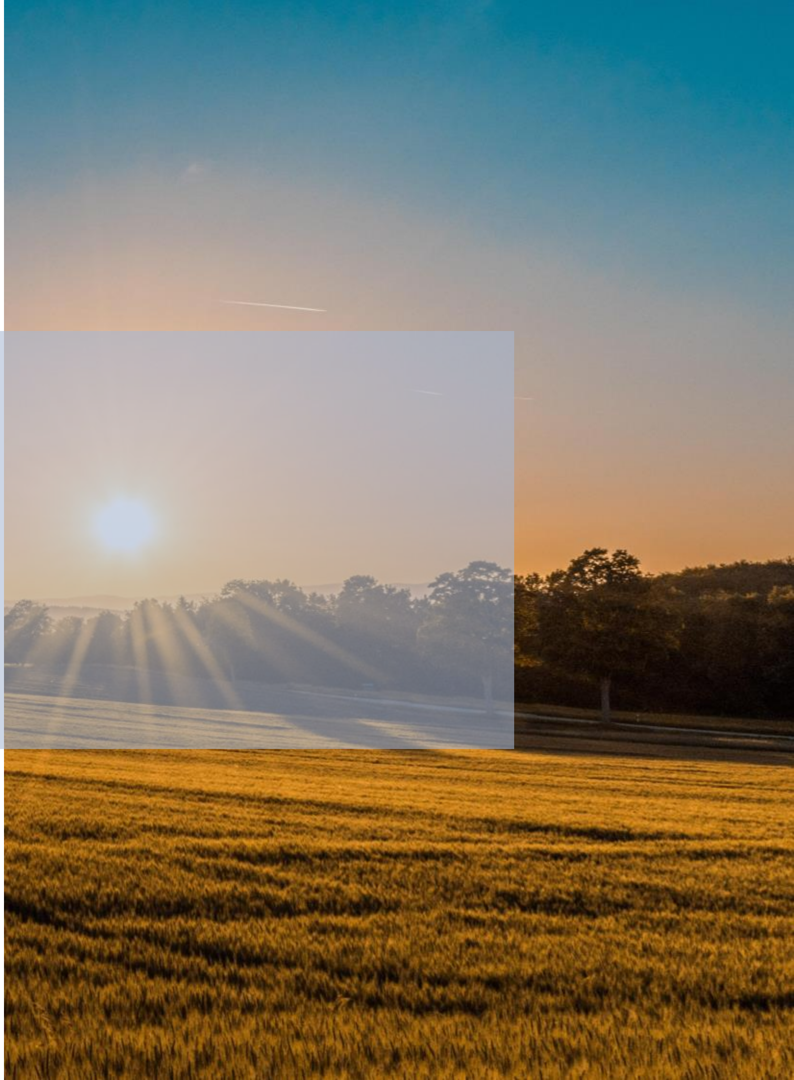
These are inspirations for our project.

WE HAVE

515 345

LINES

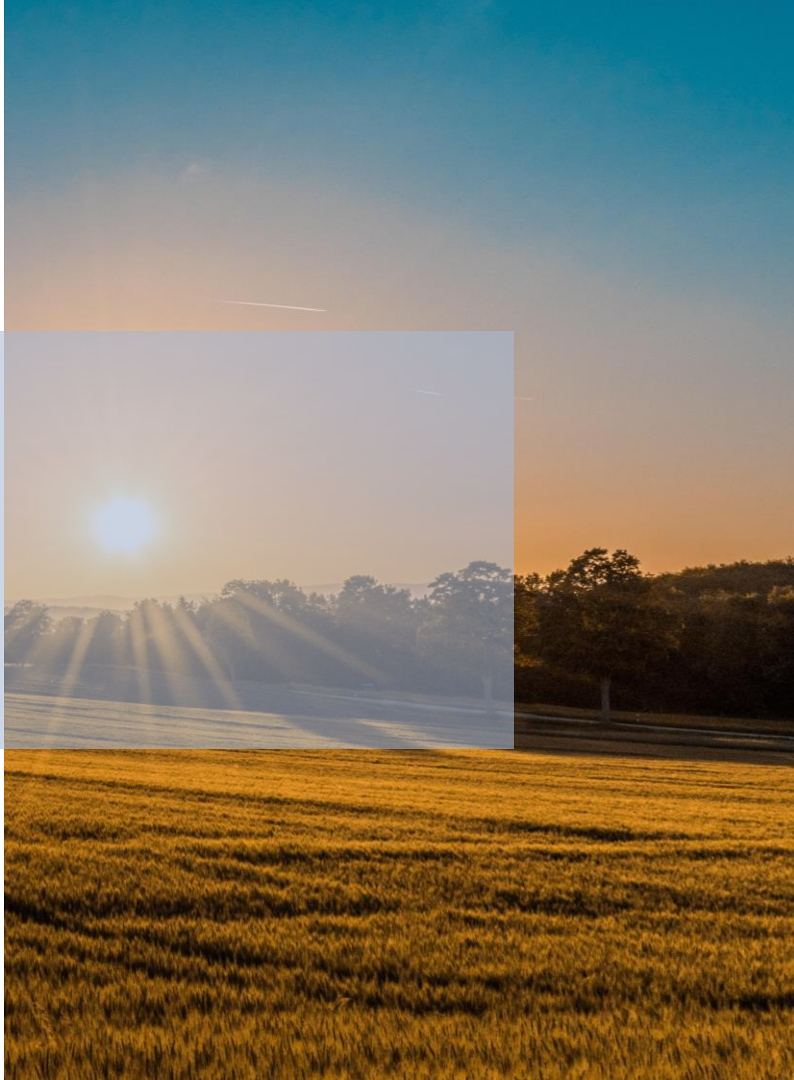
Without
missing
values



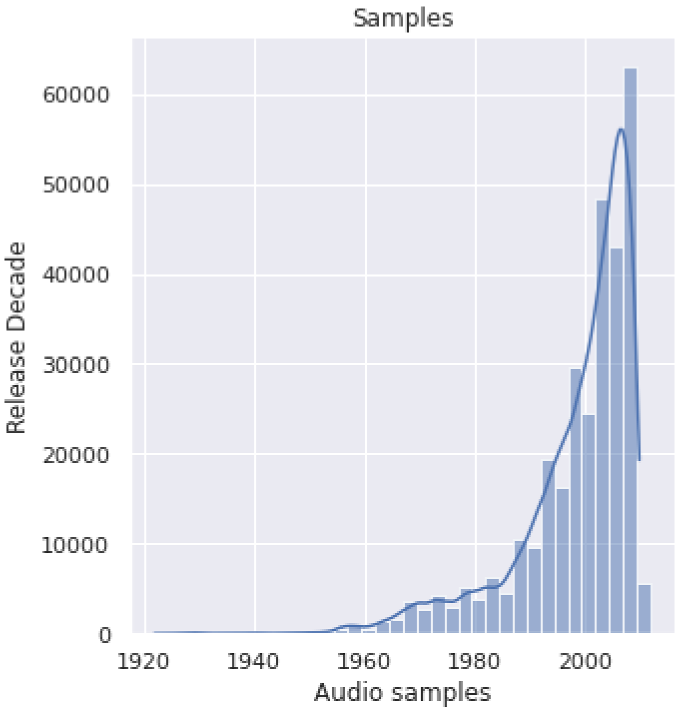
FROM YEAR

1922 to 2010

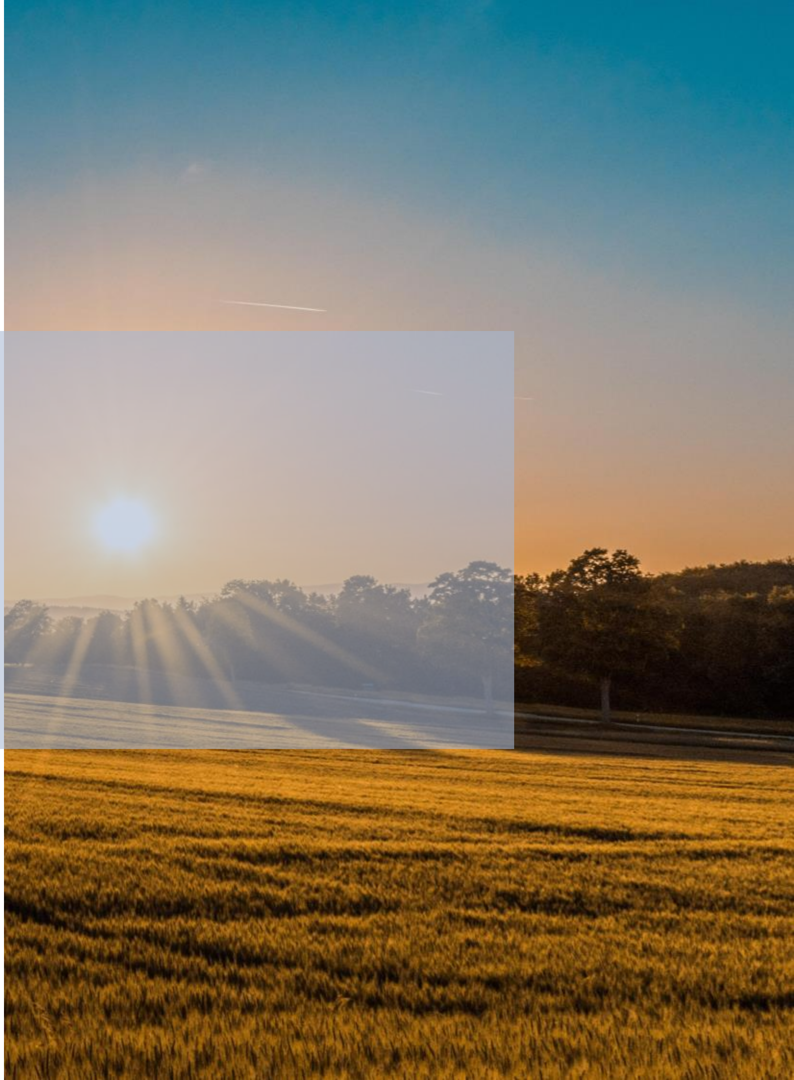
In the training set



DISTRIBUTION OF THE YEAR LABEL



We observe most of the sample around the 2007-2012 years



FEATURES

TIMBRE AVERAGE

12 columns

12 segment describe by 12-dimensional timbre vector

TIMBRE COVARIANCE

78 columns

Appears to be the covariance over the 12 Timbre Average features (Timbre Average 1 through Timbre Average 12)

DATE

1 column

Year date of the release

DECADE

1 new columns that will be created

New columns that we will create that correspond of the decade, based on the year date

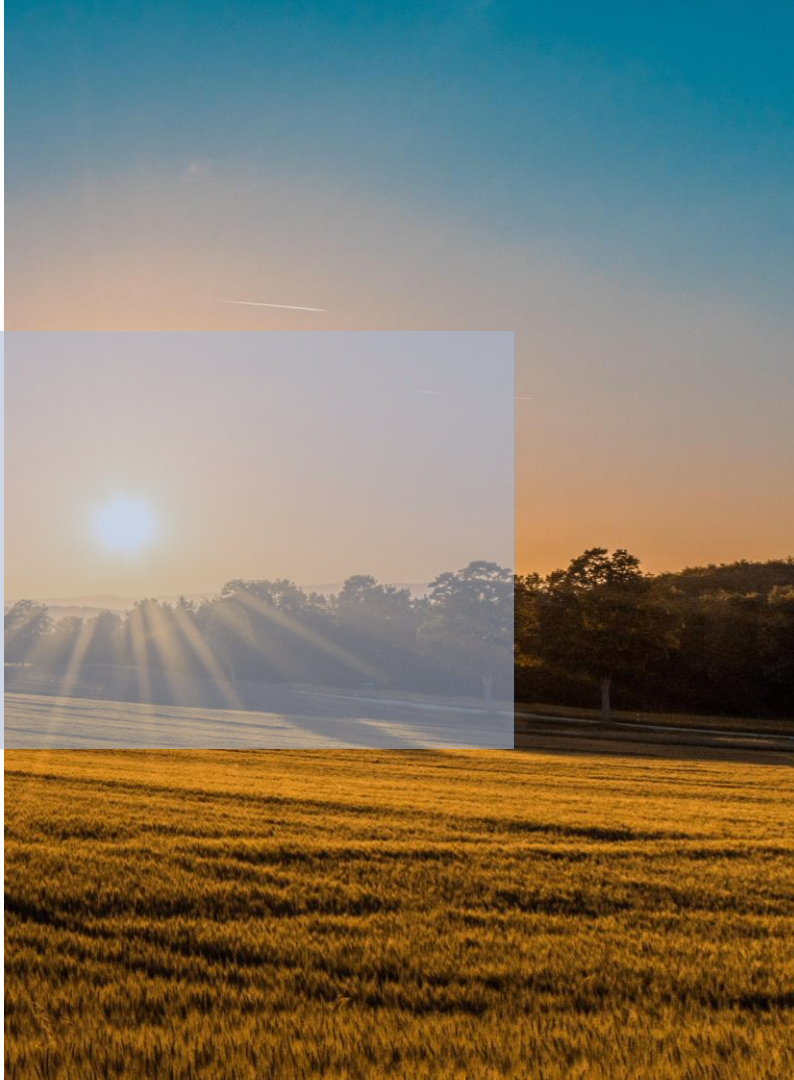
OUR GOAL

02

WE WANT

Predict the right release decade for each song using its audio features.

Make an API that automatically returns the right decade.

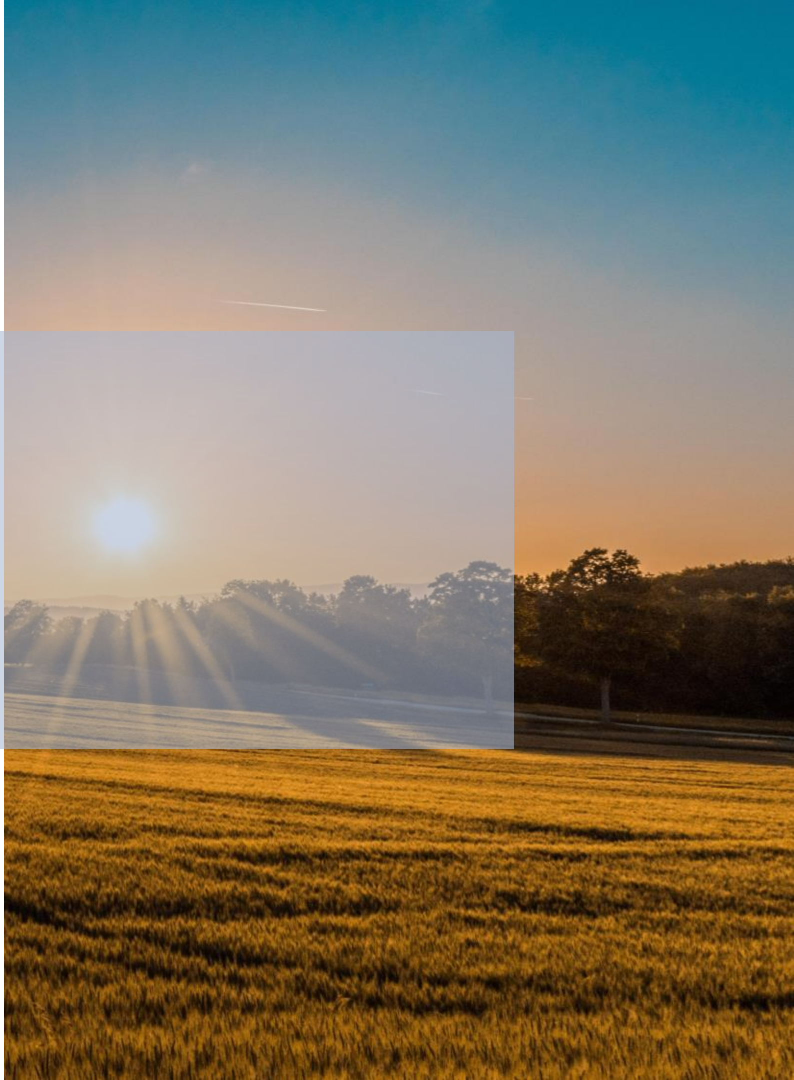


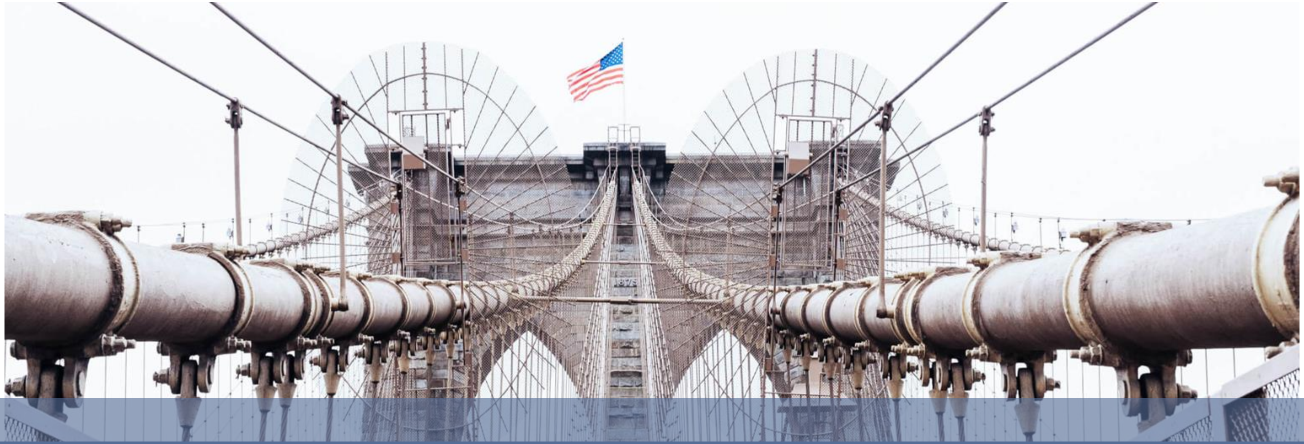
OUR REFLEXION

03

WE THINK ABOUT

resolve the problem of computation time due to the large number of features, make the numeric feature usable by the future model, and obtain significant result.





CONSTRUCT A NEW LABEL

We construct the label decade that is the decade of the release year to calculate the decade we use this formula :
Decade = (Date/10)*10

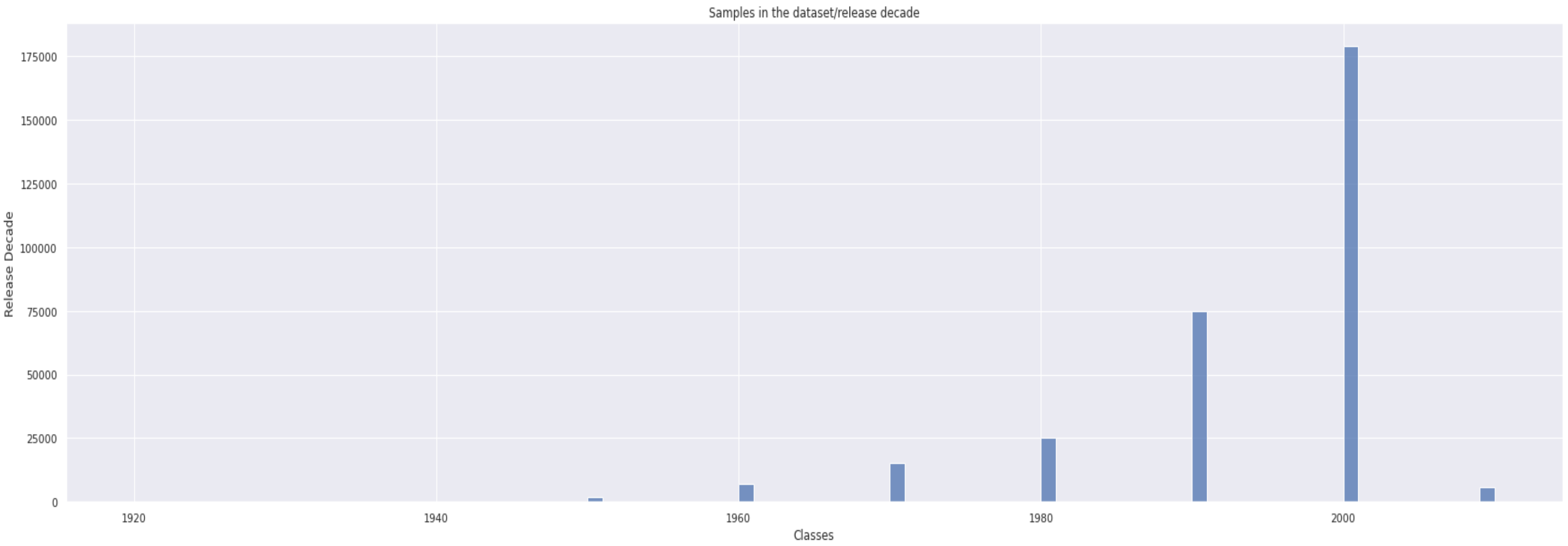
STANDARDIZE THE FEATURES

We standardize the features because it can be helpful in cases where the data follows a Gaussian distribution.

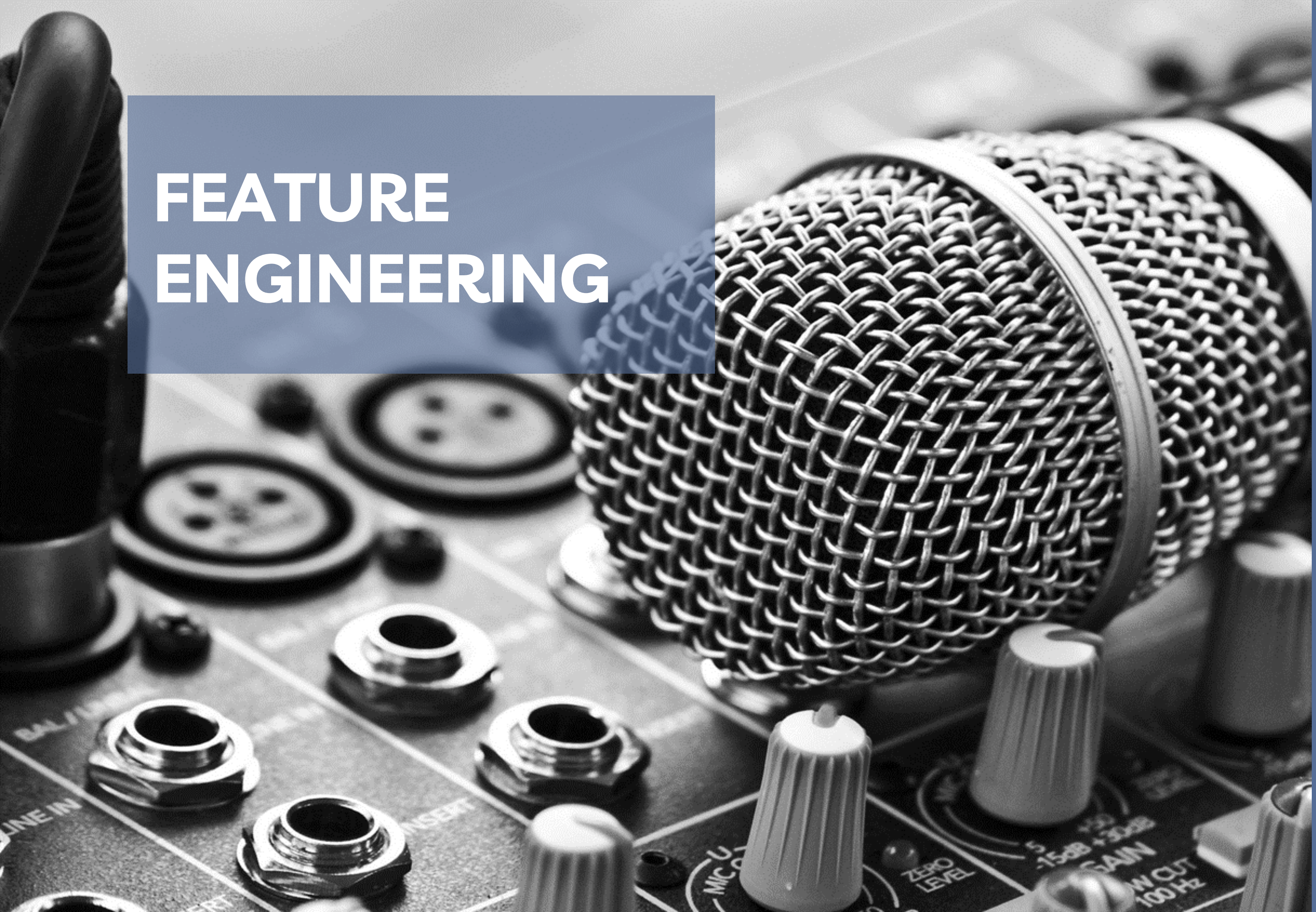
REDUCE THE DIMENSION

We are in presence of a large quantity of variables, we want to reduce the dimensionality by projecting the data to a lower dimensional subspace which capture the essence of the data **to reduce the time of computing**

DISTRIBUTION OF THE DECADE LABEL

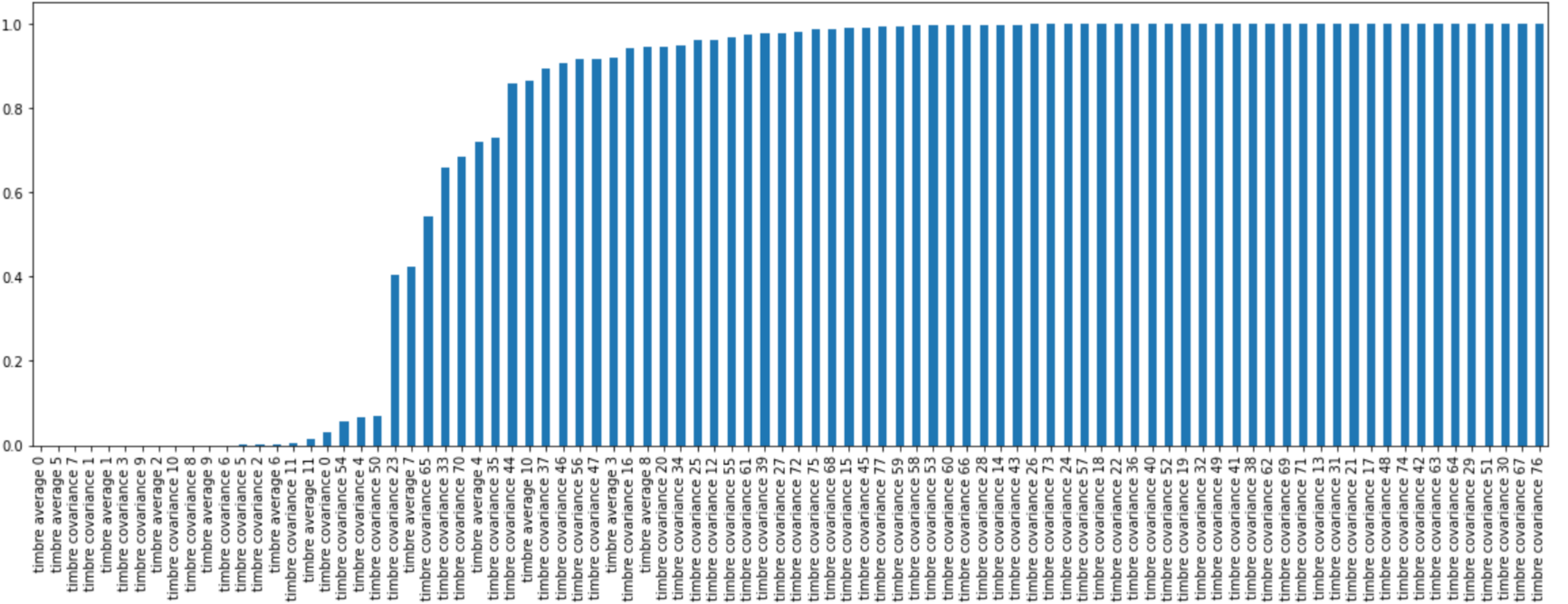


FEATURE ENGINEERING



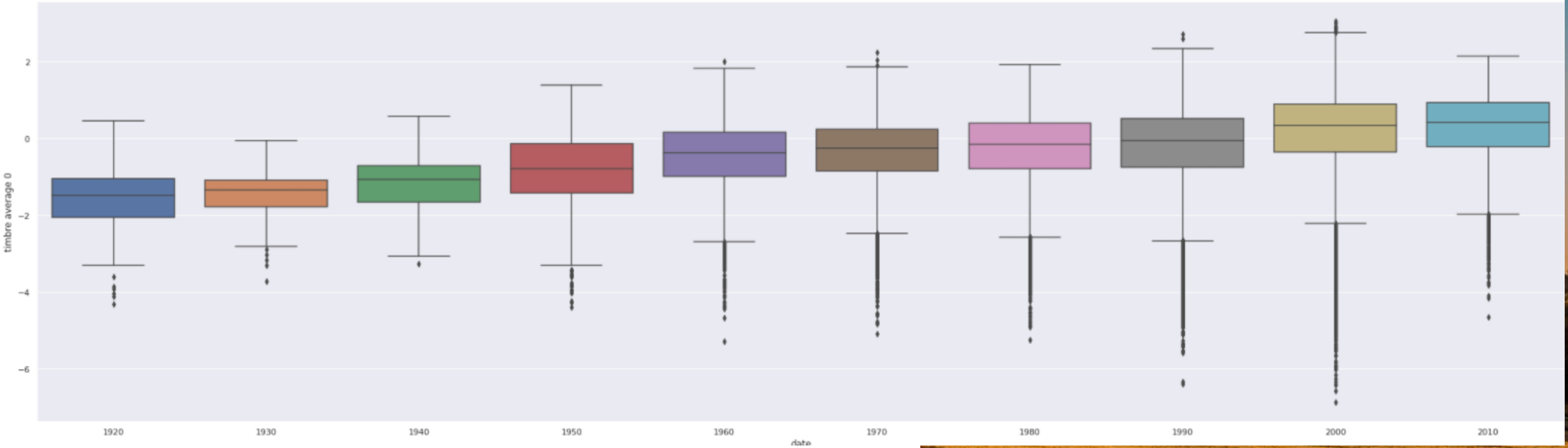
04

CHI-2 FEATURE IMPORTANCES



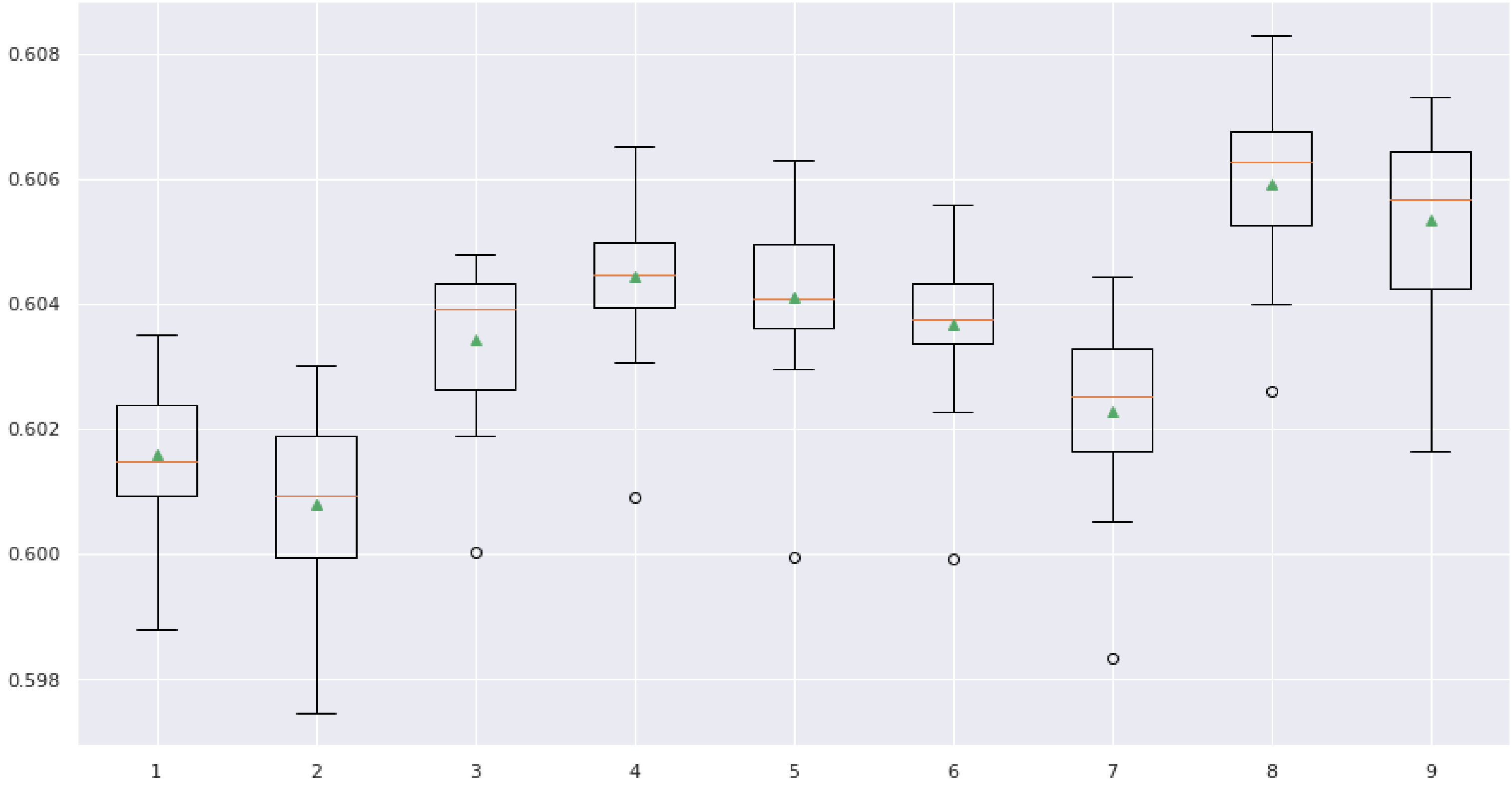
This graph shows us the importance of the features located at the left of the graph: timbre average 0, timbre average 5...
We must take in account these features in priority.

FEATURE IMPORTANCES



For timbre average 0 we observe the linearity with the decade of this value that justify the importance score.

LINEAR DISCRIMINANT ANALYSIS



We test Linear Discriminant Analysis with multiple output dimension and see the result of a naïve bayes classifier to find the best dimension : **It's 8**

OUR SOLUTIONS

05

Here our solutions:

Make a **benchmarking** of different models in order to find the one the more adapted to our problem.
The benchmark take the following measures: Accuracy, Precision, Recall, F1 Score, ROC Score

1. Realize a baseline model: **K-NN** (k nearest neighbors' algorithm) non-parametric method used for classification and apply an hyperparameter tuning.
2. Realize a **Random Forest Classifier** known to work well with non-linear data, runs efficiently on large dataset and have a lower risk of overfitting.
3. Realize a **XG Boost Classifier** known to be a high performant algorithm but complicated to fine-tune.
4. Realize a **Neural Network** known to be a high performant algorithm but complicated to fine-tune.

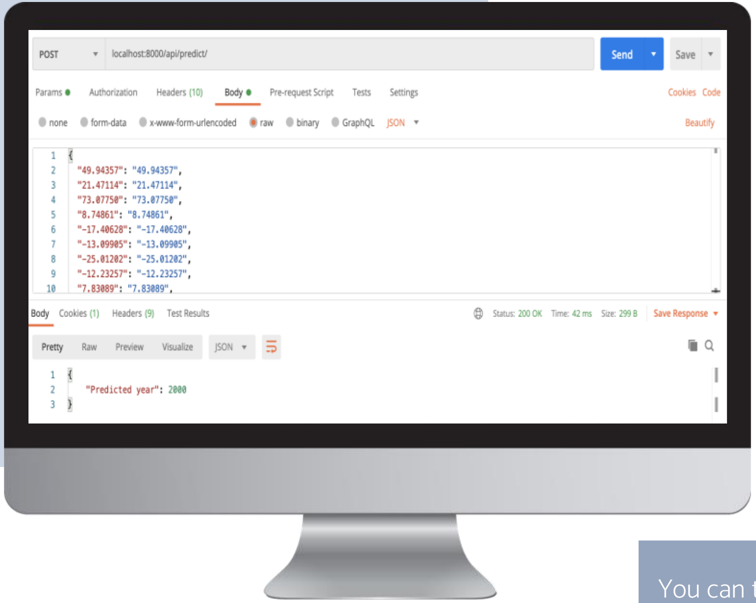
Our benchmark:

Model name	Accuracy	Precision	Recall	F1 Score
K-NN	0.600937	0.283826	0.198130	0.214031
Random Forest	0.631868	0.481588	0.175073	0.190031
XGB Classifier	0.634109	0.398034	0.178039	0.192986
Neural Networks	0.63213	0.358048	0.173856	0.184296

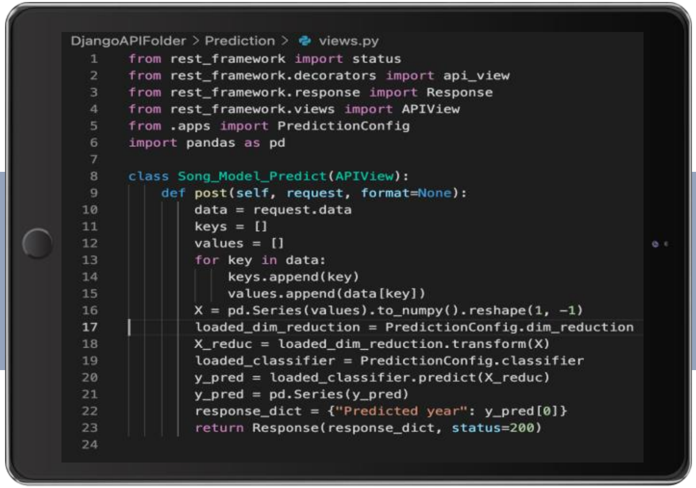
We observe a consequent increase in the benchmark when we increase the complexity of the algorithms. However, the models stagnates around 0.63 of accuracy.

We decide to put in production the Random Forest model due to its high precision and its good accuracy on the validation set.

DJANGO API



You can test our API from this GitHub repository : https://github.com/ArielTed/Song-Prediction



The Django API contains :

- A POST route /prediction
- The user can send a request to that route with a song data in the body of the request
- The user then receive a response with the predicted year of the song thanks to our model
- With that response, we can use it and display it on a website.

THANKS

www.valentintassel.com