# Utopia assessment

Andrea Maioli

November 7, 2022

## Contents

# 1 Introduction

The challenge consist in forecasting the future plays of a fixed set of songs.

My purpose is of course provide the best forecast possible in terms of performance metrics but also to ***decompose the result***.

I will provide results of different models with the purpose of show how much the autoregressive component contribute with respect to the calendar information.

This can be done building a model based only on the calendar and comparing with a model depending both on calendar at autoregressive variable (namely lagghed values of the target).

I will build models in two framework: ***deterministic*** and ***bayesian***. In the **determinist** framework I will produce classic scalar forecast value. In the **bayesian** framework, for each sample, I will produce a ditributional output.

# 2 Data Analysis

First of all I have to explore the data to better understand the problem and to make a proper model design.

I will follow these steps:

1. Data exploration and timeseries properties by tracks
2. Correlation between the tracks for different time granularity
3. Seasonal properties
4. Features engineering

## 2.1 Data structure and exploration

The dataset describe the daily number of plays of 6 tracks among different stations between 01/01/2019 and 31/12/2020.

The columns of the dataset are `Day, TrackId, StationId, Plays`.

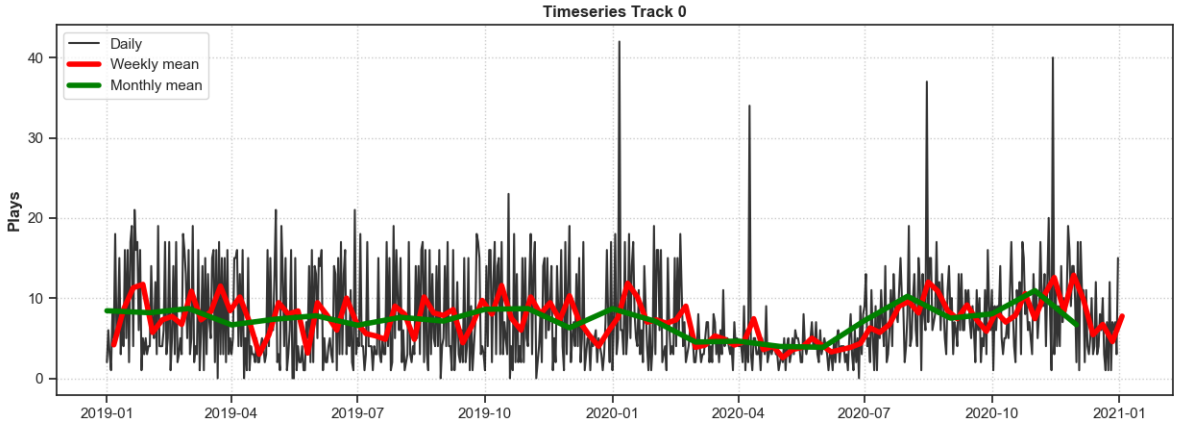
For simplicity the tracks ids and stations ids have been mapped to integers, in particular the tracks ids have been mapped to integers from 0 to 5.

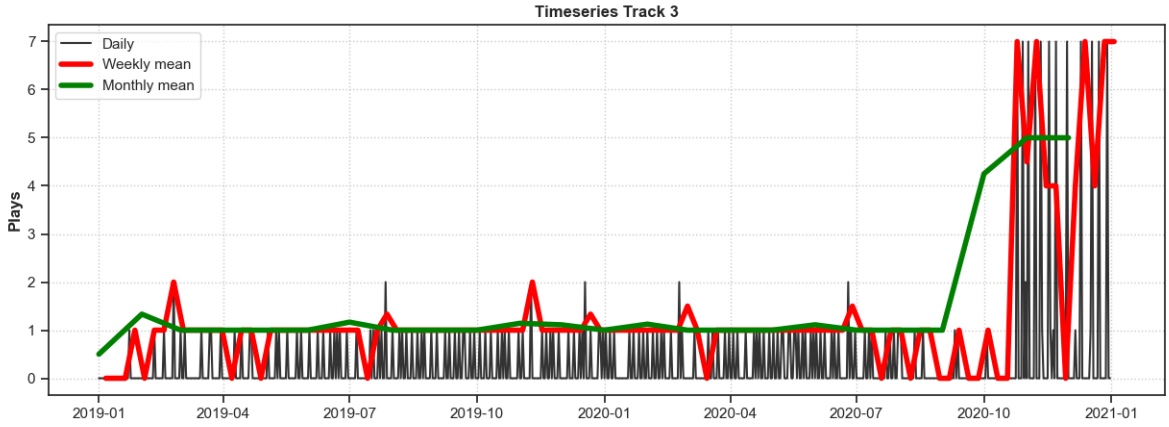
Since I am interested in predicting total amount of plays I've aggregated data with respect to TrackID. From this point on I will always use aggregated data.

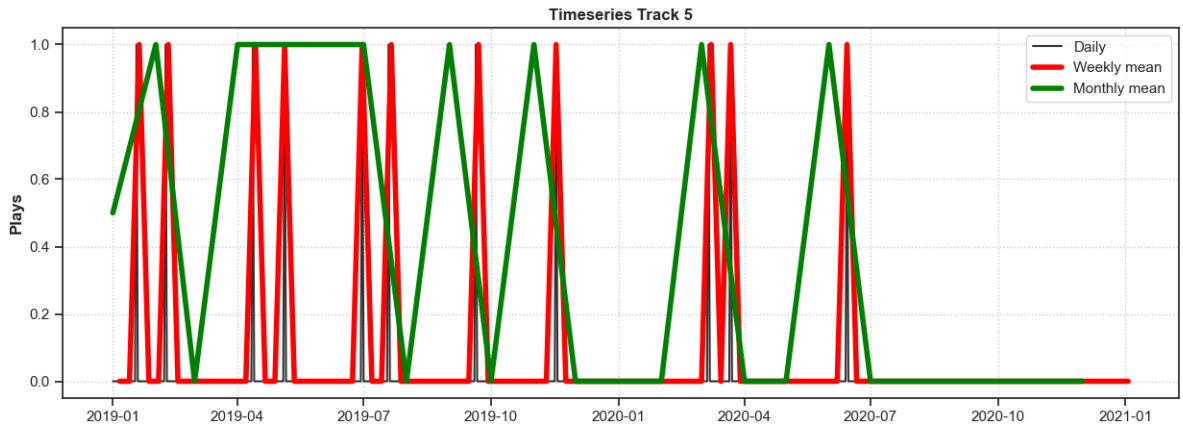
These are general statistics of each track:

| | Track 0 | Track 1 | Track 2 | Track 3 | Track 4 | Track 5 |
|---|---|---|---|---|---|---|
| count | 731 | 731 | 731 | 731 | 731 | 731 |
| mean | 7.2 | 0.5 | 7.1 | 0.3 | 2.7 | 0 |
| std | 5.8 | 1.3 | 5.2 | 1 | 4.5 | 0.1 |
| min | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 perc | 3 | 0 | 3 | 0 | 1 | 0 |
| 50 perc | 5 | 0 | 6 | 0 | 2 | 0 |
| 75 perc | 11 | 1 | 10 | 0 | 3 | 0 |
| max | 42 | 26 | 45 | 7 | 35 | 1 |

Let's have a look at few plots and then make some observations.





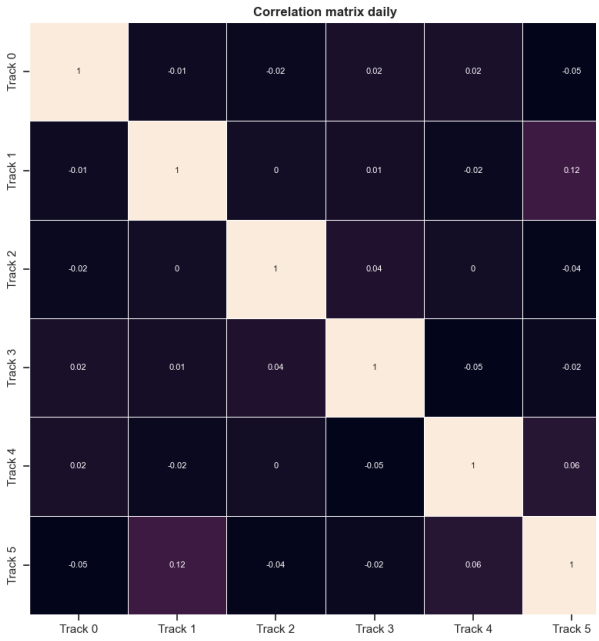

**Observations:**

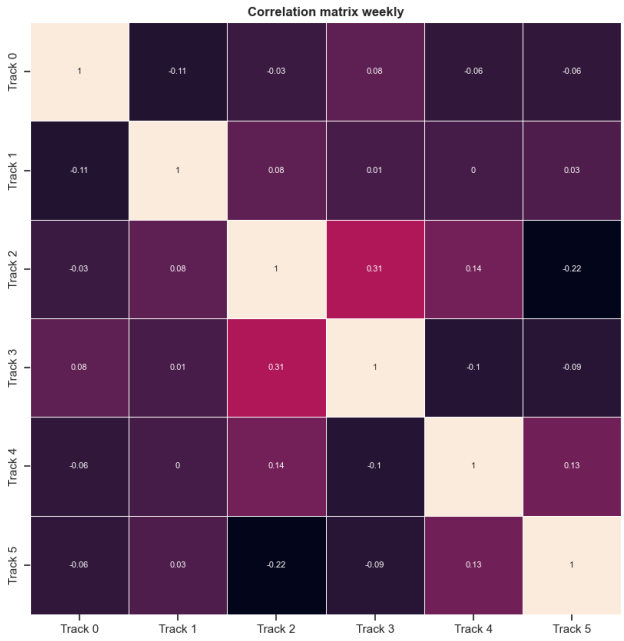
- **Track 5** is 0 most of the times and it is played at most one time per day. It is the only song having these properties.
- **Track 0** and **Track 2**: have close properties and similar shape. In particular are the tracks with highest volume on average.
- **Track 1**: present a big outlier on 02/01/2020 but in general mostly assumes values 0 and 1 (see also *75 perc* value).
- **Track 3**: the plays distribution. drastically changes after 24/10/2020.

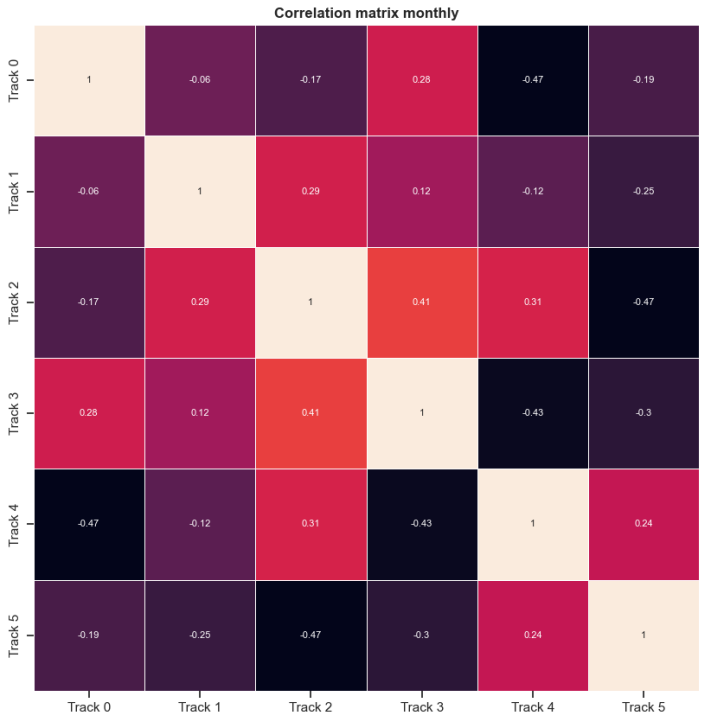
## 2.2 Correlation

If tracks are correlated I could forecast the six series simultaneously and take advantage of additional information obtained from timeseries correlation.

Exploring correations between timeseries for different granularity turns out tracks are not correlated as shown by the matrixes below:

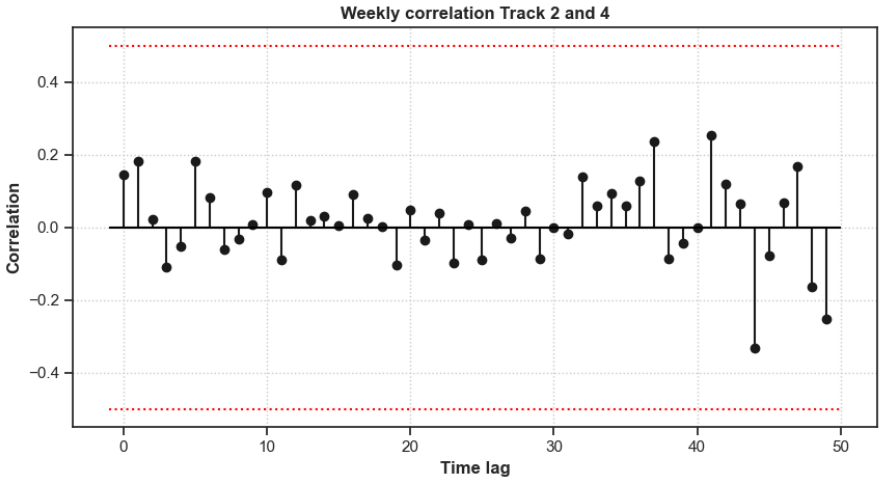


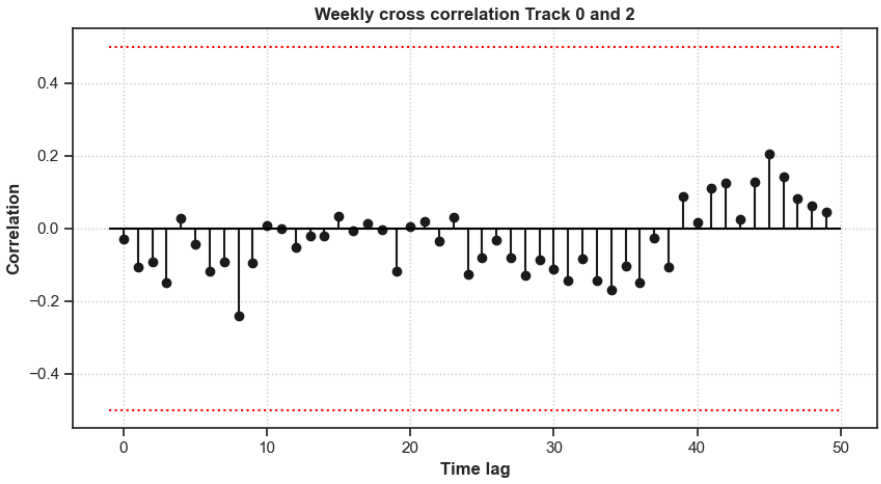



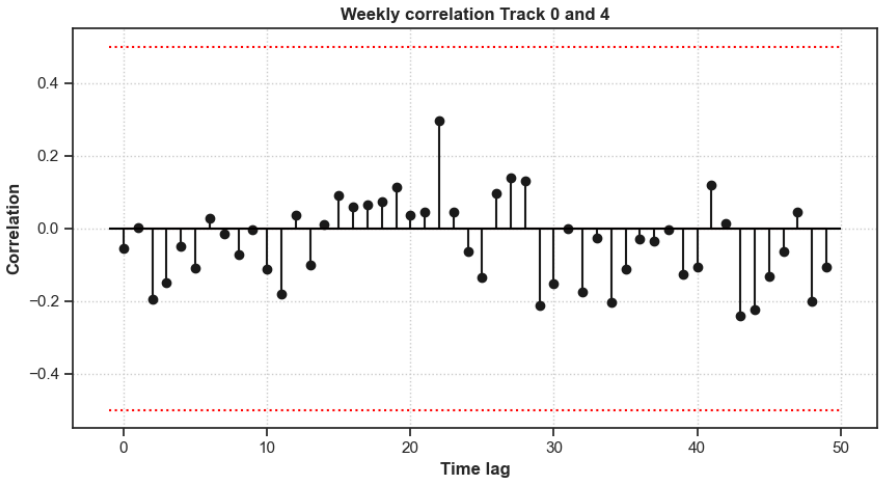




It could be that the tracks are lagged correlated, which means that there is a high correlation between a track and the lagghed time series of another track. I explore only relation between tracks 0, 2 and

4.



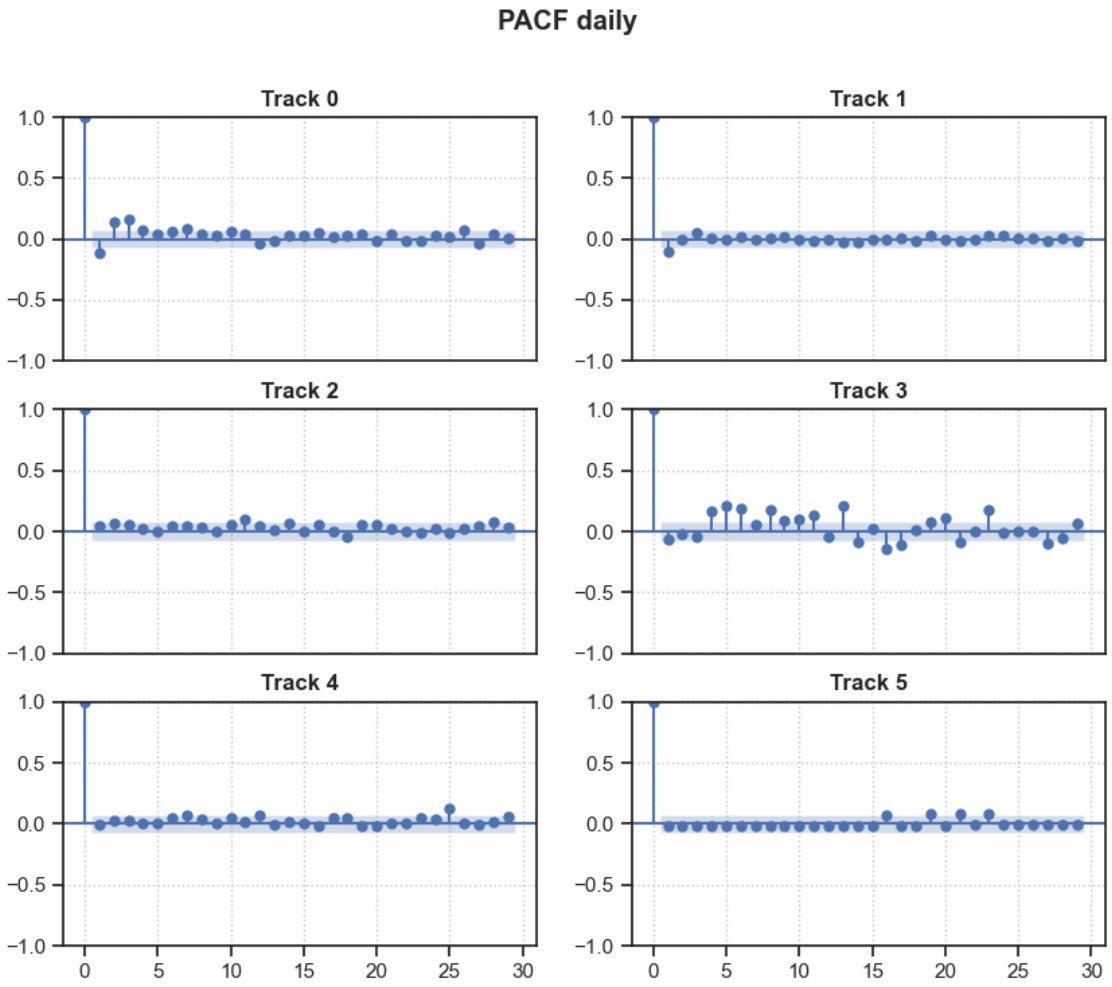



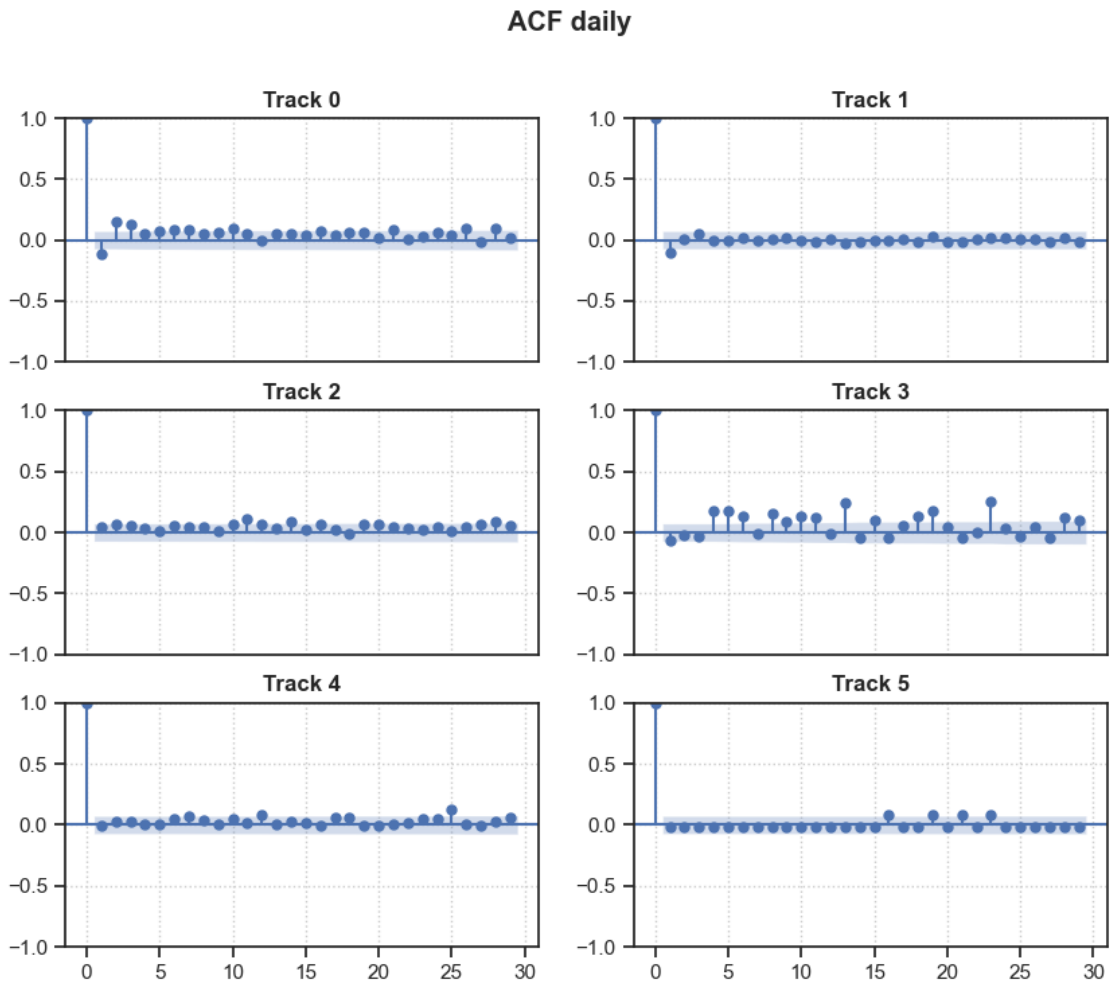

Tracks do not present high correlation and even cross correlation with different lags is very low. ***We can then work on each track timeseries separately and avoid working with multioutput models.***

## 2.3 Seasonal properties

I explore how much the timeseries depends on past values. In particular I want to verify if there are calendar patterns (*weekly, monhtly...*) and dependency on close past values.

For this purpose I compute the ***PACF and ACF*** of each track for different granularity.





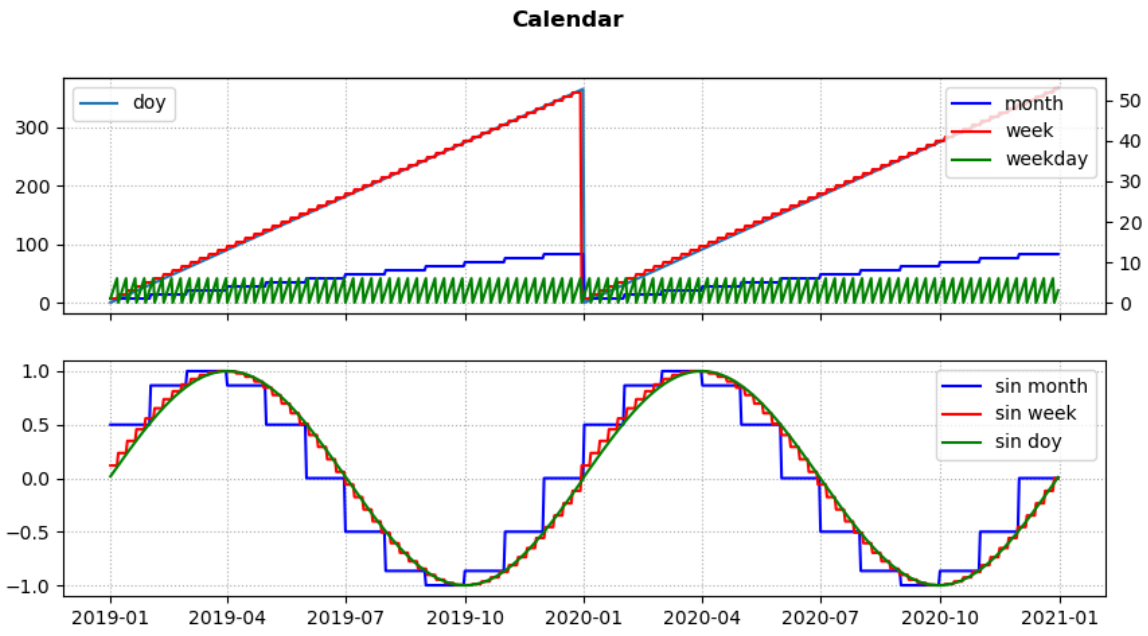

The results tell me that the autocorrelation component for all granularity is in general weak and in some cases absent. This means that autoregressive models must be initialized with low **p, q** parameters

## 2.4 Exogenous variables

The only exogenous variable avaiable is the valuedate. I decompose it and create **calendar features** that will be used as exogenous variables.

The following variables are built:

- **Standard calendar**: `year, month, week, day, weekday, doy`
- **Seasonality**: sin and cos of `month, doy, week`




# 3 Forecast

## 3.1 Deterministic framework

I start forecasting deterministic values. In particular I want to assess how many leadtime it takes to the calendar component to outperform the autoregressive one.

### 3.1.1 Models

Since we have seen that both autoregressive and calendar features actually have little impact I will not use complex non linear models.

I will use the following models:

- ***PoissonLinearRegression***: fed only with calendar features defined in `config.yaml` file.
- ***ARIMA***: an arima model with order param optimised on the training set via gridsearch.
- ***AR***: an arima model with order `(1,0,0)`

Note that both ***ARIMA*** and ***AR*** do not use calendar informations.

### 3.1.2 BackTest Methodology

When using the Poisson linear I produce forecast simply fitting linear params on training set and forecasting on test set.

For both the autoregressive models I apply a ***rolling fit/predict*** metodology:

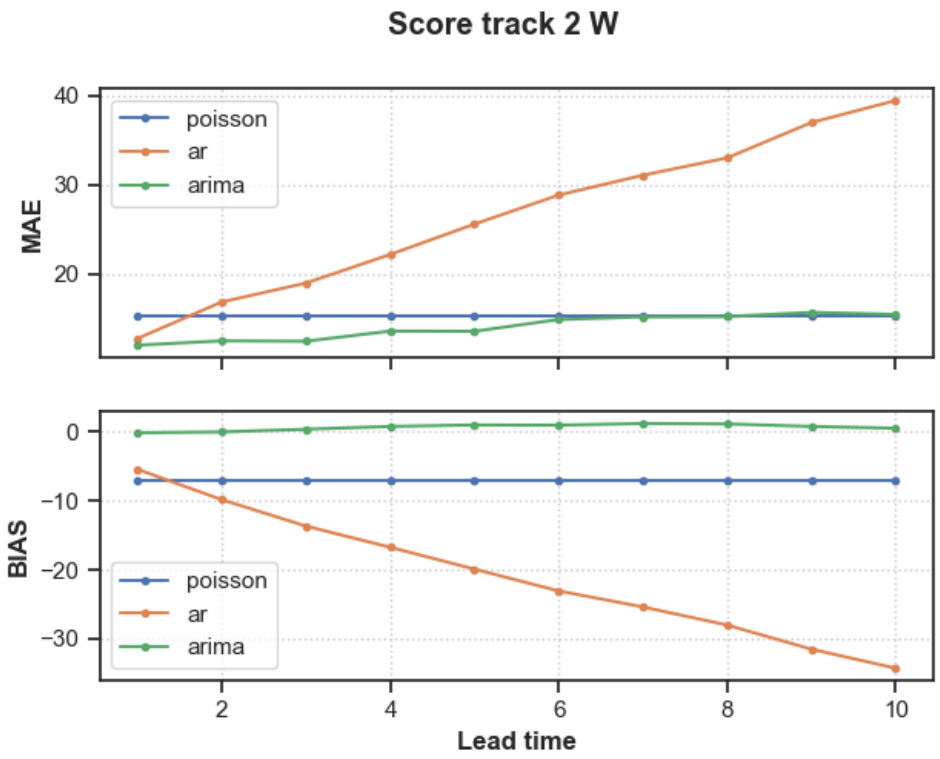
1. Fix valuedate `d` in test set
2. Slice time interval `(d-train_step_size, d-1)` from target data.

3. Fit autoregressive model on the dataset you have just sliced
4. Generate forecast for the next `step_ahead` steps and assign leadtime.
5. Iterate for each sample in test set

The result is a prediction dataset with shape `(test sample, step_ahead)`. Of course we expect the performance to get worse as soon as the leadtime increase.

> The leadtime for which the error of the Poisson linear model is below the error of autoregressive models, ***is the leadtime for which I know I can rely only on the calendar information from then on.***

Of course the performance behavious must be discussed track by track. here I show just a plot of the performance curve. Details will be discussed below.




## 3.2 Bayesian framework

> First of all: ***Why a bayesian model?*** I have no idea.

I know nothing about music industry but I think that in general is now interesting to know the exact number of times a song will be played on radio.

Instead it seems natural to me to try to give an estimate and a lower/upper bound of what will be the times a song will be played.

For example for an artist is imporant his songs are famous enough (in order to sell tickets tour or merchandise) and this can be translated is hoping for the forecast not only to be greater that a certain

amount but also to have a tight forecast distrubtion so that there is low uncertainty about the forecast it self.

### 3.2.1 Model

I've fitted a ***GAM*** model with Poisson family distribution and log link function (the so called ***PoissonGAM*** model). I've focused on weekly seasonality.

I suppose observations of previous week are available so that I use them as regressor.

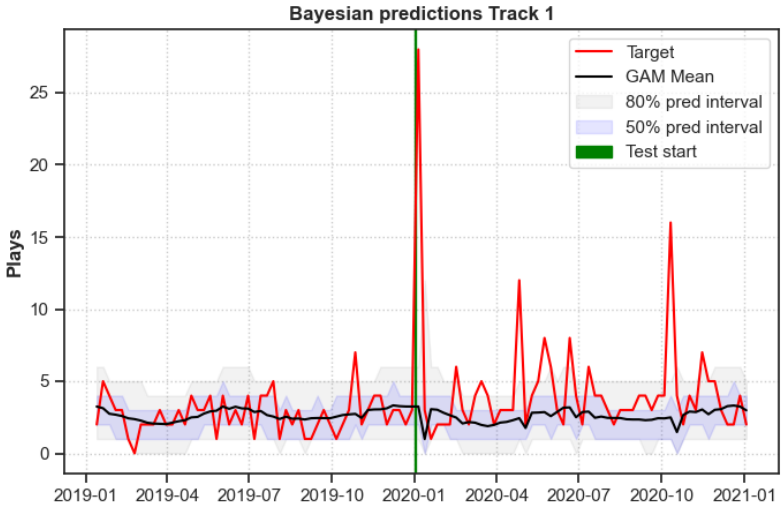
Namely the GAM formula is:

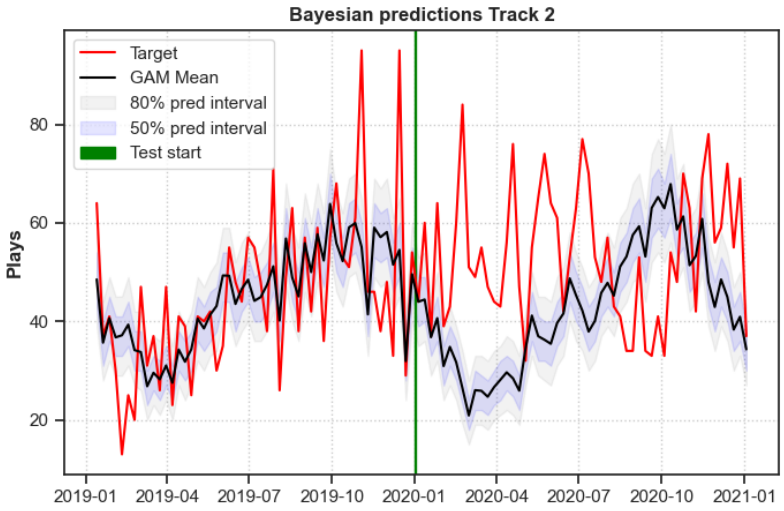
$$\log\left(Plays_t\right) \sim P\left(Plays_{t-1} + s(week_t)\right)$$

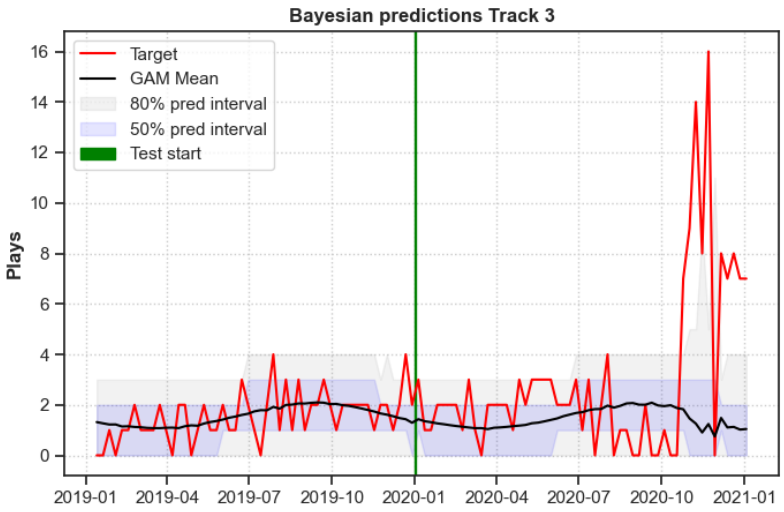
where $P$ is a Poisson distribution and $s$ is a cyclic spline.

I did not apply a time rolling retrain due to a lack of time. For this reason those tracks for which the calendar dependence change drastically from 2019 to 2020 perform poorly (see ***Track 2***).

Despite that there are cases in which the forecasts have a nice behaviour and in particular we can see target lying in the confidence interval most of the time (see ***Track 1*** and ***Track 3***).




Bayesian predictions Track 2
Target
GAM Mean
80% pred interval
50% pred interval
Test start
Plays
80
60
40
20
2019-01
2019-04
2019-07
2019-10
2020-01
2020-04
2020-07
2020-10
2021-01


Bayesian predictions Track 3
Target
GAM Mean
80% pred interval
50% pred interval
Test start
Plays
16
14
12
10
8
6
4
2
0
2019-01
2019-04
2019-07
2019-10
2020-01
2020-04
2020-07
2020-10
2021-01

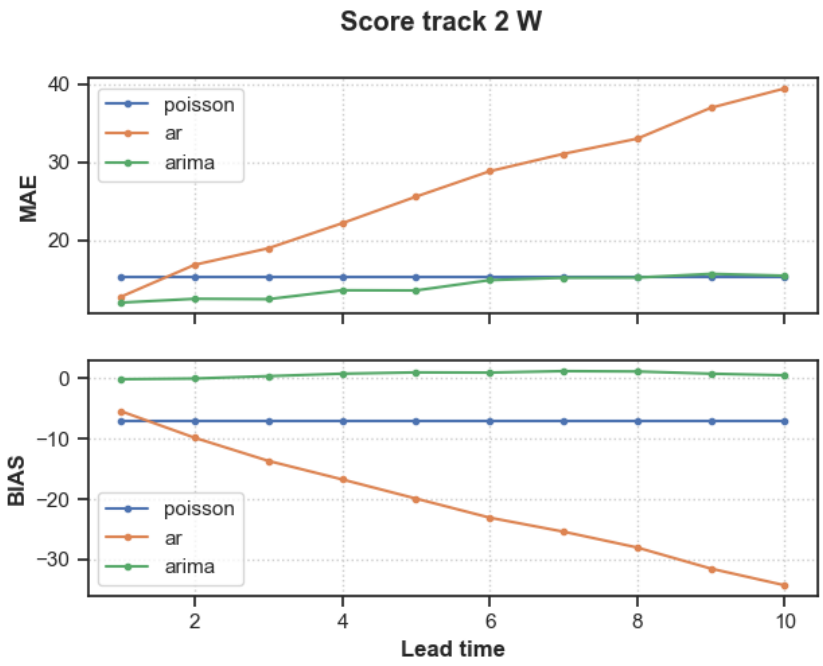
# 4 Conclusions

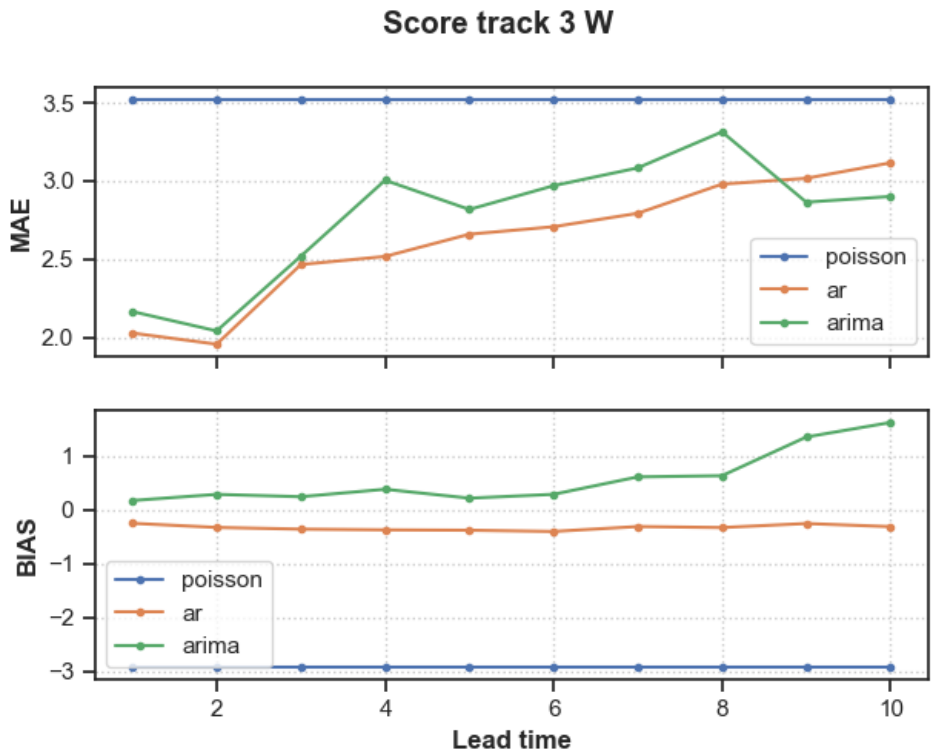
## 4.1 Performance

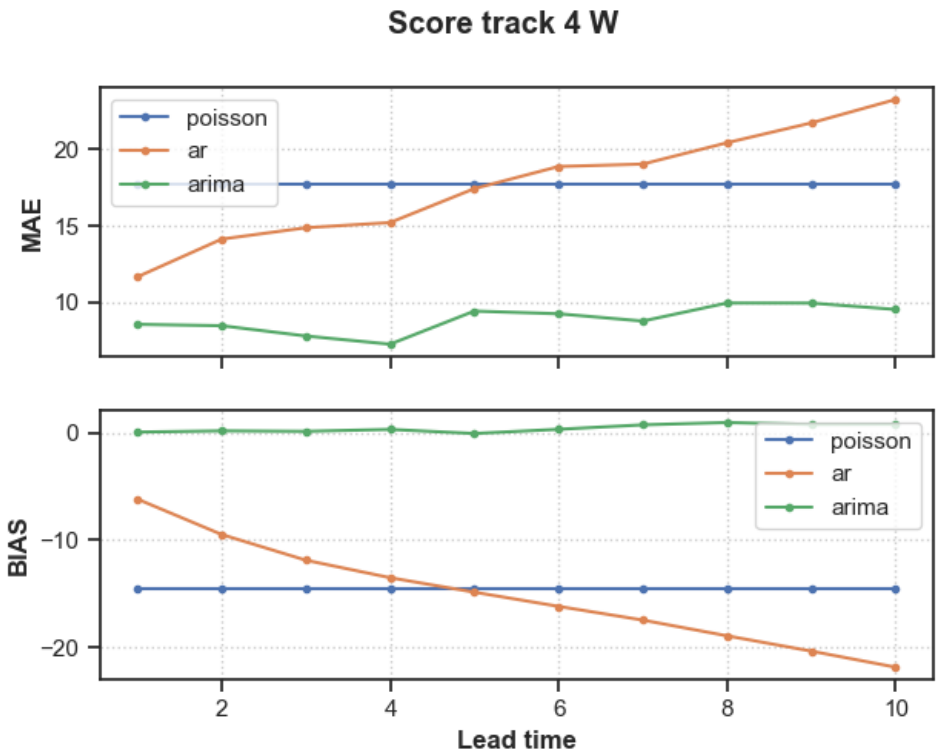
The complete table of the performance can be found in `/output/score_D.xlsx`, `/output/score_W.xlsx` and `/output/score_MS.xlsx` depending on time granularity.

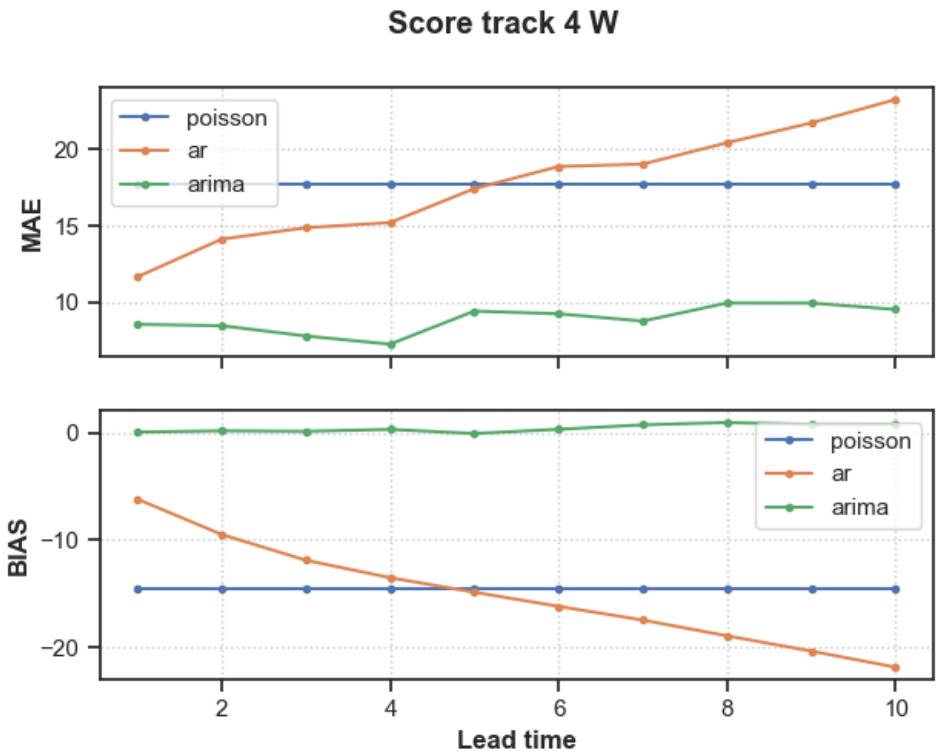
In general autoregressive models perform way better for lower leaptime and granularity ***W*** and ***D***, for example the ***Track 4*** with Weekly and Daily granularity.

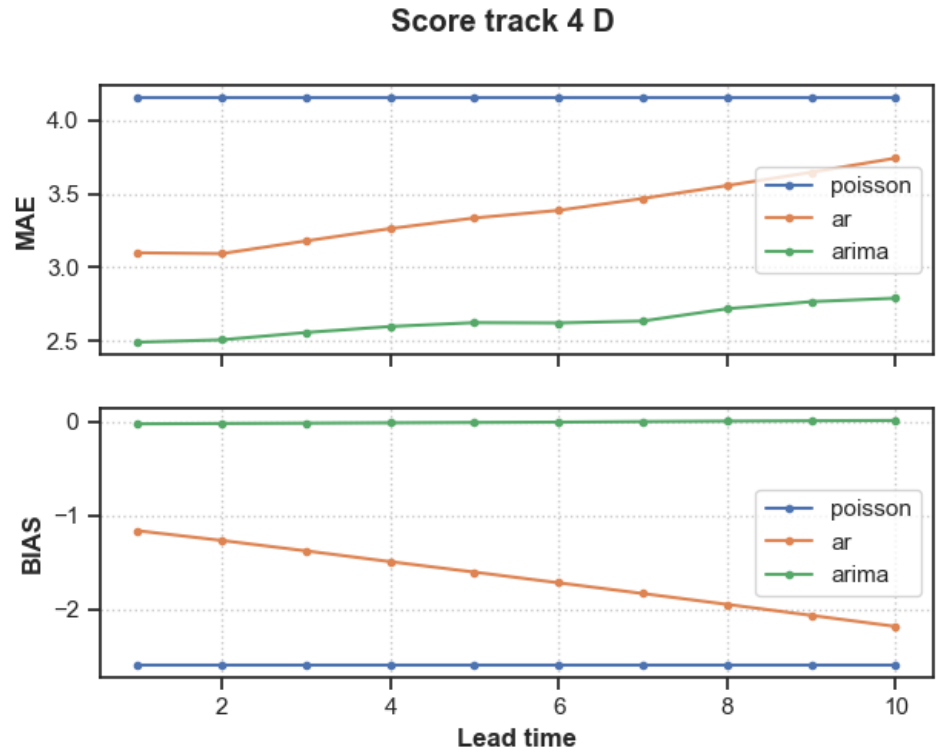
In some cases cases can be identified a breakeven point between autoregressive and calendar dependent model, for example ***Track 2*** with Weekly granularity that for which autoregressive component reach saturation at lag 6.




Score track 3 W
MAE
3.5
3.0
2.5
2.0
poisson
ar
arima
BIAS
1
0
−1
−2
−3
poisson
ar
arima
2
4
6
8
10
Lead time


Score track 4 W
poisson
ar
arima
MAE
20
15
10
BIAS
0
−10
−20
poisson
ar
arima
2
4
6
8
10
Lead time


Score track 4 W
poisson
ar
arima
MAE
20
15
10
BIAS
0
−10
−20
2
4
6
8
10
Lead time

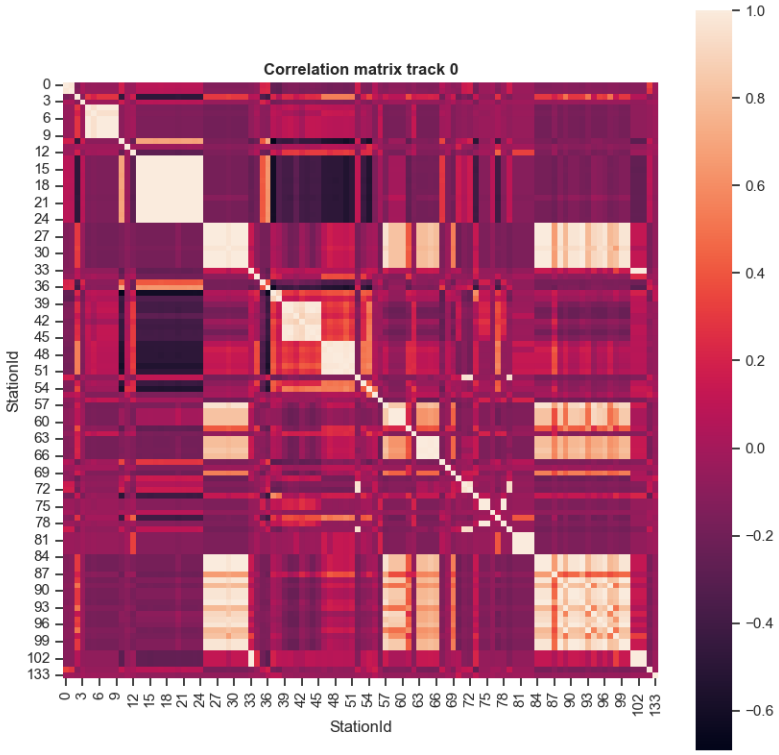

Score track 4 D
poisson
ar
arima
MAE
4.0
3.5
3.0
2.5
BIAS
0
−1
−2
2
4
6
8
10
Lead time

## 4.2 Next steps

In this last section I will list those steps it would be interesting to explore and eventually develop.

### Vectorial models by StationId

For a given track, by looking at the correlation between stations, it is evident there are groups of stations highly correlated. This suggest that vectorial models (such as ***Vectorial AutoRegressive***) could improve the performance and exploit some patterns I've missed when aggregating data.




### Holidays as feature

In the music field it seems reasable take in account also specific periods of the year. Infact one can say that a song played on summer or christmas time worth more. For example in Summer the value is very high because it is also the period when festival and tours are organised so you want your song to be played more in order to increase the hype.

Because of this it would be usefull to assign different weight to certain periods of time, that will affect the optimisation loss function, so that the forecast model will improve the performance in those intervals (most probably this means worse performance during complementary periods of time).

**Improve Bayesian forecasts**

Bayesian models seem promising for this kind of applications mostly because of the distributional output that can provide way more information that a determistic output.

Since the music industry is really time and trend dependent one have to develop a proper model that can change fast.

A solution could be a frequent retrain and assigning greater weights to last observations.