POLITECNICO DI TORINO

Master's Degree in Physics Of Complex Systems

Master's Degree Thesis

Characterization and forecast of online music consumption dynamics

Supervisors

Prof. Vittorio LORETO

Prof. Alfredo BRAUNSTEIN

Candidate

Gianluca BONI

October 2020

Abstract

A data-set of 2251 songs released between 2020/01/15 and 2020/03/24 is analyzed to formulate short and long term predictions of popularity using machine learning techniques. These public data were collected from Spotify, the largest subscription music streaming service with 96 million subscribers and 170 million users overall, and from YouTube, one of the most popular online video-sharing platforms. The first part of the work has a purely predictive character and makes extensive use of machine learning, to solve classification and regression problems. In particular, we detect the features which are the most informative for characterizing the Spotify Popularity, an integer number in a range between 0 and 100 indicating the success of the track inside the streaming-platform. Also, given a track and its related video on YouTube, we build a neural network for inferring the number of views at a given day taking into account all the information available from Spotify. As last, the converse analysis is performed, i.e. the Spotify popularity is predicted taking as input some of the YouTube statistics. Although machine learning models achieve good performances, the brutal use of these algorithms does not allow to go beyond the mere prediction. In the second part of the work, we then investigate the interconnections present between the different time-series composing the data-set. Thus, we quantify the amount of information transferred from the YouTube views time-series to the Spotify popularity time-series, and viceversa. The final result is a directed graph showing the flows of information between all the time-series analyzed. The main achievements, as well as the drawbacks of the models adopted are discussed.

Acknowledgements

Special thanks go to Bernardo Monechi , Giulio Prevedello and Enrico Ubaldi, who with their precious supports helped me to concretize the wonderful opportunity offered to me by prof. Vittorio Loreto - whom I warmly thank - by accepting me at Sony Computer Science of Paris. I also express my thanks to prof. Alfredo Braunstein, who always followed me from Turin, being very helpful in any occurrence. One last thought, as obvious as dutiful, goes to my parents and my sister, tireless patrons of all my initiatives.

Table of Contents

Chapter 1

Summary

In the last decade the transport capacity in the access networks to the Internet has increased substantially, and streaming services are nowadays emerging faster than ever. In 2006 YouTube, the well-known American online video-sharing platform, was bought by Google for US $1.65 billion. In the same year the international media services provider Spotify was founded. Both platforms offer a basic service with advertisements and a premium package via paid subscriptions. One of the pillars on which these streaming services are based is the system of advertising, which values billions of dollars. In this setting, it is crucial to predict whether a multimedia content will become successful, that is a measure for a product to reach as many people as possible. Combining the increasing amount of data available today, with the latest machine learning algorithms, it seems reasonable to try predicting the main statistics associated to YouTube videos, like the number of daily views, likes and dislikes.
The goal of this study is to understand to what extent such predictive task is achievable, especially for the evolution of the Spotify Popularity - an integer index varying in time ranging from 0 to 100 - and the number of daily views on YouTube. Moreover, the use of machine learning techniques will enable the determination of the most informative features for the binary classification *popular* vs. *unpopular*. To this end, we first build a Random Forest model, able to classify if a track will be more popular than the average. Secondly, we train Multilayer Perceptrons to perform regressions and to understand if the info from Spotify are sufficient to make predictions about the views on YouTube, and viceversa. Thirdly, we combine all the information from the two streaming platforms to give long term predictions. As we will show, even if it is possible to build models able to furnish some predictions about the evolution of time-series describing the success of the music tracks on Spotify and YouTube, it is not immediate to understand which are the drives and the interdependencies between them. Therefore, in the second half of the work we introduce the Transfer Entropy analysis and we apply it to build a directed graph,

showing the flows of information between all the time-series present in the data-set.

The thesis is structured in six main chapters:

- **Chapter 2**: In this chapter we introduce the main music track data-sets we use. Then, we describe all the features and information which are associated to each track available on the Spotify platform. Finally, the data are explored to provide insight on the main observables of interest.
- **Chapter 3**: We present a short and essential review of the state of art of the machine-learning algorithms used in this work. We focus on the mathematical definition of the algorithms, thus showing the metrics and the methods adopted to quantify the quality of the classifications and regressions performed.
- **Chapter 4**: A short introduction of the main elements of Information Theory is here provided in order to smoothly introduce Granger Causality and Transfer Entropy. Here the main goals are to understand the analogies and the differences between these concepts, and to understand how we apply them to quantify the amount of information *transferred* between YouTube and Spotify time-series.
- **Chapter 5**: In this chapter we compare Transfer Entropy and Granger Causality on synthetic time-series already analyzed in the literature. We aim at testing the code developed and getting acquainted with the results provided from the causality inference techniques.
- **Chapter 6 and 7**: In these final chapters we apply all the tools developed to the real time-series. The final goals are (1) to produce a heat-map and the related graph showing the interdependencies between the time-series of interest and (2) to understand which time-series have the strongest driving influence on the others. All the results are commented and a final discussion concludes the work.

Chapter 2

Introduction to the data

The Internet's capacity to reach millions of people around the world has brought artists to take part in the streaming system. Today Spotify counts tens of millions of music tracks, making the Swedish platform one of the most competitive reality in its field[1]. Only a small minority of artists prefers to not be present in the Spotify catalog, even if complaints about low revenues have not been absent in the last years [2]. In fact, unlike physical or download sales, which pay artists a fixed price per song or album sold, Spotify pays royalties based on the number of artist streams as a proportion of total songs streamed. It distributes approximately 70% of its total revenue[3] to rights holders, who then pay artists based on their individual agreements. Despite of the criticism moved by who argues that a fairly renegotiation for the license deals is needed, the service offers to the paying users a complete access to all the musical production on the platform. Users have also the access to *Playlist*, i.e. a collection of songs of different artists. Playlists are continuously updated by curators, and the success of tracks is also determined by its presence (or absence) in the most popular playlists. There are teams all over the world working on compiling Spotify playlists, using extremely sophisticated information - e.g. how frequent a track is skipped, how many times a track is listened to in its entirety, if it is added to the personal playlists of users, etc. - in order to decide which songs must be replaced or added [4],[5].
Other parameters of interest to determine the global success of a track, and so the economical business in the music industry, are the statistics related to the YouTube platform. In the last decade we have assisted to a constant increase in the amount of resource invested in the production of musical video, and streaming numbers have became crucial to the pop music ecosystem as much as festival ticket sales [6].
In our analysis we have selected 134 popular playlists and we have built different data-sets containing several info from 2251 songs that have appeared between the 15th January 2020 and the 24th March of the same year. The first data-set built contains the following *static features*, which do not vary in time:

- **name**: The name of the track;
- **album_name**: The name of the album the track belongs to;
- **artist_name**: The name of the artists/band associated to the track;
- **disc_number**: An integer number specifying the number of discs the album is composed of [int];
- **duration_ms**: The duration of the track in milliseconds [int];
- **explicit**: A Boolean variable (True/False) indicating the presence of explicit contents in the lyrics of the track;
- **track_number**: The number of the tracks inside the album [int];
- **n_available_markets**: The number of markets in which the track is available [int];
- **album_total_tracks**: The album total tracks [int];
- **n_artists**: The number of artists involved in the track [int];
- **danceability**: It describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable [float];
- **energy**: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy [float];
- **key**: The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation . E.g. 0 = C, 2 = D, and so on. If no key was detected, the value is -1 [int];
- **loudness**: The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 dB [float];

- **speechiness**: It detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks [float];

- **acousticness**: A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic [float];

- **instrumentalness**: Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0 [float];

- **liveness**: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live [float];

- **valence**: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry) [float];

- **tempo**: The beats per minuts (BPM) of the track [int];

- **time_signature**: An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure) [int].

In addition to these static features, a *popularity index* is associated to each track and artist. It ranges between 0 and 100 and it varies in time. The algorithm used by the streaming platform to compute the popularity indices is not public, but we suppose to be strictly linked to the number of plays. Dedicated data-sets have been created to daily store the the popularity indices, and to keep track of the number of *followers* of the playlists and artists as they appear in the selected 134 playlists. Finally, four more data-sets have been included to the analysis to keep track of the numbers related to YouTube videos. For each track the YouTube statistics of the associated video clip associated are recorded, by monitoring the number of *views*, *likes*, *dislikes* and *comments* day by day. All the metadata, both from

Spotify and YouTube, have been accessed through specific API web services, tools that allow an automatic access to data. On the other hand, a sophisticated *ad hoc* routine was realized to associate the correct YouTube video to a given track. After searching the name of the track on the YouTube API tools, the most clicked video is chosen to be the representative one. Our basic assumption is that users prefer to watch the official video of a musical track rather than amateur remakes made by other users, such as videos with the lyrics. In Figure 2.1, we plot the distribution of the static features assigned to the tracks composing the data-set in order to have a first insight about the subject of investigation. We clearly see that the data-set is mostly composed by tracks characterized by high values of danceability, loudness and energy, and low values of speechiness, acousticness, instrumentalness and liveness. Concerning the so called dynamic features, we first remark that some tracks from the data-set present a delay between the Spotify release day and the day in which YouTube statics are available. This happens every time that a track is added to one of the 134 playlist of interest some days after its release on the market. Generally speaking, Figure 2.2 shows that for the majority of the tracks, YouTube statistics are available at most 3 days after the publication. In Figure 2.4 we plot in log scale the YouTube views time series of the first class, the one composed by the tracks whose statistics on YouTube are recorded with no delay. We realize that not all the time series have the same length and that this class is heterogeneous: we can find very popular track but also videos with few views after several days from the release.

As a final remark,, the typical behaviour of the Spotify Popularity index is highlighted in Figure 2.3. The main characteristic of this index is that normally it reaches a *plateau* which can define the long term popularity of the track.

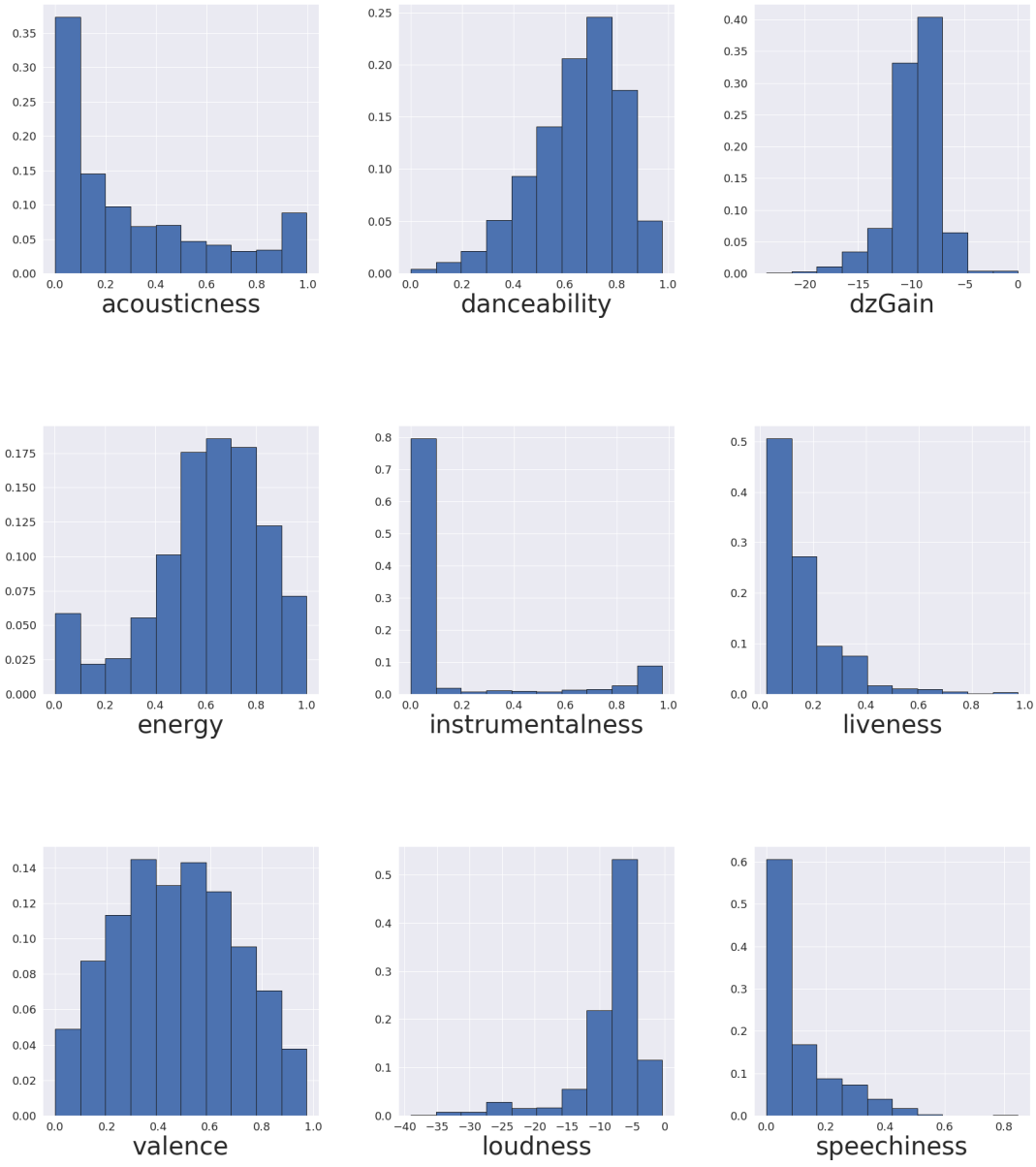

Figure 2.1: Distributions of the static features.

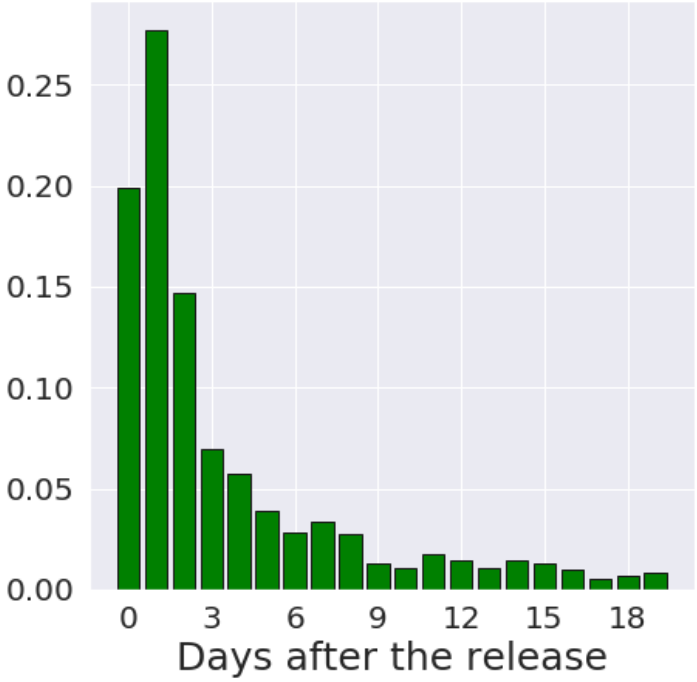

Figure 2.2: Distribution of the days of delay in recording the statistics from YouTube. The day zero corresponds to the release day of the tracks.

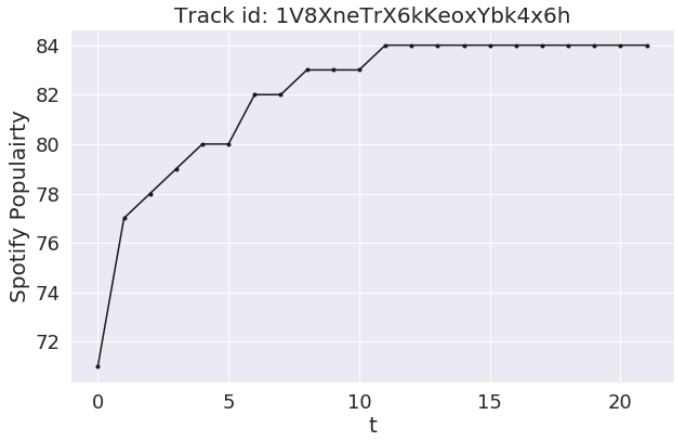

Figure 2.3: Spotify Popularity time series for the track '1V8XneTrX6kKeoxYbk4x6h'. The index reaches a plateau around 84 point of popularity after 10 days.

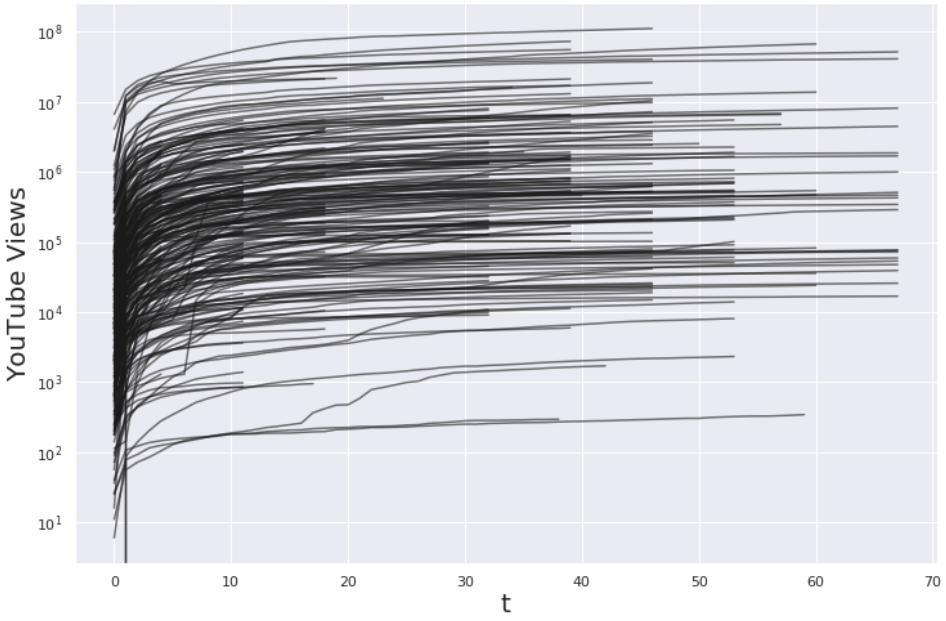

Figure 2.4: Cumulative Views for the tracks whose statistics are recorded from the release day. The resolution of the time is daily, which means t takes only integer values. $t = 0$ stands for the day of release which is not the same for all the tracks.

Chapter 3

Success prediction with Random Forest and Neural Networks

In this chapter we discuss the mathematical formulations of the algorithms used in chapter 4, the Random Forest Classifier (RF) and the Multilayer Perceptron (MP). We will use the following notation:

- D: a data-set;
- L: the size of the data-set, i.e. the number of data-points forming the data-set;
- F: the set of features attached to any points;
- x_i : the i-th data-point inside the data-set ($i <= L$);
- f_j^i : the j-th features of the i-th data-point;
- t_i: the target, i.e. the value to predict. If t_i can assume only integer values, a classification task is performed, otherwise a regression.

3.1 Random Forest

First introduced in 1995, the Random Forest is a feasible supervised algorithm, which can perform both classification and regression tasks [7]. As the name suggests, a Random Forest classifier is composed of an ensemble of Random Trees models. To each node of such decisional structure a feature $f \in F$ and a binary question q are attached. Answering to a node-question has as effect the splitting of the subset of D involved in that node in two groups, whose elements in general belong to

different classes. If this is the case, the algorithm proceeds appending two more nodes (one for every splitting) with the respective binary questions. The algorithm stops as soon as a stopping condition is met, e.g. *max_ depth*, *max_nodes*, all nodes are pure etc. The process of adding nodes can be visualized as a progressive adding of linear classifications of a multi dimensional space, whose dimension corresponds to the number of features. Once a tree is built, it is possible to classify data-points of the train-set, just answering the stream of questions across the tree. It is evident that it is needed a formal definition how to best split data. In order to quantify the *impurity* Υ_j of the node j and to handle the splitting procedure, two kind of measures are commonly used:

- **Gini Criterium**: each node is associated with the so called Gini Impurity, that is the measure of how often a randomly chosen element from the set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset. Let K be the number of the classes and $p_{T,j}$ the fraction of elements belonging to the class j at the node $T \subseteq D$, the Gini impurity of the node T is then given by:

$$G(T) = \sum_{i=1}^{K} p_{T,i} \sum_{j \neq i} p_{T,j} = \sum_{i=1}^{K} p_{T,i}(1 - p_{T,i}) = 1 - \sum_{i=1}^{K} p_{T,i}^2 \tag{3.1}$$

- **Entropy**: let $T \subseteq D$ the subset of D forming a given node. The impurity of such a node can be quantified as the entropy $H(T)$ (see Chapter 4 for more details about entropy in the context of Information Theory).

Given a feature f we indicate with q_f one of the possible binary question associated to f, and with $a(x; q_f)$ the Boolean variable coding the *answer* of the data-point x to q_f. Let $S_{\mathrm{T}}^{q_f} = \{x \in T \subseteq D | a(x; q_f) = \mathrm{True}\}$ and $S_{\mathrm{F}}^{q_f} = \{x \in T \subseteq D | a(x; q_f) = \mathrm{False}\}$ be the partitioning of $T \subseteq D$ generated by q_f , the Information Gain associated to f is given by:

$$IG(f; T) = \Upsilon(T) - \max_{q_f} \left\{ \frac{|S_{\mathrm{T}}^{q_f}|}{|T|} \Upsilon(S_{\mathrm{T}}^{q_f}) + \frac{|S_{\mathrm{F}}^{q_f}|}{|T|} \Upsilon(S_{\mathrm{F}}^{q_f}) \right\} \tag{3.2}$$

regardless of the choice made for computing Υ. Thus, at each step the feature f^* that maximizes the Information Gain is chosen, i.e. $f^* = \mathrm{argmax}\, IG(f; T)$. Although this learning model is flexible and it is not *biased*, it's affected by high *variance*. Indeed, the random tree decision model tends to *overfit* the data-set since its great flexibility, resulting a high variance of the learning parameters as the training data vary importantly. In order to cure this major drawback, in the Random Forest scheme each tree is trained with different subsets of D and F, or in other words, each trees is *susceptible* to different groups of features. This

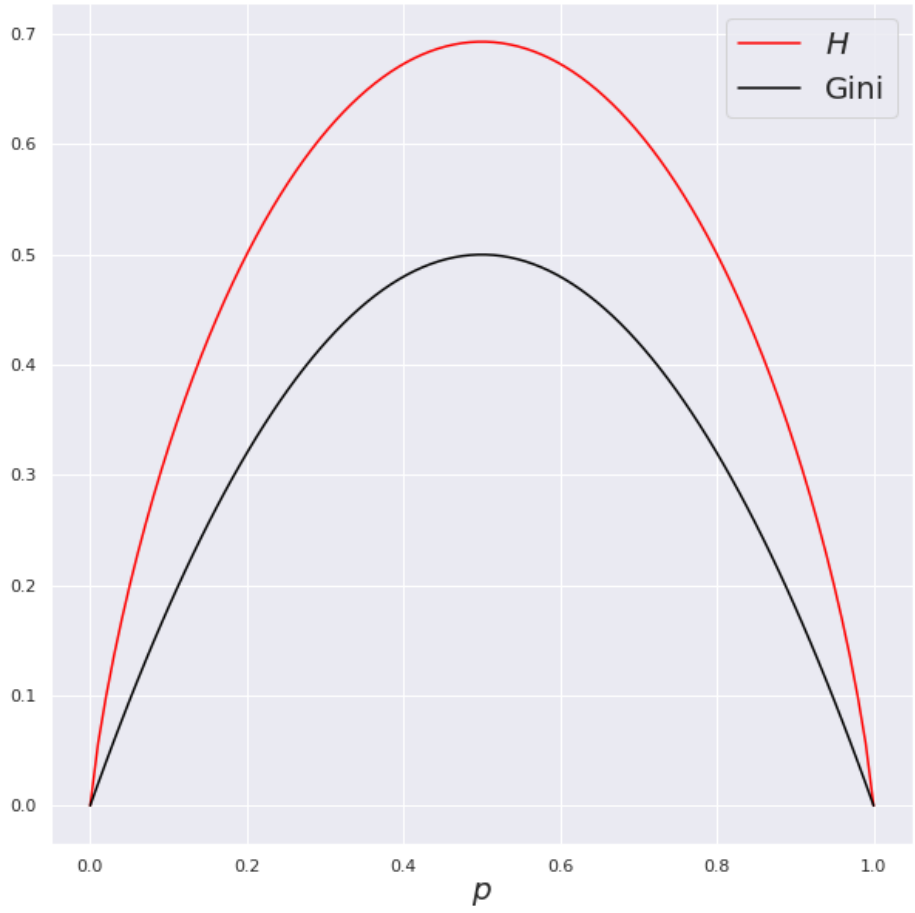

Figure 3.1: Comparison between the Gini Index and the Entropy as measures of *impurity* for $K = 2$. On the x-axes p, the fraction of elements belonging to the first class. As expected, for both the the measures the impurity is maximum for $p = 0.5$.

kind of approach to the training, known as *bootstrap aggregating* (or bagging), is aimed to improve the stability and accuracy, reducing variance and overfitting. Once the training is complete, each prediction is the result of a pool among the possible different classifications returned as output from the tree models forming the forest. The output of the classifier is a vector $p_{output}(c; x)$ providing a probabilistic

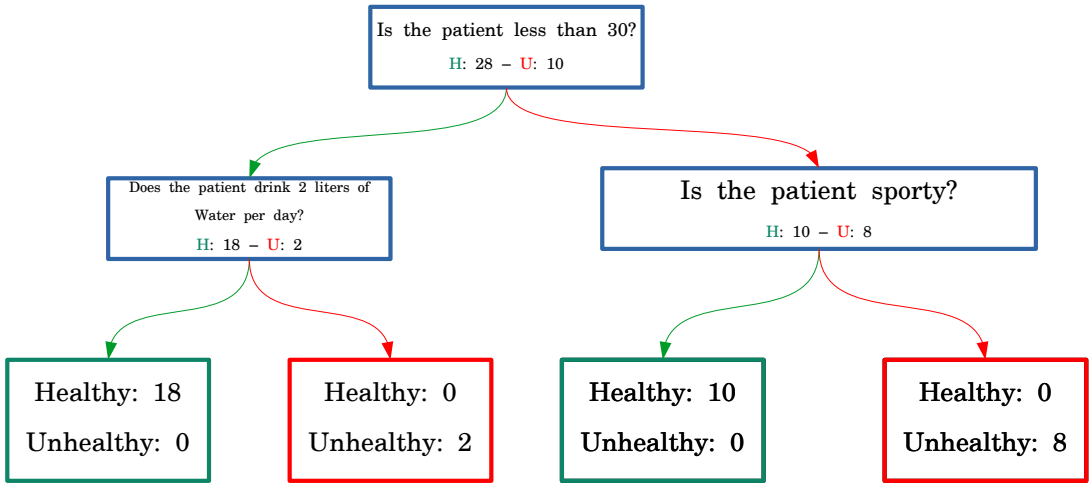

Figure 3.2: An example of binary classification based on a Random Tree classifier.

information about the classification, e.g. $p_{output}(c_1; x) = Prob(x \in c_1)$. To test the quality of the classification, it is common usage to build the so called *Confusion Matrix*. While an ideal machine learning algorithm is able to properly classify all the elements in the testing set, in practice the classification on the test-set will produce *false positive*(FP) and *false negative*(FN), as well as the *true positive*(TP) and the *true negative*(TN). In the case of a binary classification, the confusion matrix provides this information in a 2×2 table. In addition to the confusion matrix the quality of a binary classification can be quantified using the *Receiver Operating Characteristic* curve, also known as ROC curve. A classification can be characterized by the following parameters:

- TPR = *True Positive Rate/* = $\frac{TP}{TP+FN}$
- TNR = *True Negative Rate/Specificity* = $\frac{TN}{TN+FP}$
- FPR = *False Positive Rate* = $1 - \text{TNR} = \frac{FP}{TN+FP}$
- FNR = *False Negative Rate* = $1 - \text{TPR} = \frac{FN}{TN+FP}$

An ideal model would provide a classification with $TPR = 1$ and $FPR = 0$; 100% of the actual positives are classified as such and no one among the negatives are classified as positive. Therefore, the ROC curve is created by plotting the value of the TPR compared to the FPR at various *threshold settings*, i.e. the threshold in

probability for which an element is classified to belong to the positive class. Once the ROC curve is plotted, it's possible to compute the *Area under the ROC*, the so called AUROC. If AUROC > 0.5 the model is able to perform a classification more efficiently than random assignments of the items into the two classes.

3.2 Multilayer Perceptron

A Multilayer Perceptron (MLP) is a mathematical architecture composed of nodes, also referred as *neurons*, displayed into at least three layers - the input layers, one *hidden layer* and *the output layer* [8]. Nodes belonging to a layer communicate to the neurons of the immediate nearest layers through direct connections, the *synapses*. To each synapse is associated an *activation weight*, which stands for the strength of the connection: we shall define $w^l_{i,j}$ the weight associated to the synapse connecting the the node i in the layer l to the node j of the layer $l+1$, and the matrix W^l_i the matrix storing all the weights between the node i of the layer l to the nodes of the layer $l+1$. Once the set of weights is specified, the activation of a neuron is controlled by a non-linear *activation function*, $g : \mathbb{R} \to \mathbb{R}$. The whole model can be mathematically formulated as function $\mathcal{M} : \mathbb{R}^d \to \mathbb{R}^o$, with d being the dimension of the input and o the dimension of the output:

$$\mathcal{M}(\vec{x}) = s(\vec{b_h} + \mathbf{W}_h \cdot \vec{g}(...(\vec{b_2} + \mathbf{W}_2 \cdot \vec{g}(\vec{b_1} + \mathbf{W}_1 \cdot \vec{x})))), \tag{3.3}$$

where h is the number of hidden layers, $\vec{g}$ the natural extension to vector of the scalar-to-scalar activation function g, $\vec{b_i}$ the *bias* associated to the i-th layer and s the activation function that generates the final output of the network. Typical choices for g include the *hyperbolic tangent*, the *logistic sigmoid function* and the *Rectified Linear activation function*. In the case of multi-class classification, class-membership probabilities can be obtained by choosing s as the *softmax function*. As the RF model, the MLP can be trained to perform both regression and classification task. In both case it's needed to define a *loss function*, whose optimization provides the learning of the parameter of the model $\theta = \{\{W\}, \{b\}\}$. A typical choice for the loss function is the *squared error* function:

$$\mathcal{L} = \sum_i^N (\hat{y}_i - y_i)^2 \tag{3.4}$$

with $\hat{y}_i$ and y_i indicating respectively the output of the model to the input x_i and the actual response of the system to the latter. To perform the learning is commonly perform a *gradient descent* method, relying on a *back-propagation* scheme. Thus, the update of the weights is given by

$$w_{ji}(n+1) = w_{ji}(n) - \eta \frac{\partial \mathcal{L}(n)}{\partial w_{ji}(n)} \tag{3.5}$$

ReL	$g(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$
Sigmoid	$g(x) = \dfrac{1}{1+e^{-x}}$
tanh	$g(x) = \tanh(x) = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
Softmax	$g_i(\vec{x}) = \dfrac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$ with i indicating the class index

Table 3.1: Examples of activation functions.

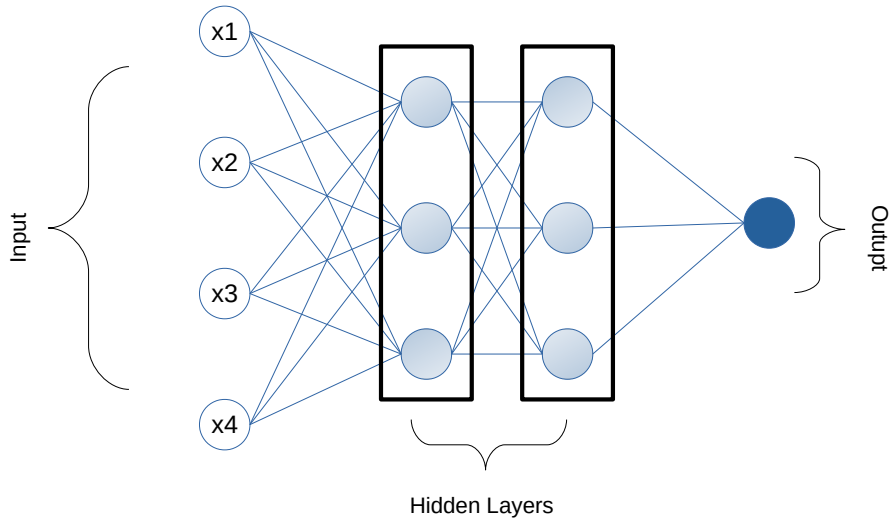

Figure 3.3: MP with 2 hidden layers and 3 neurons each. The input is a 4-dimensional data-point and the output is a single-value number. The blue lines represent the synapses. Bias are omitted.

where the integer n indicates the step of the iteration. While for the weights associated to the output layers a derivation is straightforward, the analysis is more difficult for the change in weights to a hidden node (see Appendix A). For a given Training-Set, back-propagation learning may thus proceed in one of the two basic ways:

- *Sequential Mode*: it's also referred to as *on-line*, *pattern*, or *stochastic mode*.

In this mode of operation weight updating is performed after the presentation of each training example;

- *Batch Mode*: weight updating is performed after the presentation of all the training examples that constitute an epoch. Here, for each epoch we define the loss function as the average of (3.4) over all the examples.

In practical implementation *stochastic gradient descent* algorithms are preferred to improve the efficiency of the learning.
To avoid overfitting it's quite common to add a regularization term to (3.4). In the case of a L_2 regularization, the loss function becomes:

$$\mathcal{L} = \sum_{i}^{N} (\hat{y}_i - y_i)^2 + \lambda \sum_{i} w_i^2 \tag{3.6}$$

where λ is known as the *penalty parameter*. If we think to the loss function as an *energy* function we aim to minimize, the regularization term can be viewed as a constraint which controls the excessively fluctuating function such that the coefficients don't take extreme values, thus reducing the variance of the model, without a substantial increase of its bias. In this framework, in order to quantify the quality of a regression, the *mean absolute error* (MAE), the *mean square error* (MSQ) and the *coefficiente of determination* (R^2) are introduced:

$$\begin{aligned} \text{MAE}(\{y\}, \{\hat{y}\}) &= \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \\ \text{MSE}(\{y\}, \{\hat{y}\}) &= \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ R^2(\{y\}, \{\hat{y}\}) &= 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \end{aligned} \tag{3.7}$$

The coefficient of determination is a statistical measurement that describes the proportion between the variability of the data and the correctness of the statistical model used: given two models with the same square error on different data-sets, this coefficient penalizes the model which is implemented on the data-set affected by the smallest variance. Conversely, given two models trained on the same data-set, the R^2 indicates as the best model the one affected by the smallest square error on the predictions. Therefore, it provides an indication of how well unseen samples are likely to be predicted by the model, through the proportion of explained variance. As such variance is data-set dependent, R^2 may not be meaningfully comparable across different data-sets. Best possible score is 1, but it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.

3.3 Implementation

The models introduced are implemented on a *Jupyter notebook*, making extensive use of *Scikit-learn*[9], a free software machine learning library for the Python programming language. Largely written in Python, it uses the *numpy* library extensively for high-performance linear algebra and array operations. Scikit-learn provides intuitive and user friendly implementations of machine learning model. In few lines of code it's possible to build a RF importing the class *RandomForestRegressor*. The most important tunable parameters are[1]:

- **n_estimators**: the number of trees in the forest;
- **criterion**: the function to measure the quality of a split;
- **max_depth**: the maximum depth of the tree. min_samples_split samples;
- **min_samples_split**: the minimum number of samples required to split an internal node;
- **min_samples_leaf**: the minimum number of samples required to be at a leaf node;
- **max_features**: the number of features to consider when looking for the best split;

From the same library we import also the class *MLPClassifier* to implement the MP. In this case the most important parameters to be set are:

- **hidden_layer_sizes**: tuple indicating the number of the neurons in each hidden layers;
- **activation**: activation function for the hidden layer;
- **alpha**: L2 penalty (regularization term) parameter;
- **learning_rate{'constant', 'invscaling', 'adaptive'}, default='constant'**: learning rate schedule for weight updates;
- **max_iter**: maximum number of iterations. The solver iterates until convergence or this number of iterations.

[1]Look at https://scikit-learn.org/stable/ for a complete documentation.

3.4 Random Forest for trend predictions.

We have seen in the previous chapter some typical trends of the popularity index of Spotify. In Figure 3.4 we plot how the Spotify Popularity index related to 800 different tracks of the data-set (black lines) varies in time[2] and the average trend (red line). The question is: "is it possible to build a model able to predict if a track will have a popularity above or below the mean trend, i.e. if its associated black line will be above the red one ?". We know that a Random Forest algorithm can be trained to perform this task. First, we assign to each track of the data-set a binary variable $\phi = 0,1$: if during the first week the track has a popularity greater than the average value then $\phi = 1$, otherwise $\phi = 0$. If during the first week the index results to be above the red line for some days and below for the remaining days, the variable ϕ is assigned just on a majority criterion. We then split the data-set into training and test set in a such way that the training set is made of 70% of the tracks in the whole data-set. After performing a Grid Search, which is simply an

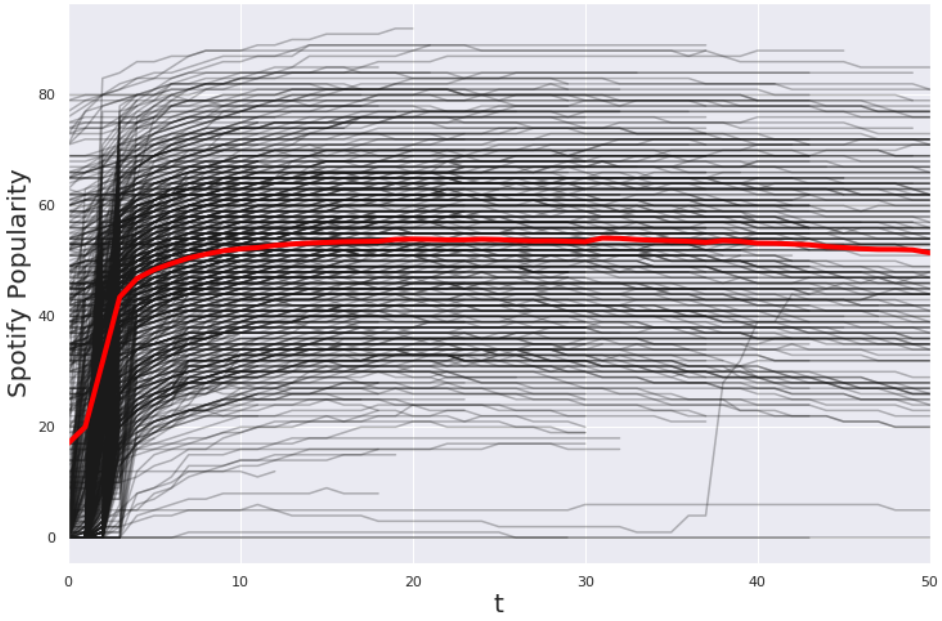

Figure 3.4: Spotify Popularity time series of 800 tracks of the data-set, in red the average dynamic.

[2] $t = 0$ corresponds to the day of first appearing in one of the 134 playlists.

exhaustive searching through a manually specified subset of the hyperparameter space, we tune 'criterion' = 'gini', 'max_features'= 'auto' and 'n_estimators'= 60. We found that RF performs a good classification with this choice of the parameters[3]. In particular, in Figure 3.5 and 3.6 we report respectively the ROC curve and the Confusion Matrix built on the test-set with a threshold of 0.5. We obtain an AUROC = 0.89 on the test set and in Table 3.2 we resume the values of the indices before introduced for evaluating classification:

TPR	$\frac{224}{224+53} = 0.80$
TNR	$\frac{272}{272+59} = 0.82$
FPR	$\frac{59}{272+59} = 0.18$
FNR	$\frac{53}{224+53} = 0.20$

Table 3.2: TPR, TNR, FPR and FNR on the test set.

Finally but not less interestingly, we show the ranking of the 10 most significant features. Indeed, to each features f is possible to associate a score ρ_f which is updated every time that f is used in any node of any tree of the Random Forest. As much the Gini impurity is reduced after the splitting generated by posing a binary question about f, as ρ_f increases. At the end of the training all the ρ_i are normalized to one and a ranking list can be produced. In Table 3.3 we report what found for the classification task performed. Not surprisingly we find that the most informative feature is the artist popularity at the day zero.

[3]The default values are used for the other parameters.

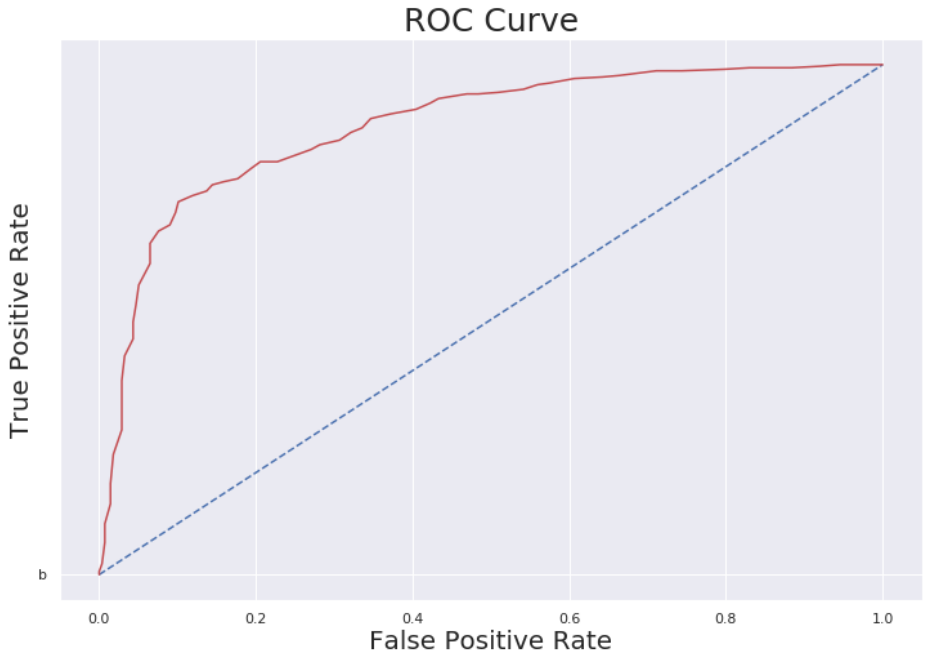

Figure 3.5: In red the ROC curve, in dashed blue line the ROC curve of the base model performing a completely random classification.

	f	ρ_f
1	artist_popularity	0.3123
2	loudness	0.0767
3	energy	0.0488
4	acousticness	0.0481
5	duration_ms	0.0438
6	instrumentalness	0.0422
7	dzGain[4]	0.0422
8	speechiness	0.0405
9	tempo	0.0377
10	danceability	0.0355

Table 3.3: Features' ranking list.

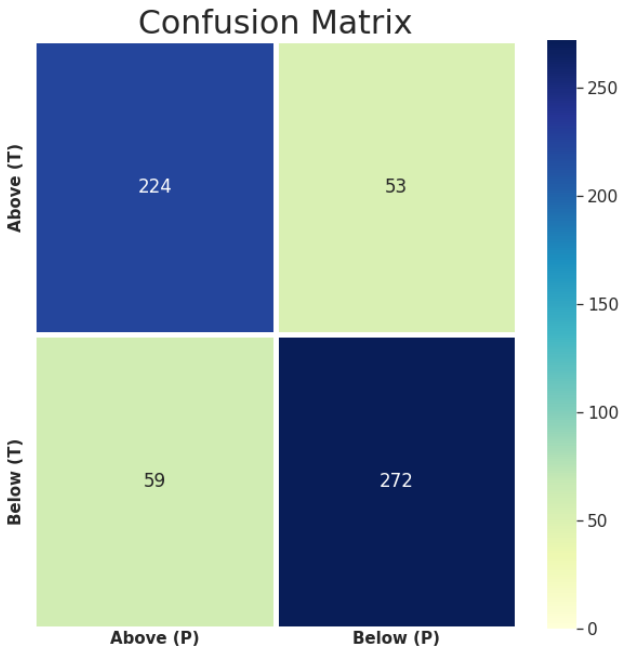

Figure 3.6: Confusion Matrix on the test-set. The uppercase letters T and P stand respectively for True and Predicted so that Above(T)-Above(P) is the cell for the tracks which are predicted to have a popularity above the mean trend and indeed it's true. The threshold is set to 0.5, which means that there is no bias in the assignment of the data-points into the two classes.

3.5 Regressions with multilayer perceptron

The good results provided by the Random Forest model classifier suggest that a further refinement of success prediction is achievable. Therefore, we aim at implementing a machine learning model able to furnish daily predictions about the popularity of tracks on Spotify and the number of views on YouTube when the same kind of information are known just for the first days after the release date. First we build a MP trained to predict the Spotify popularity at day 14 after the release taking as input the statistics from YouTube for the first 7 days. In order to do that, we first prepare a data-set $\mathbb{D}$ including only the tracks whose statics are recorded since their release on the market. After, for each track we associate as input variables the number of daily Views(VW), Likes(LK), Dislikes(DLK), Comments(CMM) as well as the cumulative values. We also introduce as inputs the *likeness* defined as $\frac{LK}{LK+DLK}$ and the *response*, $\frac{CMM}{VW}$. The architecture of the multilayer involves three layers, each of them composed of 250 neurons, and we tune max_iter = 1000, n_iter_no_change= 50.

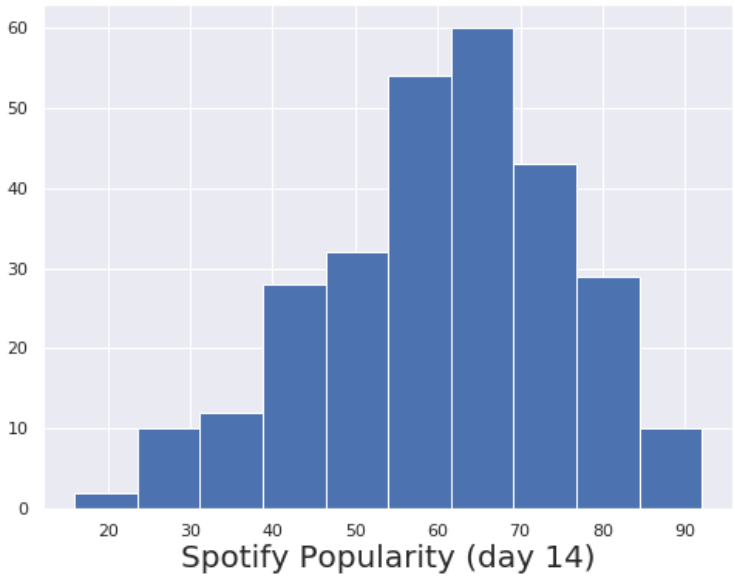

Figure 3.7: Distribution of the Spotify Popularity at day 14 after the release.

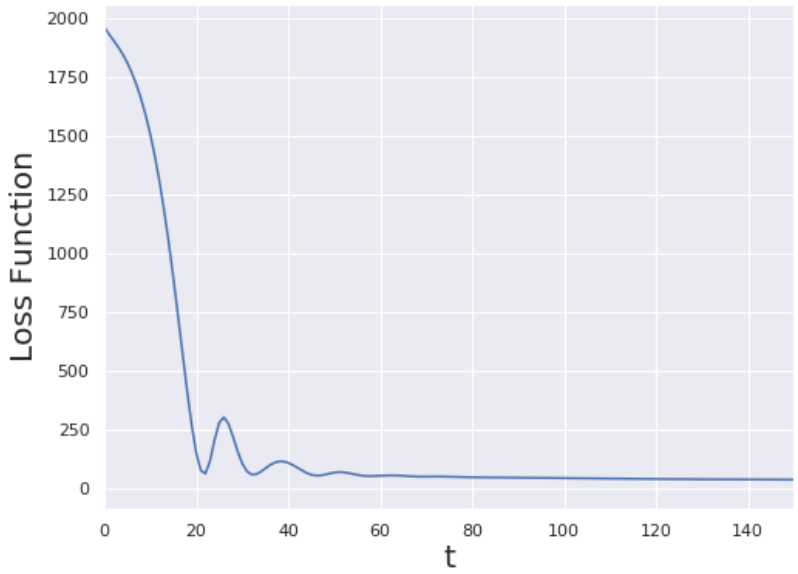

Figure 3.8: Convergence of the Loss Function. The time t is an integer number standing for the iterations of the weight-updating.

In Figure 3.7 we show the distribution of the Spotify Popularity index at day 14 after the release of the track forming $\mathbb{D}$; we clearly see that the sample is heterogeneous in this sense, i.e. the Spotify Popularity values range approximately from 20 to 90, covering almost the whole spectrum of possible values. Predicting the popularity at day 14 means to furnish a prediction of the *plateau*, i.e. having a guess for the long term success of a given track. How the loss function converges is shown in Figure 3.8, while the general performance of the model is visually illustrated by means of a scatter plot in Figure 3.9. In this plot the x-coordinate of a point represents the actual value of the Spotify Popularity at day 14, while the y-coordinate the prediction provided by the MP. We distinguish the tracks constituting the training set with the ones in the test using two different colors, and we plot the bisector in red: the better the performance of the model, the closer the points will be to the bisector. Quantitatively we evaluate the model using three metrics: MAE, MSQ and R2. On the test-set we find a $R^2 = 0.64$ and a $MAE \approx 6$, meaning that on average the prediction is affected by an error of 6 points on a value that can range between 0 and 100.

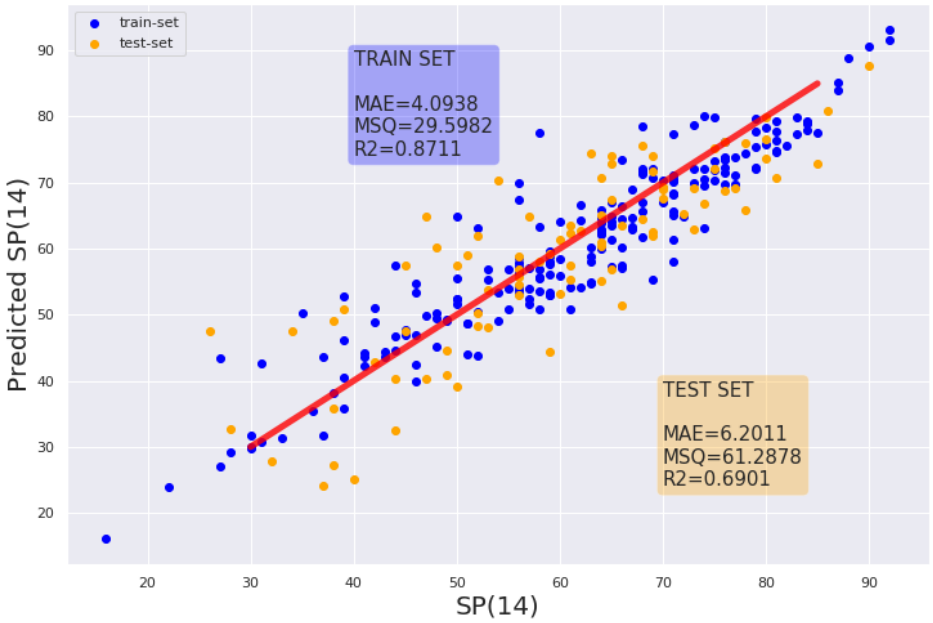

Figure 3.9: Scatter plot between the actual value of the Spotify Popularity index at day 14 (x axis) and its prediction (y axis). In blue the model on the train-set, in orange on the test-set. The red line is the curve $y = x$.

The good results obtained by means of this model push us to implement the same model but in a reverse fashion: predicting the number of views at day 14 taking as inputs all the features provided by Spotify for the first 7 days. In particular we pass as input the popularity of the track, the popularity of the artist, the number of the followers and the following static features: 'disc_number', 'duration_ms', 'track_number', 'n_available_markets', 'album_total_tracks', 'n_artists', 'danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo'. The target variable is the log of the number of views, since in a such way the algorithm doesn't tend to fit better only the tracks with a large number of views. In Figure 3.10 we present the distribution of the log(VW) at day 14 after the release and in Figure 3.11 the convergence of the Loss Function.

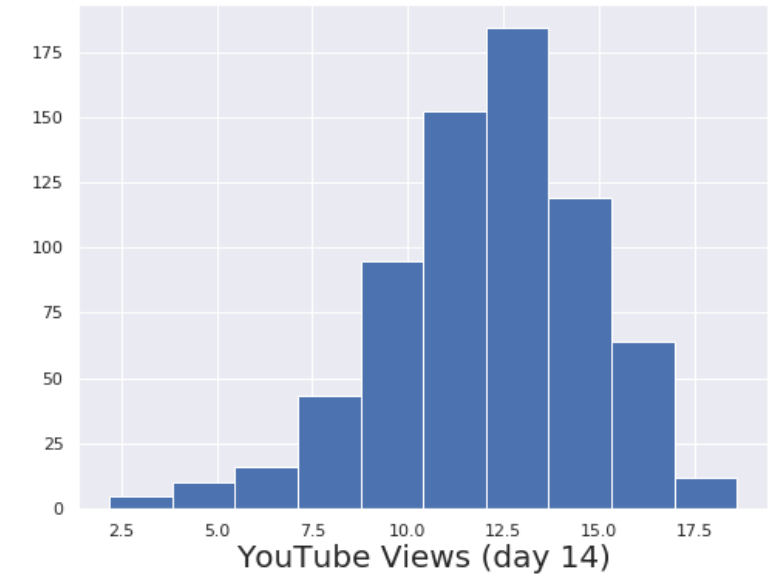

Figure 3.10: Distribution of the log(VW) at day 14 after the release.

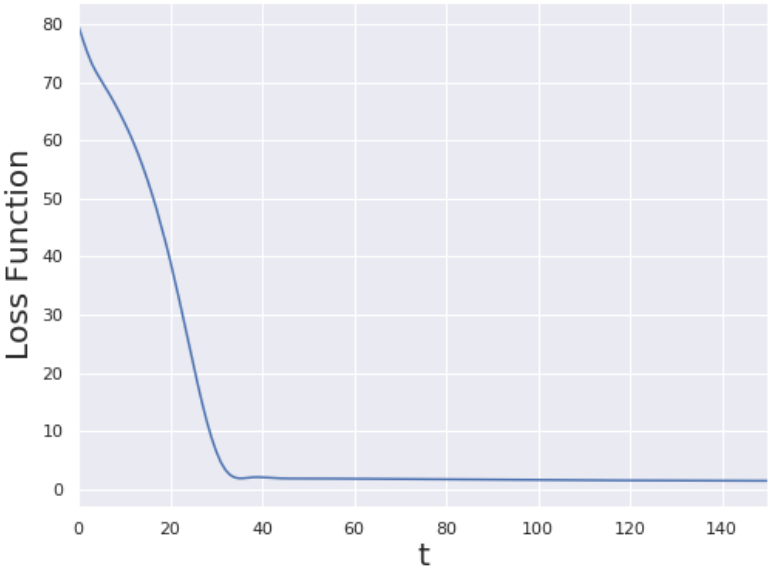

Figure 3.11: Convergence of the Loss Function. The time t is an integer number standing for the number of iteration.

Like in the previous case, the goodness of the model is evaluate graphically by the scatter plot in Figure 3.12 and quantitatively by the usual metrics. In this case we find on the test set a $R^2 = 0.60$ and a $MAE \approx 1.4$ To conclude this section we

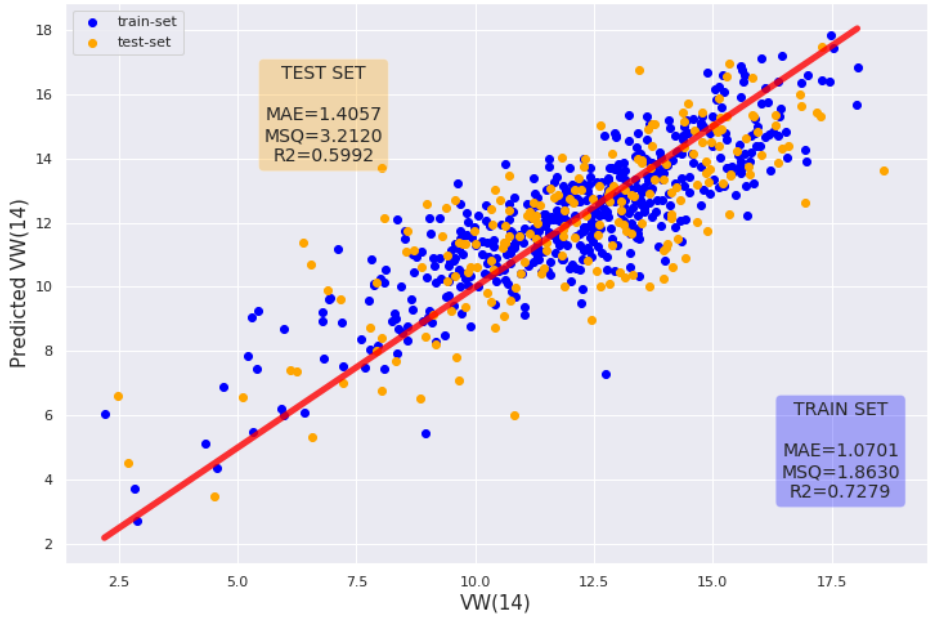

Figure 3.12: Scatter plot between the actual number of the YouTube views at day 14 (x axis) and its prediction (y axis). In blue the model on the train-set, in orange on the test set. The red line is the curve $y = x$.

propose a model for predicting entire time series starting from the first days of recording. The implementation is straightforward: we gather all the information available and for each day and for each time series a MP is trained to predict the value of the next day. The latter is then used as starting input, together with the whole time series until that day, and the regression goes on. The first 6 days of statistic are given as input to the model. Since the error of a prediction is affected by all the errors generated through all the previous predictions, this kind of model is not robust, and so the goodness of the predicted time series strongly depends on the time series itself. As an example, in Figure 3.13 and 3.14 three regressions of this type are shown: the YouTube Views time series, the Spotify Popularity and the Artist Popularity. For the track 6nxOW44RKSH3OBw6ZP8yuZ the accordance between the actual time series and the predicted ones is quite good, while for 7iZGc02ejD2XfExCveq5I4 it's evident the mismatching between the predicted

YouTube time series and the recorded one.

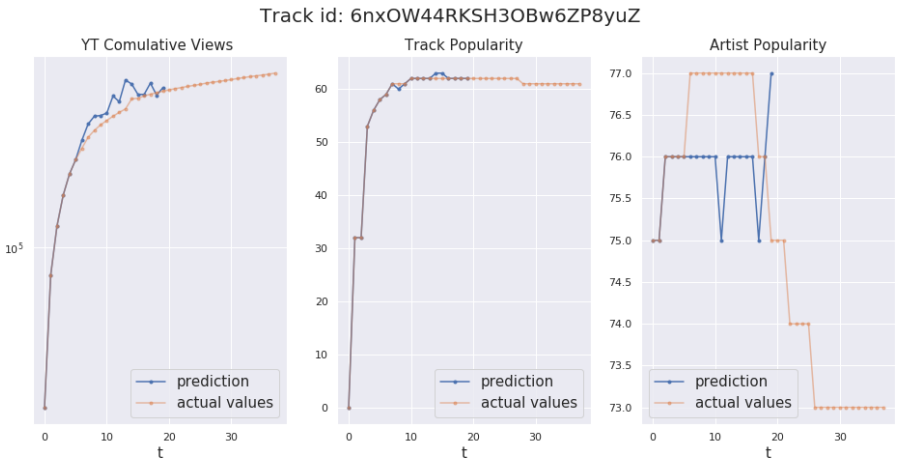

Figure 3.13: Track_id: 6nxOW44RKSH3OBw6ZP8yuZ. Left panel: YouTube Views time series regression. Central panel: Spotify Popularity regression. Right Panel: Artist Popularity regression. In orange the actual time series, in blue the regressions.

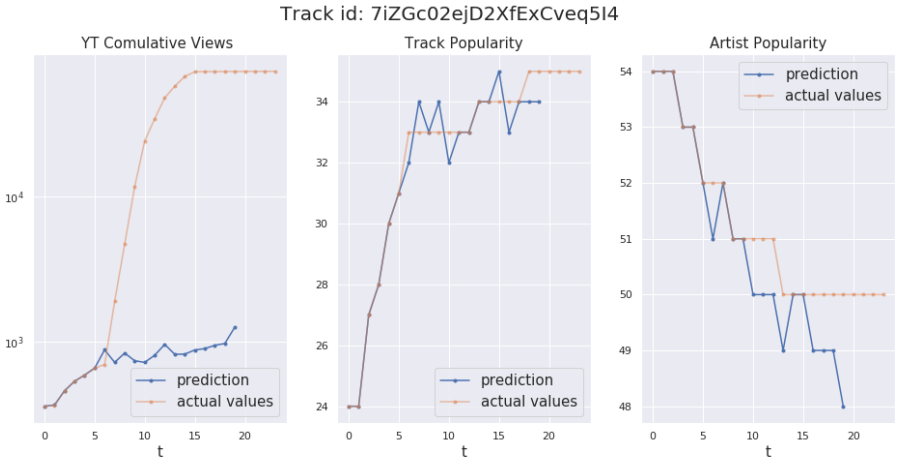

Figure 3.14: Track_id: 7iZGc02ejD2XfExCveq5I4. Left panel: YouTube Views time series regression. Central panel: Spotify Popularity regression. Right Panel: Artist Popularity regression. In orange the actual time series, in blue the regressions.

Chapter 4

Detecting Causality: Granger Causality and Transfer Entropy

In this chapter we first remark some elements of Information Theory in order to smoothly introduce the measure of *Transer Entropy* (TE) as tool for detecting the flow of information between time series, first defined by Thomas Schreiber in 2000. We also throw some lights on the concept of causality, presenting the definitions proposed in 1969 by Clive W.J. Granger, Nobel Prize of Economics Science in 2003.

4.1 Elements of Information Theory

The key concept of this chapter and of the next analysis is the physical quantity of *entropy*. It's the measure of uncertainty associated to a random variable X. It doesn't depend on the actual values taken by X, but only on the probabilities. Let X be a discrete random variable with alphabet $\mathcal{X}$ and probability mass function $p(x) = \Pr\{X = x\}, x \in \mathcal{X}$, the formal definition of entropy is the following [10]:

$$H[X] = -\sum_{x \in \mathcal{X}} p(x) \log p(x) \tag{4.1}$$

The log is meant to be in base 2 and entropy is expressed in bit. Moreover, since adding terms of zero probability to the possible realisation of the random variable does not change the state of knowledge about the random process, we will use the convention that $0 \log 0 = 0$, which is also justified by continuity since $x \log x \to 0$ as $x \to 0$. Although other definitions for the entropy has been provided, as the

Tsallis Entropy[1] , it is possible to show that 4.1 can be derived enforcing certain simple and plausible requirements[11]:

- if $\forall x \in \mathcal{X}, p(x) = \frac{1}{|\mathcal{X}|}$, then the information gained should be a monotonically increasing function of $|\mathcal{X}|$: wider is the range of possible outcomes, larger is our lack of knowledge;
- if our systems is composed of two independent sub-system, then we should be able to write the total information gained as the sum of the information gained for the two sub-systems: additivity of the entropy for independent random variables;
- for any partition of the set X into non overlapping subsets Θ , the optimal number of questions $H[X]$ to elicit X should equal the sum of the optimal number of questions $H[\Theta]$ to elicit the subset Y where X belongs, and the expected optimal number of questions $H[X|Y]$ to elicit X within Y.

The entropy $H[X]$ of a random variable X can be regarded equivalently as the information content gained once the value of X is elicited. Indeed, we remark that 4.1 can be formulated starting from the definition of the so called *surprisal* or *Shannon information content* of an event x [12],

$$\eta(x) = -\log p(x) \tag{4.2}$$

and so $H[X] = \mathbb{E}[\eta(x)]$. One can interpret the values of $\eta(x)$, in bits, as the optimal number of binary questions that one needs to ask (on average) to determine the value of X when its outcome is x. Therefore entropy associated to a random variable X corresponds to the *minimal expected number* of yes/no questions needed to elicit the value assumed by the random variable. In a straightforward manner equation 4.1 can be generalized for two variables, the *joint entropy* is simply:

$$\mathrm{H}[X, Y] = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y) = \mathbb{E}[\eta(X, Y)] \tag{4.3}$$

and so for any number of variables. We also need the idea of *conditional entropy*, the uncertainty left after we have taken into consideration some information. If the outcome of the random variable Y is y and this result is known, the lack of knowledge on X is possibly reduced (or in the "worst" case is the same) and it's given by:

$$\mathrm{H}[X|Y = y] = -\sum_{x \in \mathcal{X}} p(x|y) \log p(x|y) \tag{4.4}$$

[1]It's a non-additive generalization of the Shannon Entropy: $H_T[p](q, k) = \frac{k}{q-1}\left(1 - \sum_i p_i^q\right)$ were k is a positive constant. 4.1 is recovered in the limit $q \to 1$, fixing $k = 1$

The *conditional entropy* is thus given averaging Eq. 4.4 over all the possible realizations of Y:

$$\begin{aligned} \mathrm{H}[X|Y] &= \sum_{y\in\mathcal{Y}} p(y)\,\mathrm{H}[X|Y=y] = \\ &= -\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(y)p(x|y)\log p(x|y) = \\ &= -\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x,y)\log p(x|y) = \\ &= \mathbb{E}_{X,Y}[\eta(X|Y)] \end{aligned} \tag{4.5}$$

where we have introduced the $\eta(x|y) = \eta(x,y) - \eta(y)$, the *conditional Shannon information content*. Finally we introduce the concept of *mutual information* between two random variables X and Y. It answers to question: " How much information we gain about one of the two random variables when the other is elicited?". If X and Y are independent we expect to gain no information about X if we elicit Y, and vice versa must be true too. In [13] Fano proved that the only expression satisfying certain constraints is:

$$\begin{aligned} \mathrm{I}[X:Y] &= \mathrm{H}[X] - \mathrm{H}[X|Y] \\ &= \mathbb{E}[\eta(X)] - \mathbb{E}[\eta(X|Y)] = \\ &= \mathbb{E}[\eta(X) - \eta(X,Y) + \eta(Y)] = \\ &= \mathbb{E}\left[-\log\frac{p(X)p(Y)}{p(X,Y)}\right] = \mathbb{E}\left[i(x:y)\right] = \\ &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \end{aligned} \tag{4.6}$$

where we have introduce the *point-wise local mutual information* $i(x:y) = \log\dfrac{p(x,y)}{p(x)p(y)}$. It's immediate to realize that the mutual information is a symmetric measure, i.e. it is invariant under the exchange $X \leftrightarrow Y$. This is completely consistent to the fact that the two random variables provide the same amount of information on the other. Moreover, this definition also satisfies the requirement that for independent random variables $I[X:Y] = 0$: in that case $\forall(x,y) \in (\mathcal{X},\mathcal{Y}), \log\dfrac{p(x,y)}{p(x)p(y)} = 0$.

The mutual information is strictly connected with the *Kullback-Leibler Divergence* (KLD) [14], a measure of the distance between two distributions, $p(x)$ and $q(x)$ on the same alphabet $\mathcal{X}$:

$$\mathrm{D}[p|q] = \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)} \tag{4.7}$$

Usually interpreted as *distance* between probability distribution, the relative entropy D$[p|q]$ is a non-negative [2] measure of the inefficiency of assuming that the distribution is $q(X)$ when the true distribution is $p(X)$. For example, if we knew the true distribution $p(X)$ of the random variable X , we could construct a code with average description length H$[X]$. If, instead, we used the code for a distribution $q(X)$, we would need $H[X] + \mathrm{D}[p|q]$ bits on the average to describe the random variable [15]. In this framework Eq. 4.6 can be interpreted as the *misleading information* gained considering X and Y to be independent each other.
In the case of a *continuous* random variable, a naïve generalization of Eq. 4.1 and 4.7 leads to

$$H\left[p\right] = \int_{\Omega_X} p(x) \log p(x) dx \tag{4.8}$$

and

$$\mathrm{D}[p|q] = \int_{\Omega_X} p(x) \log \frac{p(x)}{q(x)} dx \tag{4.9}$$

where with Ω_X we indicate the support of X. It's important to stress that while the entropy of a discrete distribution is a non-negative measure, invariant under the transformation $X \to Y = f(X)$, in some case the 4.8 can assume negative values. This is due to the fact that the number of expected binary questions needed to elicit the value of a continuous random variables tends to infinity. To overcome this issue, it's possible to define the so called *differential entropy*. Given Δ the resolution at which the continuous variable is resolved, it's possible to map the X to its discrete version $\hat{X}$ defining a probability mass function as

$$p(x_i) = \int_{i\Delta}^{(i+1)\Delta} dx p(x) \tag{4.10}$$

and to write the entropy associated to the quantized random variable as the sum of two contributions

$$\begin{aligned} H\left[\hat{X}\right] &= -\sum_i p\left(x_i\right) \Delta \log\left[p\left(x_i\right) \Delta\right] \\ &\simeq h[X] - \log \Delta \end{aligned} \tag{4.11}$$

with the fist term known as the *differential entropy*

$$h[X] = -\int_{\Omega_X} dx p(x) \log p(x). \tag{4.12}$$

We conclude this paragraph remarking the link between the Shannon entropy and the *Gibbs entropy*. It's possible to show that, given a system described by an

[2] It can be easily proved using Jansen Inequality

Hamiltonian cost function $\mathcal{H}$ defined on the set of possible states of the systems $\{\xi_i\}$, the Boltzmann distribution, i.e. the statistic describing a *canonical ensamble*, can be derived looking for the distribution that maximizes the entropy, keeping the average energy of the system $\mathbb{E}[\mathcal{H}] = E$ fixed. Let Z being the canonical partition function, if one defines the *Helmotz free energy* $F = -k_B T \log Z$, it's immediate to show the equivalence between the Shannon entropy and the Gibbs entropy S:

$$\begin{aligned} S = \frac{U - F}{T} = \frac{1}{T} \sum_i p(\xi_i)\mathcal{H}(\xi_i) + k_B \ln Z \sum_i p(\xi_i) = \\ = -k_B \sum_i p(\xi_i) \left(-\frac{\mathcal{H}(\xi_i)}{k_B T} - \ln Z \right) = -k_B \sum_i p(\xi_i) \ln p(\xi_i). \end{aligned} \tag{4.13}$$

4.2 Measure of causality in time series

4.2.1 Granger Causality on the VAR model

Detection and clarification of cause–effect relationships among processes, has been a fundamental question in a wide variety of context: examples include neuroscience, physiology[16], ecological modelling[17], information flow in stock market[18], financial time series[19] and the emergence of collective behaviour[20]. Generally speaking, causation is a relationship that holds between events and it is presumed that the cause chronologically precedes the effect. Causality expresses a kind of a "law" necessity, while probabilities express uncertainty, a lack of regularity. Despite of these definitions, causal relationships are often investigated in non-deterministic or deterministically chaotic systems. A first attempt to measure causality among stochastic time series is due to Clive William John Granger, a British econometrician Nobel Memorial Prize in Economic Sciences in 2003. In [21] he proposes some heuristic definitions of causality, shaping the so called *Granger causality*. We shortly report the core ideas.
Given A_t a stationary stochastic process, let A_t^* represent the set of past values $\{A_{t-j}, j = 1, 2, ..., \infty\}$ and A_t^{**} be the set of past and present values $\{A_{t-j}, j = O, 1, ..., \infty\}$. Further let $A(k)$ be the set $\{A_{t-j}, j = k, k+1, ..., \infty\}$. Denote the optimum, unbiased, least-squares predictor of A_t using the set of values B_t by $\hat{A}_t(B)$. Thus, for instance, $\hat{X}_t(X^*)$ will be the optimum predictor of X_t using only past X_t. The predictive error series will be denoted by $\epsilon_t(A|B) = A_t - \hat{A}_t(B)$. Finally, let $\sigma^2(A|B)$ be the variance of $\epsilon_t(A|B)$ and U_t be all the information in the universe accumulated since time $t-1$ and let $U_t - Y_t$ denote all this information apart from the specified series Y_t. We then have the following definitions:

CAUSALITY: If $\sigma^2(X|U) < \sigma^2(X|U^* - Y^*)$, we say that Y is causing X, denoted by $Y_t \Rightarrow X_t$. We say that Y_t is causing X_t if we are better able to predict X_t using all available information than if the information apart from Y_t had been used.

FEEDBACK: If $\sigma^2(X|U^*) < \sigma^2(X|U^* - Y^*)$ and $\sigma^2(Y|U^*) < \sigma^2(Y|U^* - X^*)$, we say that feedback is occurring, which is denoted$Y_t \Leftrightarrow X_t$, i.e., feedback is said to occur when X_t is causing Y_t and also Y_t is causing X_t.

INSTANTANEOUS CAUSALITY: If $\sigma^2(X|U^*, Y^{**}) < \sigma^2(X, U^*)$, we say that instantaneous causality $Y_t \Rightarrow X_t$, is occurring. In other words, the current value of X, is better "predicted" if the present value of Y, is included in the "prediction" than if it is not.

CAUSALITY LAG If $Y_t \Rightarrow X_t$, we define the (integer) causality lag m to

be the least value of k such that $\sigma^2(X|U - Y(k)) < \sigma^2(X|U - Y(k+1))$. Thus, knowing the values $Y_{t-j}, j = \{0,1,...,m-1\}$, will be of no help in improving the prediction of X_t. The definitions have assumed that only stationary series are involved. In the non-stationary case, $\sigma^2(X|U^*)$ etc. will depend on time t and, in general, the existence of causality may alter over time.
In [21] Granger propose to implement this formulation in a simple vector autoregression model (VAR), a stochastic process model used to capture the linear interdependencies among multiple time series. VAR models generalize the univariate autoregressive model (AR model) by allowing for more than one evolving variable. Given two stationary stochastic time series X_t and Y_t, formally we define the VAR(m) model as follow:

$$\begin{array}{l} X_t = \sum_{j=1}^{m} a_j X_{t-j} + \sum_{j=1}^{m} b_j Y_{t-j} + \epsilon_t \\ Y_t = \sum_{j=1}^{m} c_j X_{t-j} + \sum_{j=1}^{m} d_j Y_{t-j} + \eta_t \end{array} \tag{4.14}$$

where ϵ_t and η_t are taken to be two uncorrelated white-noise series, i.e., $\mathbb{E}[\eta_t \epsilon_s] = 0 \quad \forall t, s$ and $\mathbb{E}[\epsilon_v \epsilon_w] = \mathbb{E}[\eta_v \eta_w] = \delta_{v,w} \quad \forall v, w, \quad$ with $\delta_{.,.}$ being the Kronecker delta. In 4.14 m can be regarded as a measure of the "memory" of the process. For a non-markovian process m should tend to infinity but in practice, due to the finite length of the available data, m will be assumed finite and shorter than the given time series. The definition of causality given above implies that Y_t is causing X_t provided some b_j is not zero. Similarly X_t is causing Y_t if some c_i is not zero. If both of these events occur, there is said to be a feedback relationship between X_t and Y_t. If we assume *a prori* that the process are independent, the AR(m) model can be implemented as the simplification of the VAR(m):

$$\begin{array}{l} X_t = \sum_{j=1}^{m} a_j X_{t-j} + \phi_t \\ Y_t = \sum_{j=1}^{m} d_j Y_{t-j} + \gamma_t \end{array} \tag{4.15}$$

, where, again, ϕ_t and γ_t are taken to be two uncorrelated white-noise series. In order to detect the causality inside time series several types of tests can be performed. The most common one is based on the Granger-Wald test. Let $\hat{\sigma}^2_{\mathrm{VAR}}$ be the estimator of the variance of ϵ_t, $\hat{\sigma}^2_{\mathrm{AR}}$ be the estimator of the variance of ϕ_t, and N the length of the time series, therefore we introduce:

$$\mathrm{GW} = N \frac{\hat{\sigma}^2_{\mathrm{AR}} - \hat{\sigma}^2_{\mathrm{VAR}}}{\hat{\sigma}^2_{\mathrm{VAR}}}. \tag{4.16}$$

The GW statistic follows the $\chi^2(m)$ distribution under the null hypothesis of no causality [22].
Other approaches based on *spectral methods* are discussed in [21].

4.2.2 Transfer Entropy

The major drawback of the Granger Causality is to be effective only in systems where interactions between time series are linear. Moreover, Granger approach is a parametric estimation in the sense that the analysis can be performed only once the parameter m in Eq. 4.14 is fixed (one can also in principle use different values of m for each terms in 4.14). To overcome these issues it is possible to perform measures of *Transfer Entropy*. It was introduced by Thomas Schreiber in 2000 [16] and it's designed as Kullback-Leibler distance of transitions probabilities. The key idea is to extend the measure of Mutual Information (see Eq. 4.6), which is a symmetric measure, to a measure that can detect the direction of the information between two time-series, thus breaking the symmetry.
Consider a system that may be approximated by a stationary Markov process of order k, that is, the conditional probability to find I in state i_{n+1} at time $n+1$ is independent of the state i_{n-k} : $p(i_{n+1}|i_n, ..., i_{n-k-1}) = p(i_{n+1}|i_n, ..., i_{n-k})$. Henceforth we will use the shorthand notation $i_n^{(k)} = (i_n, ..., i_{n-k-1})$ for words of length k. The average number of bits needed to encode one additional state of the system if all previous states are known is given by the *entropy rate*:

$$h_I = -\sum p(i_{n+1}, i_n^{(k)}) \log p(i_{n+1}|i_n^{(k)}) \tag{4.17}$$

where the sum is meant to run over all the possible realizations of $k+1$ symbols. The physical meaning of the entropy rate is just the quantification of the surprise to see at time $n+1$ a given realization of the process I, once the previous k realizations are known, indeed

$$h_I = H_{I^{(k+1)}} - H_{I^{(k)}} \tag{4.18}$$

Let's now consider another Markovian process J. In the absence of any interaction between I and J we have that $p(i_{n+1}|i_n^k, j_n^l) = p(i_{n+1}|i_n^k)$ and also $p(j_{n+1}|j_n^k, i_n^l) = p(j_{n+1}|j_n^k)$. Schrieber measure the incorrectness of this assumption making use of a Kullback entropy by which he defines the Transfer Entropy:

$$T_{J\to I} = \sum p\left(i_{n+1}, i_n^{(k)}, j_n^{(l)}\right) \log \frac{p\left(i_{n+1} \mid i_n^{(k)}, j_n^{(l)}\right)}{p\left(i_{n+1} \mid i_n^{(k)}\right)} = H(i_{n+1}|i_n^{(k)}) - H(i_{n+1}|i_n^{(k)}, j_n^{(l)}). \tag{4.19}$$

We remark that $T_{J\to I}$ is not invariant under the exchange $I \leftrightarrow J$, which indeed defines the $T_{I\to J}$. The *net flow of information* from J to I is naturally defined as:

$$\Phi_{J\to I} = T_{J\to I} - T_{I\to J}. \tag{4.20}$$

It is easy to check that $\Phi_{J\to I} + \Phi_{I\to J} = 0$, thus the direction of the flow of information can be assigned according to the signs of the net flows, e.g. if $\Phi_{J\to I} > 0$ then the process J is said to drive the process I.

Although the straightforward derivation of the Transfer Entropy starting from basic requirements coming from information theory, it's not an easy task extracting information theoretic quantities from real observed data [23]. The goal of the overall approach is to estimate probability distributions and then entropies. The estimation gets even harder for continuous-state systems when there is not enough data to reliably estimate the probability of each state sequence, especially for high resolution measurements. The first naive attempt is to approximate probability density functions (pdf) with histograms. The main drawback of histograms is that they strongly depend on the number of bins and how they cover the support of the Random Variable.

A more sophisticated tool for estimating pdf is the so called *Kernel Density Estimator* (KDE) [24]. The key idea is to replace the concept of the fixed binning to the advantage of an estimator which relies on the point x_i of the data-set. To each point x_i, whatever its embedded dimension is, the KDE assigns a real positive number which magnitude depends on the number of point on the data-set that are "close enough" to x_i. Given a dataset ($\{x_i\}$), formally the KDE can be written as

$$\hat{f}(x) \propto \sum_{i=1}^{n} K\left(\frac{x - x_i}{d}\right) \tag{4.21}$$

where K is the kernel, i.e. a function that given two points returns a real value encoding their proximity, and d is the analogous of the *bin-width* diameter of the kernel, i.e. the resolution at which the data are probed by the Kernel. In [16] the author proposes a *flat* kernel

$$\hat{p}_r\left(x_{n+1}, x_n^{(k)}, y_n^{(k)}\right) \propto \sum_{n'} \Theta\left(d - \left|\begin{pmatrix} x_{n+1} - x_{n'+1} \\ x_n^{(k)} - x_{n'}^{(k)} \\ y_n^{(k)} - y_{n'}^{(k)} \end{pmatrix}\right|\right) \tag{4.22}$$

The norm $|\cdot|$ is simply the maximum distance but other norms and kernels can be considered, while $\Theta(x)$ is the Heaviside step function, i.e. $\Theta(x) = 1$ if $x \geq 0$ and $\Theta(x) = 0$ if $x < 0$.
In Fig. 4.1 we show a comparison between histograms and KDE. We show estimations based on a small sample of 20 data points drawn form the following bi-modal distribution:

$$X \sim \begin{cases} \mathcal{N}(0,1) & \text{with } p = 0.3 \\ \mathcal{N}(5,1) & \text{with } p = 0.7 \end{cases} \tag{4.23}$$

where the standard notation $\mathcal{N}(\mu, \sigma)$ for a *Gaussian Distribution* with mean μ and standard deviation σ is used. While the underlying probability density function is quite well visualized on the top left panel, the bi-modal nature of the probability density function describing the random variable 4.23 is not enhanced on the top

right panel. We remark that this is due only to a different choice of the positioning of the binning and not on their width, which is fixed. In the low left and in the ow right panels the pdf is estimated using respectively a *flat kernel* and a *gaussian kernel*: in both cases the bi-modal nature of the distribution generating the small sample is well visualized. Computing entropy from probability densities is done using the *resubstitution* formula[25]:

$$H = -\frac{1}{N}\sum_{i=1}^{N}\log\rho(x_i^{(m)}) \tag{4.24}$$

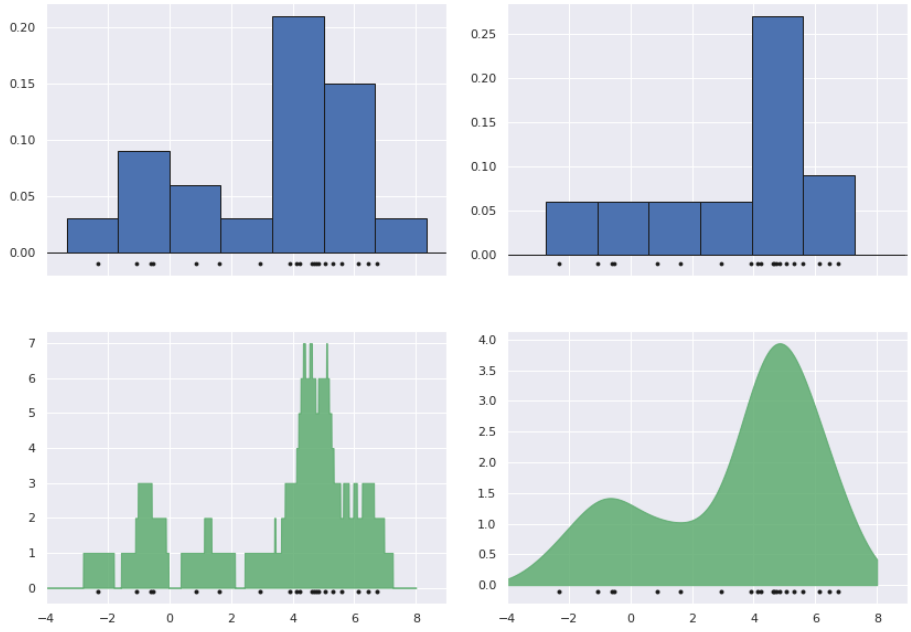

Figure 4.1: Top left: histogram estimation. Top right: histogram estimation. Button left: KDE with flat kernel. Button right: KDE with gaussian kernel. Black points represent the 20 data points forming the sample.

As an example, we illustrate in Figure 4.2 the reconstruction of a Gaussian probability distribution with 0 mean and unitary variance starting from samples. In blue the estimation of the probability density function and in red the actual pdf are given. In Figure 4.3 we plot the estimator of the entropy $\hat{H}$ as the number of the data-points on the sample increases. To be more precise, on the x-axis we report the

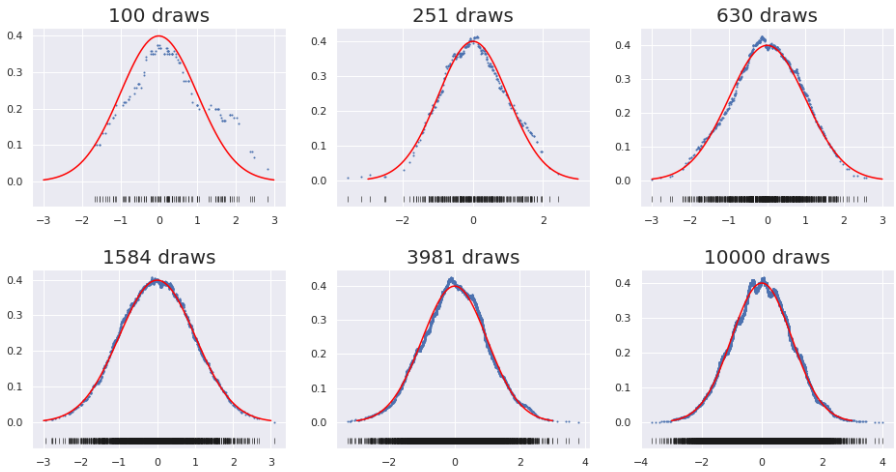

Figure 4.2: KDE estimation for 4 samples of different size generated by the same normal distribution. In red the normal distribution, in blue the estimated one via a KDE method.

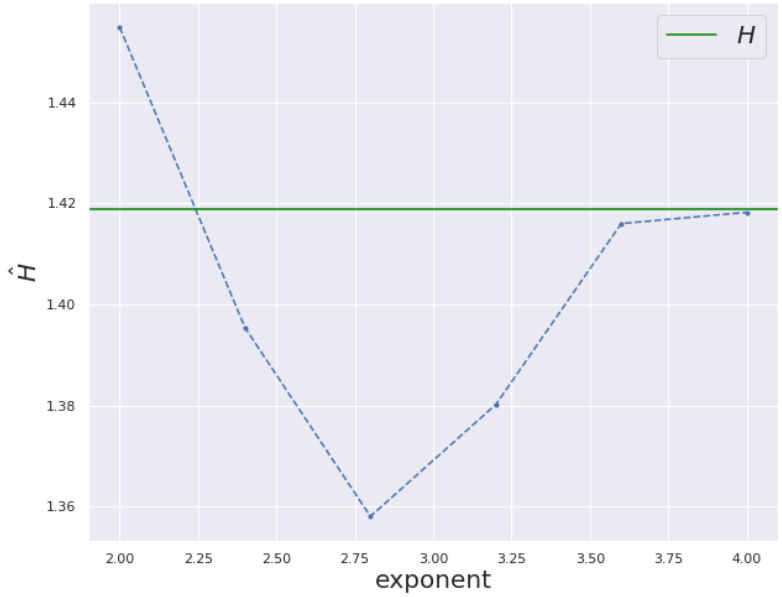

Figure 4.3: $\hat{H}$ vs $\log(\#_{data-points})$. The green horizontal line is the true line in the case of a Normal distribution. The black ticks graphically represent the draws.

logarithm in base 10 of the number of data-points forming the sample. We conclude this section introducing the the *partial transfer entropy* [26]. Consider a stochastic systems $\mathrm{U} = \{X, Y, Z, W, ...\}$ made of more than two stochastic time series. Since in principle one time series can be driven by more than one of the others, one can be interested in detecting the graph representing the causal connection between time series. Therefore, given the time series X_t and Y_t we can define the environment in which they interact as $\mathrm{E} = \mathrm{U} \setminus \{X, Y\}$. The partial transfer entropy is built conditioning Eq. 4.19 on E, i.e.:

$$\hat{T}_{X \to Y} = \mathrm{H}(y_{n+1} | y_n^{(m)}, \mathrm{E}^{(\mathrm{m})}) - \mathrm{H}(y_{n+1} | y_n^{(m)}, x_n^{(m)}, \mathrm{E}^{(\mathrm{m})}) \tag{4.25}$$

Partial transfer entropy addresses the issue of effective connectivity, being not sensitive to the effects of indirect connections which are detected as causal relations by the measure introduced by Schreiber.
In the next chapter we present some application of Granger Causality and Transfer Entropy on synthetic data already analyzed in the literature. The aim is manifold: first reproducing some results from other paper can guarantee the good functioning of the code developed, second a better understanding of the measure is provided.

Chapter 5

Transfer Entropy in Controlled Environments

In this fifth chapter we reproduce some results from the literature. Synthetic time series are generated and the TE between different stochastic processes is analyzed varying the parameters that define the couplings between them. In the first simulation we present we emphasize the importance of the diameter d of Eq. 4.21, while in the second one we remark the limit of the causality detection based on the Granger analysis, i.e. we will show how non-linearity is not detected by measures of Granger Causality. In the last two set up we introduce tests for the significance of a TE measure and we will perform a measure of partial TE in a toy model in order to show how it is possible to detect direct causality links between processes, thus how standard TE measures are effected by the global interactions among the processes living in the same stochastic environment.

5.1 TE in chaotic logistics map

In [27] Hahs and Pethel analyze the following dynamical system:

$$x_{n+1} = f(x_n), \qquad y_{n+1} = (1-\xi)f(y_n) + \xi g(x_n;\alpha) \tag{5.1}$$

where $f(x) = rx(1-x)$, $r = 4$, is the chaotic logistics map, $\xi \in [0,1]$ is the coupling strength, x_n is the state of the drive process X, and y_n is the state of the response Y. The coupling function $g(x;\alpha) = (1-\alpha)f(x) + \alpha f^2(x)$ includes a tunable parameter $\alpha \in [0,1]$ that adjusts the relative strength of the anticipatory term in the coupling function. Note that f^2 indicates that the map f is applied twice. Hence, when $\alpha = 0$ the Y system is driven towards the present X system output, and when $\alpha = 1$ the Y system is driven towards a prediction of the future X system output. When the coupling strength is above the synchronization threshold ($\xi_{sync} \geq 0.5$) the manifolds $y_n = x_n$ and and $y_n = x_{n+1}$ are stable for $\alpha = 0$ and $\alpha = 1$, respectively.

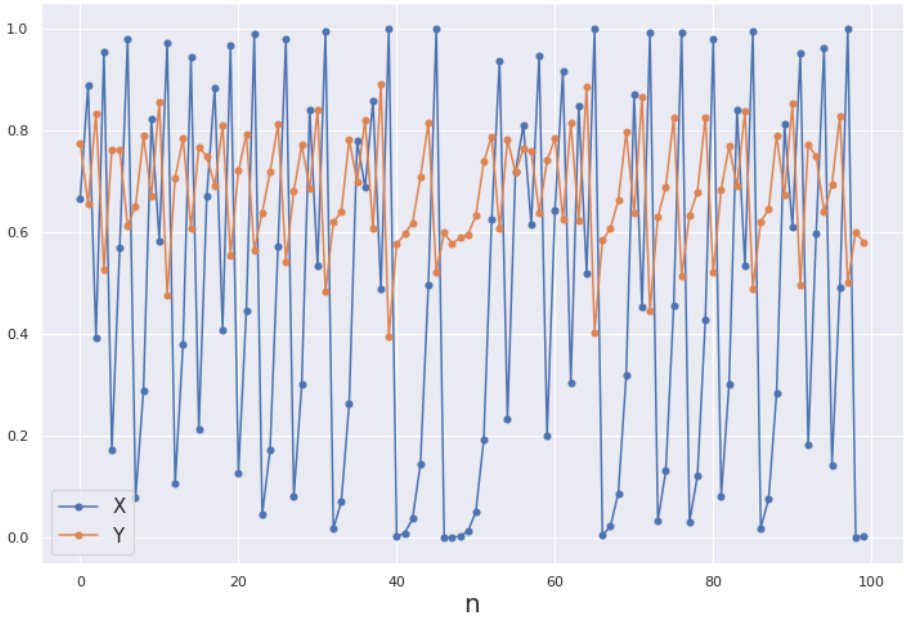

Figure 5.1: Time evolution of the dynamical system 5.1: $\alpha = 0.6$ and $\xi = 0.4$.

For α between 0 and 1 there is mixture of anticipatory and non-anticipatory chaotic dynamics. In all cases the drive and response dynamics are bounded in the [0,1] interval. Because chaotic maps generate information[28] we can investigate the information flow between the drive and response process in the dynamical

system 5.1. Here, we wish to compute the transfer entropies $T_{X\to Y}$ and $T_{Y\to X}$ as a function of the kernel diameter d for the two limit cases $\alpha = 0$ and $\alpha = 1$, setting $\xi = 0.4 < \xi_{sync}$. In the first case, due to the absence of any anticipatory drive, for all the values of all the value of d used the measure of the transfer entropy well detects the correct direction of the information flow, i.e. from the process X to Y. Differently in the second case, the expected result is obtained for $d \lessapprox 0.34$. In

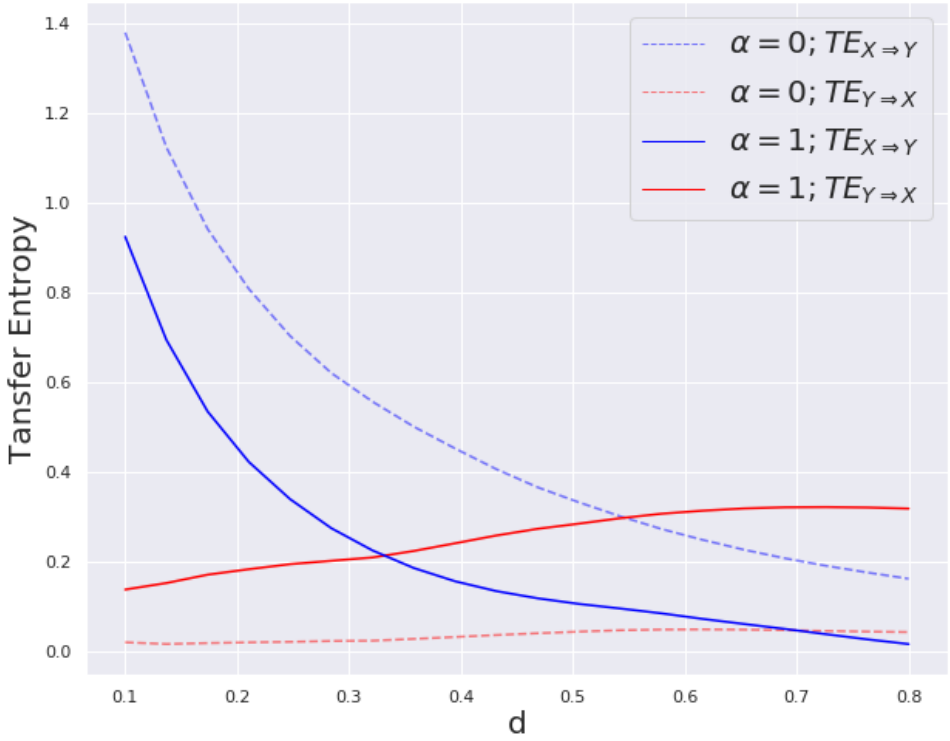

Figure 5.2: Measures of TE in the anticipatory and non-anticipatory regime for different values of d.

conclusion, we have shown that anticipatory dynamics strongly bias the transfer entropies computed from observed time series. A good choice of the kernel diameter is then crucial to accurately resolve the underlying probability distributions and thus to regard the transfer entropy as an indicator of coupling direction rather than a merely measure of where information first becomes measurable in the system.

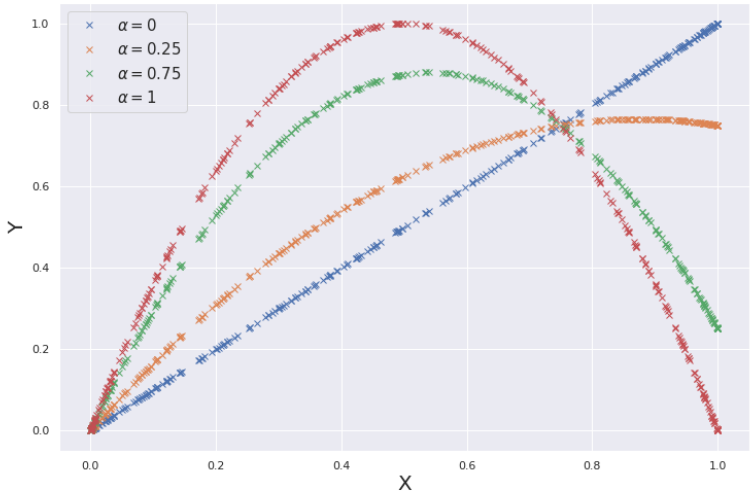

Figure 5.3: Phase space of the dynamical system 5.1. The dynamics for four different values of the parameters α are shown. The coupling parameter ξ is set to 1, i.e $x_{n+1} = f(x_n)$ and $y_{n+1} = g(x_n; \alpha)$.

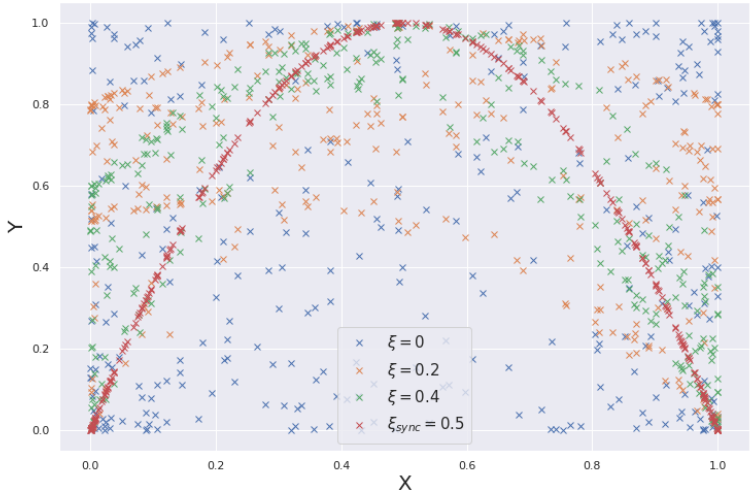

Figure 5.4: Phase space of the dynamical system 5.1. The dynamics for four different values of the parameters coupling parameter ξ are shown. The anticipatory parameter α is set to 1, i.e $x_{n+1} = f(x_n)$ and $y_{n+1} = (1 - \xi)f(y_n) + \xi f^2(x_n)$.

5.2 Non-linearity Detection

The goal of this section is to stress the major limit of Granger Causality, i.e. the failure to detect nonlinear causality interconnection between time series.
Let first consider the following stochastic system:

$$\begin{aligned} x_1(n) &= 0.4x_1(n-1) + w_1 \\ x_2(n) &= 0.5x_1(n-1) + w_2 \\ x_3(n) &= -0.4x_1(n-1) + w_3 \\ x_4(n) &= -0.5x_1(n-1) + 0.25\sqrt{2}x_4(n-1) + 0.25\sqrt{2}x_5(n-1) + w_4 \\ x_5(n) &= -0.25\sqrt{2}x_4(n-1) + 0.25\sqrt{2}x_5(n-1) + w_5 \end{aligned} \tag{5.2}$$

where $w_i \sim \mathcal{N}(0,1)$ and $x_i(n-1)$ indicates the state of the stochastic process X_i at the iteration $n-1$. Starting from random initial conditions, we run the iteration for 1000 times. In Figure 5.5 we plot the last 100 values of the time series defined

Figure 5.5: The last 100 outcomes of the stochastic process 5.2.

by 5.2. The system's dependencies are shown in Figure 5.6 which are not possible to detect just by a visual inspection of the scatter plots.

In order to reconstruct the dependencies shown in 5.6, and so to detect which series drives which other, we implement a Granger Test. Thus for each pair (x_i, x_j) we train both an AR(1) and a VAR(1) and we compare the estimators of the variance of the noise of such models, i.e. $\hat{\sigma}^2_{\text{AR}}$ and $\hat{\sigma}^2_{\text{VAR}}$. Then we compute the empirical index

$$\text{GW} = N\frac{\hat{\sigma}^2_{\text{AR}} - \hat{\sigma}^2_{\text{VAR}}}{\hat{\sigma}^2_{\text{VAR}}}. \tag{5.3}$$

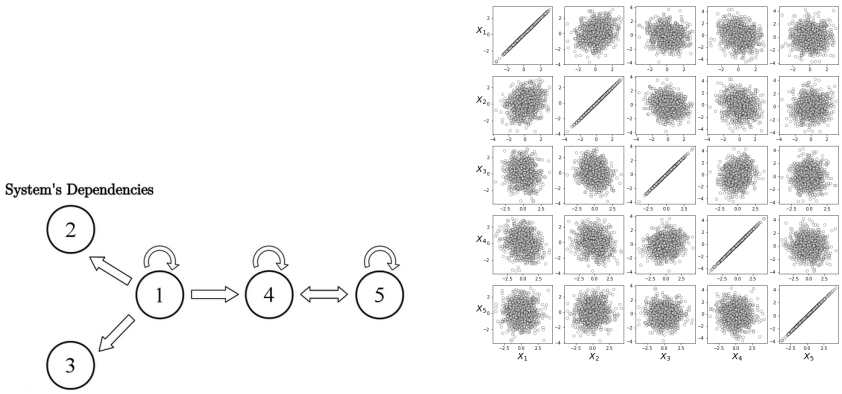

Figure 5.6: On the left a graph showing how the time series are coupled, on the right the scatter plots for each pair of time series.

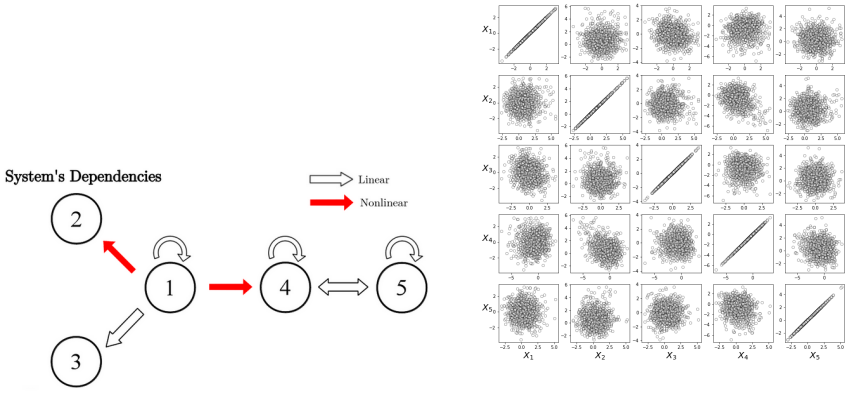

Figure 5.7: On the left a graph showing how the time series are coupled, on the right the scatter plots for each pair of time series.

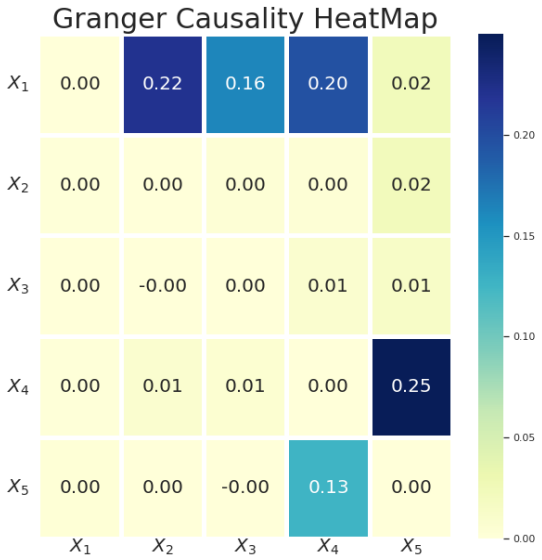

Figure 5.8: Granger Causality HeatMap for the system 5.2.

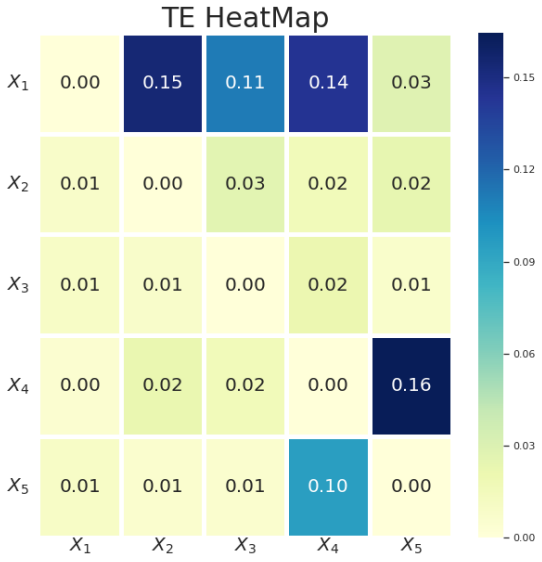

Figure 5.9: Transfer Entropy HeatMap for the system 5.2.

measuring how much the VAR(1) model performs better than the AR(1) model, which for the postulates proposed by Granger is a measure of causality.

Finally, we also estimate the TE between each pair of time series using a flat kernel with $d = 2$, and fixing $m = 1$. The results of both the analysis are shown in Figures 5.8 and 5.9. We can see how both the analysis succeed to detect all the dependencies of the system: x_1 is found to drive x_2, x_3, x_4, and x_4 and x_5 to drive each other mutually as shown in 5.6. Let now modify the linear system 5.2 adding some non-linear couplings between time-series, e.g. let consider the following system:

$$\begin{aligned}
x_1(n) &= 0.4x_1(n-1) + w_1 \\
x_2(n) &= 0.5x_1^2(n-1) + w_2 \\
x_3(n) &= -0.4x_1(n-1) + w_3 \\
x_4(n) &= -0.5x_1^2(n-1) + 0.25\sqrt{2}x_4(n-1) + 0.25\sqrt{2}x_5(n-1) + w_4 \\
x_5(n) &= -0.25\sqrt{2}x_4(n-1) + 0.25\sqrt{2}x_5(n-1) + w_5
\end{aligned} \tag{5.4}$$

As shown in Figure 5.7 a non-linear coupling is now set between x_1 and x_2 and x_1

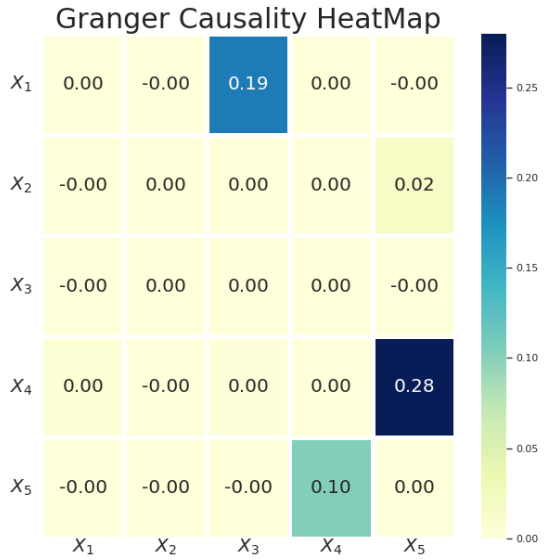

Figure 5.10: Granger Causality HeatMap for the system 5.4.

and x_4. Again a visual inspection it doesn't help to get some hints about drives. But differently from the previous case, in Figure 5.9 we clearly see a different response of the Granger Causality to the non-linearity added. This is not the case

of the detection based on TE, which still measures a causality relation between x_1 and x_2, and x_1 and x_4.

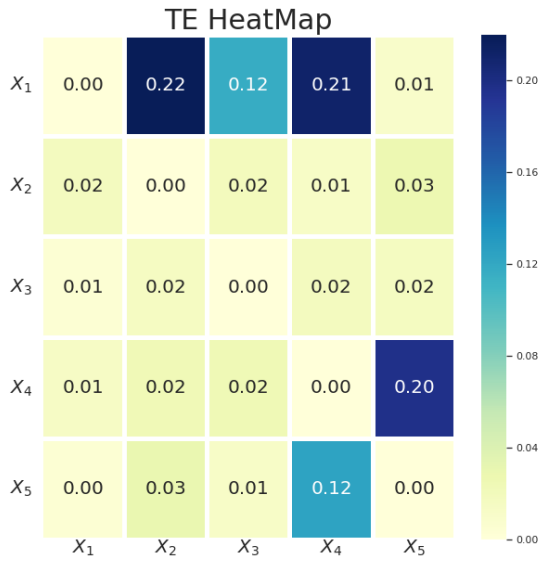

Figure 5.11: Transfer Entropy HeatMap for the system 5.4.

We want to conclude this paragraph by showing the robustness of the Granger Causality measure when a memory lag larger than necessary is chosen. This robustness represents a great quality, since in general we do not know which is the *right* lag m to use. Thus, we tend to choose the largest possible lag and it is desired that the Granger Causality remains as constant as possible as the memory lag is reached. Indeed, this is what we find. Let consider the following toy model:

$$\begin{aligned} X_t &= aX_{t-1} + bY_{t-1} + bY_{t-2} + \epsilon_t \\ Y_t &= 0.4Y_{t-1} + \eta_t \end{aligned} \tag{5.5}$$

with the same notation before introduced. In this case, the Y time series drives the X (this the direction in which we perform measures of GC), while the Y is completely independent from the other. The memory lag is $m = 2$, but what happens if we perform a Granger Causality Test using a memory lag larger than two? In Figure 5.12 for different values of the ratio $\frac{b}{a}$ we plot the Granger Causality index 4.16 for different values of the parameter m. Each point is obtained averaging over 20 measures of GC obtained on time series of length 1000. It is evident how the signal of causality increases as the ratio between the causal coefficient b and the feedback coefficient a increases, but we can also notice how the measure

doesn't change for $m > 2$, once the ratio $\frac{b}{a}$ is kept fixed. From this we learn that a significant measure of Granger Causality can't decrease importantly as the lag increases.

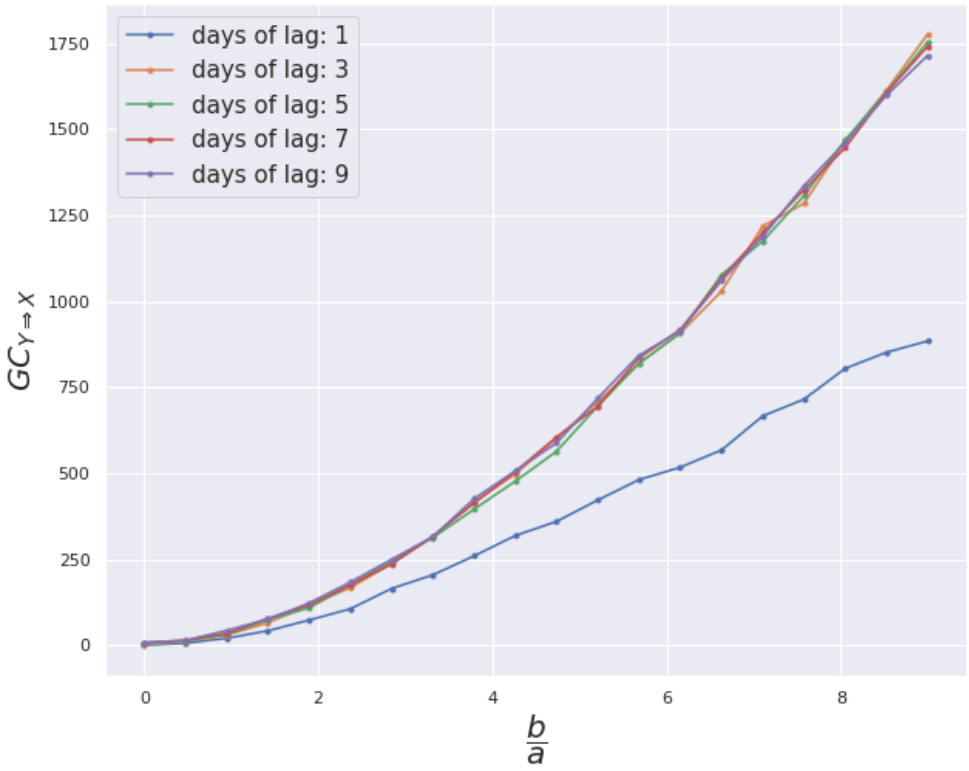

Figure 5.12: On the x-axis the ratio $\frac{b}{a}$ (see eq. 5.5, on the y-axis $GC_{Y \Rightarrow X}$. Different lines correspond to measurements performed using different values for m. is obtained averaging over 20 measures of GC obtained on time series of length 1000. The overlapping of the lines related to $m > 2$ are the of the robustness of the measurement to the overestimation of the *memory* which characterizes the system.

5.3 Partial Transfer Entropy: an application

In this last section we follow Vakorin, Krakovska and McIntosh [26]. The goal here is to show how measure *partial transfer entropy* can be performed in a stochastic system to detect direct driving links. Indeed, standard measures of transfer entropies cannot be interpreted as direct causal-effect footprints when the system is composed of processes which interact each other. To understand this, let consider the following simple stochastic system:

$$\begin{aligned} x_1(n) &= 0.9x_1(n-1) + w_1 \\ x_2(n) &= \xi x_1(n-1) + 0.9x_2(n-1) + w_2 \\ x_3(n) &= \xi x_2(n-1) + 0.9x_3(n-1) + w_3 \end{aligned} \tag{5.6}$$

with the same notation introduced before. Here ξ is meant to be a coupling parameter. In Figure 5.14 we show a directed graph where the nodes represent the

Figure 5.13: Last 300 realizations of 800 iterations of the stochastic process defined in 5.6 with $\xi = 0.3$.

time series and the edges the direction of the drive: $X_1 \to X_2 \to X_3$. There is no direct link between X_1 and X_3, but still a standard measure of TE detects a flow of information from X_1 to X_3. Setting $\xi = 0.3$ and normalizing the time series such that to have stochastic processes with mean zero and unitary variance, we analyze the time series measuring the usual net flow of information $\Phi_{X_i \to X_j}$ and the partial on $\Phi_{X_i \to X_j | X_k}$ for several values of the Gaussian KDE diameter. In Figure 5.15 we can see that $\Phi_{X_1 \to X_3 | X_2}$ is zero for a large range of values of d. In Figure 5.16 we

also test the estimation of the net flow of information for different values of the coupling parameter ξ keeping fixed the diameter d. We can clearly see how the measure is robust, i.e. $\Phi_{X_1 \to X_3 | X_2} = 0$ regardless of the increase in magnitude of the coupling χ.

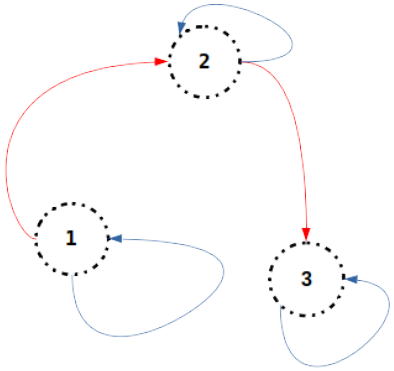

Figure 5.14: Graph of the dynamical stochastic system 5.6. Nodes represent the three time series, red edges couplings between time series and blue edges feedback mechanisms.

5.4 The testing procedure

In this last section we present an *ad hoc* testing procedure to determine the significance of a measure of TE. It is a crucial problem mostly when the analysis is performed using very short time series, and it can be used to have a rough idea about the amount of data needed to well estimate the transfer of information between two stochastic processes. The second test of the procedure is more strict but it will demand more time computation, and that is why we have preferred to build a *progressive test*. For the first part of the procedure we follow [29]. Boba *et al* stress that given two time series of length N $(x_1, x_2, ..., x_N; y_1, y_2, ..., y_N)$ the transfer entropy must be regarded as a random variable. Thus, they compute the so-called Z-score as follows:

$$Z(\mathrm{TE}) = \frac{\mathrm{TE} - \overline{\mathrm{TE}_s}}{\sigma(\mathrm{TE}_s)} \tag{5.7}$$

where $\overline{\mathrm{TE}_s}$ is the arithmetic mean of a sample of values under a null hypothesis of independence, and $\sigma(\mathrm{TE}_s)$ is the respective standard deviation of the sample.

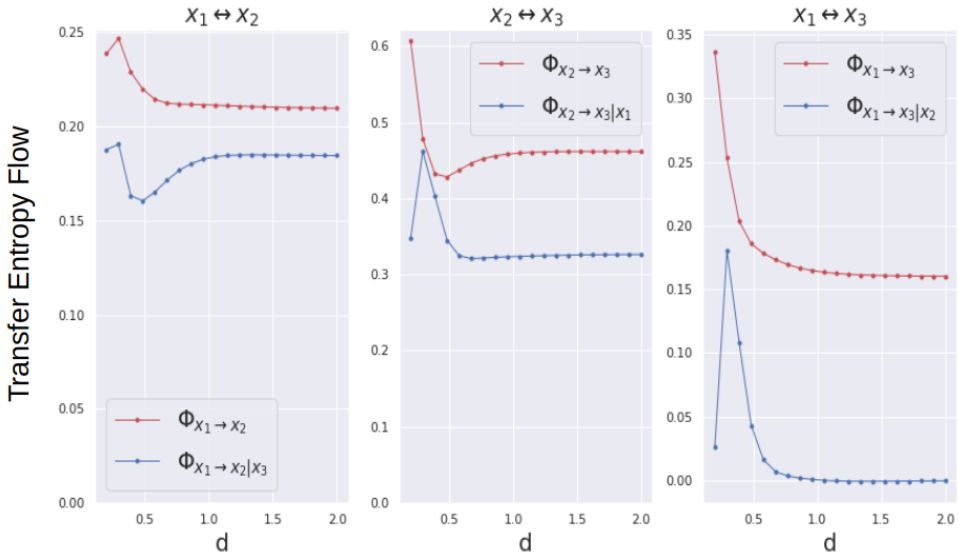

Figure 5.15: Standard and partial net flow of information for different values of the diameter d. Analysis performed using a Gaussian kernel on time series of length 800 with $\xi = 0.3$ and $m = 1$.

Assuming a normal distribution of the TE values, Z-score corresponds to one-tailed p-value. In this first part of the procedure the null hypothesis is detection of *random time-series* we propose a computational null model in which the order of elements in time-series is shuffled. $Z < 0$ or $Z \approx 0$ implies that the TE of the original data cannot be distinguished from pure random samples or shows in almost all cases less information between X and Y than a randomized sample. Only if $Z > 0$ we proceed with the second part of the test, otherwise we must conclude that the measure is not significant.
The second test we propose relies on a *bootstrap* technique[30]. We remind that the first step for measuring TE is estimating entropies, which requires a good estimator for probability density functions. To be more precise, for a given value of m one has to compute $H(i_{n+1}, i_n^m)$, $H(i_n^m), H(i_{n+1}, i_n^m, j_n^m)$ and $H(i_n^m, j_n^m)$, and so estimate the corresponding pdf. A way to test the significance of a non zero value of transfer entropy, τ, is then to compare τ with the distribution of values $\tilde{\tau}$ we obtain performing the same measure on time series which are independent each other, but that separately generate the same marginal probability distributions: this defines our hypothesis zero H_0. Let $\{xx^m\}$ and $\{yy^m\}$ be the two sets of observations of time-window of size $m + 1$ built from the time series X and Y, a value of $\tilde{t}$ can be obtained with the following bootstrap procedure:

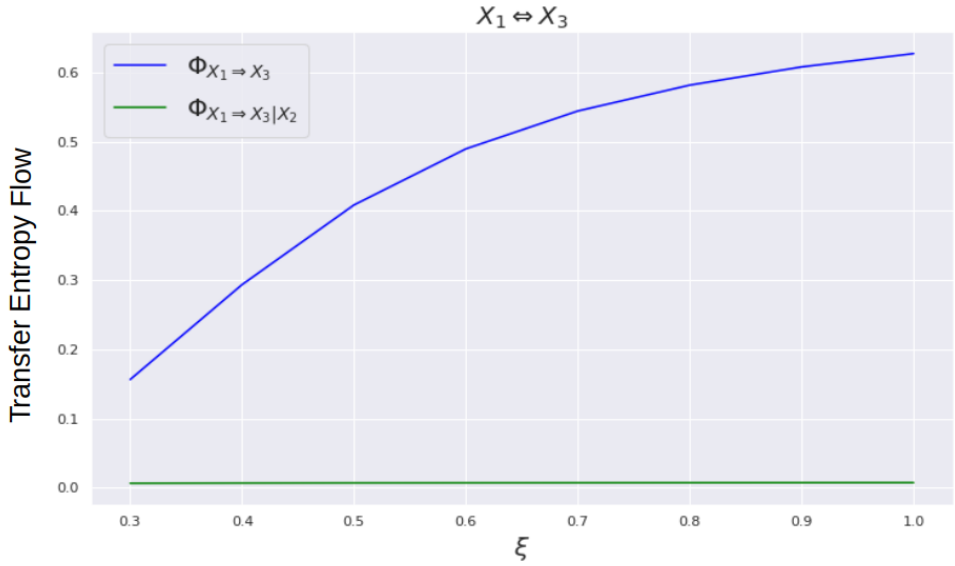

Figure 5.16: Estimation of net flow of information between the time series X_1 and X_3 for different values of the parameter ξ ($d = 1$, $m = 1$).

- **Step 1**: sample with replacement N (with N being the size of $\{xx^m\}$ and $\{yy^m\}$) time-windows from $\{xx^m\}$ and $\{yy^m\}$ independently;
- **Step 2**: to each sampled xx_i^m and yy_j^m sum a noisy vector η of the same size of the time windows. Each entry of the vector is a random variable drawn from a Gaussian distribution centred in zero, with standard deviation $\sigma = \frac{d}{2}$, where d is the diameter of the Kernel used to estimate the pdf;
- **Step 3**: perform a measure of TE $\tilde{\tau}$ on the new set $\{xx^m\}^*$ and $\{yy^m\}^*$ and repeat the entire procedure;

Step 2 guarantees that the marginal distributions of the two time series are not modified, on the other hand adding independent noises to the two set of time windows weakens any eventual drives. One can finally compares the estimation of the information flow $\Phi_{X\to Y}$ with the distribution of the bootstrapped flow and determine the *achieved significance level* of the test, abbreviated ASL. Having observed a positive flow $\Phi_{X\to Y}$, the ASL is defined to be the probability of observing at last that large a value when the null hypothesis is true,

$$\text{ASL} = \text{Prob}_{H_0}\{\Phi^*_{X\to Y} \geq \Phi_{X\to Y}\} \tag{5.8}$$

The smaller the value of ASL, the stronger the evidence against H_0:

$\text{ASL} < 0.10$ borderline evidence against H_0

$\text{ASL} < 0.05$ reasonably strong evidence against H_0

$\text{ASL} < 0.025$ strong evidence against H_0

$\text{ASL} < 0.01$ very strong evidence against H_0

Chapter 6

TE analysis on YouTube and Spotify time-series

In this final chapter we finally approach real data. We analyze the time-series associated to 751 tracks for which data from YouTube and Spotify are available since their first appearance on the market. As for this kind of measure it is crucial to have synchronized time-series, the 751 songs are those whose videos on YouTube and tracks on Spotify were uploaded at the same day. The six time-series analyzed are then:

- Spotify Popularity (SP)
- Followers of the Artist (FLL)
- Like on YouTube (LK)
- Dislike on YouTube (DLK)
- Views on YouTube (VW)
- Comments on YouTube (CMM)

Our hypothesis is that SP is a discrete time-series compared the others, which are characterized by a richer dynamic. Therefore, we expect a net flow of information from the indices of YouTube to the Spotify Popularity, which is meant to be an index that is assigned by the Swedish platform recapitulating the interest for that track and artist. The final goals are (1) to produce a heat-map and the related graph showing interdependencies between all the time-series already introduced and (2) to understand which time-series has the strongest driving influence on the others.

6.1 Causality Inference

Before performing any measure to detect drives we first normalize the data. We remind that the Spotify Popularity index ranges between 0 and 100 assuming only integer values, differently from the other indices which do not have any upper bound. Therefore given a time-series X , e.g. the VW, we build the *daily percentage increment* associated time-series $X_{daily}(t) = \frac{X_{t+1}-X_t}{X_t}$[1]. Moreover, in order to exclude from the analysis the plateau which characterizes the Spotify Popularity index of the majority of the tracks, we consider only the first 15 days. This is done for each track. After that, we merge all the time-series relating to a given index into a unique signal keeping in mind that sequences of different tracks do not "interact". Indeed our assumption is that the processes according to which time-series are mutually influenced are stationary on the time scale we analyze the data. In other words, we assume that for any two time-series X and Y there is an interaction which doesn't vary in time and which does not depend on the music track is associated to. It is also reasonable to assume that such an interaction is not deterministic and that the process is affected by some noise.
For the first analysis we propose is a Granger Causality test. Therefore, for each pair of time-series we implement both an AR(9) and VAR(9) model. We look then for the coefficients in 4.14 that minimize the sum of the squares of the differences between the observed dependent variables (values of the variables being observed) in the data-set and those predicted by the models, i.e. we implement a simple ordinary time least squares method. We remark that since our assumption of universality of the function describing the responses between time-series, we are not looking for a set of coefficients for each track, rather one that can fit the entire data-set. In Figure 6.1 we show the analysis for two pairs of time-series. On the left panels we plot the estimation of variance of the noise as function of m, for both the VAR (solid line) and AR (dashed line) model. Just by looking at how the solid lines overlapping the dashed ones, and the fluctuations of the Granger Index, we can conclude that in all the cases VAR model doesn't perform better than an AR model. These evidences suggest that no linear relationships between the time-series analyzed are present, and that we have to investigate for non-linear interactions. Indeed, if any information flow is detected performing measures of TE, we can safely conclude that this is not due to linear interactions, otherwise Granger Analysis would have detected them.

[1] In the case of the FLL we multiply the associate daily percentage increment by 100 in order to have a signal of the same magnitude of the others.

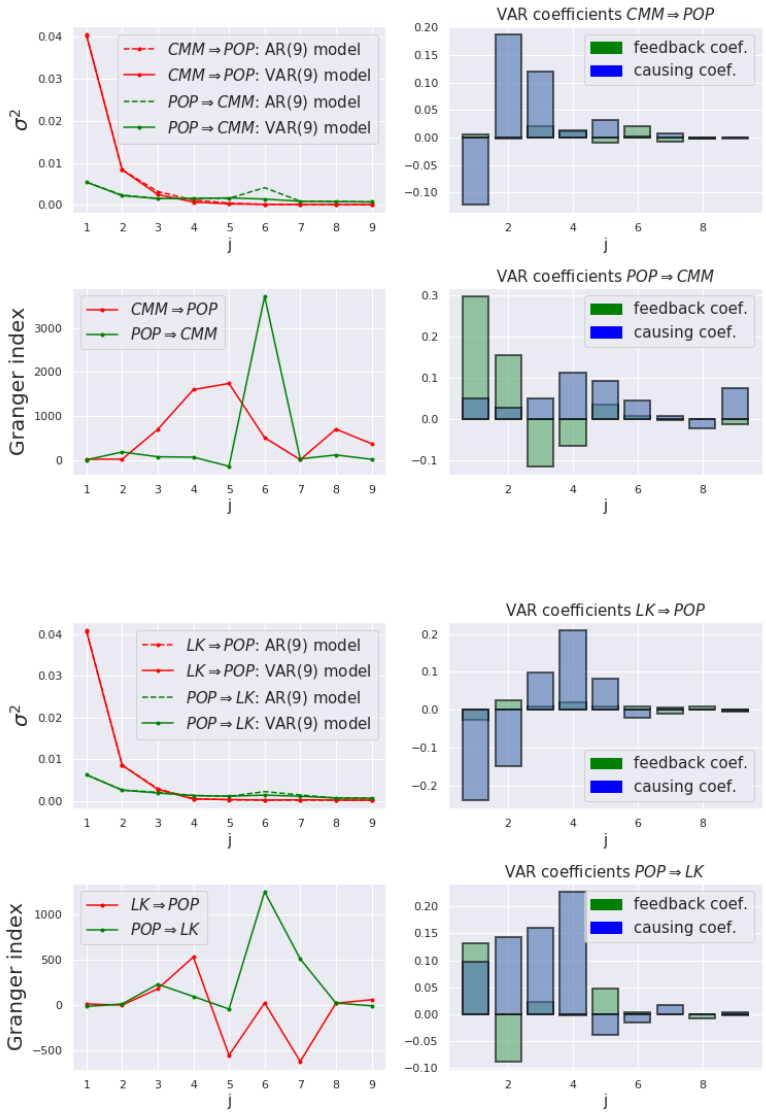

Figure 6.1: Top left panel: estimated variance for the two predictive model as function of m. Solid line are associated to VAR regression model, dashed line to AR. Bottom left panel: Granger Index versus m. On the right side the coefficients of the VAR(9) are plotted. Feedback and causing coefficients are referred to the notation of Granger. The two plots are representative of the pairs (POP, CMM) (upper panels) and (LK, POP) (lower panels).

In order to perform measure of TE it's necessary to fix the size of the diameter

of kernel d and the size of the time window m. Although it is always recommended to use a large value for m since in principle for a non Markovian process m should tend to infinite, there is no precise prescription in the literature about how to chose the right value for d, only some rule-of-thumb recommendation. Given a kernel diameter d and a pair (X, Y), we measure $T_{X\to Y}$ and $T_{Y\to X}$ for several values of m ($m_{max} = 11$). If the measure pass the first step of the testing procedure, i.e. the Z-score associated is positive, we compute the net flow $\Phi_{X\to Y}$. We observe that, after performing several shuffling and consequently measuring the information flow, we find a value of the $\sigma(\mathrm{TE}_s)$ of the order of 10^{-6}. This means that as soon the measure of transfer entropy on the original data is larger than the signal we measure on the shuffled data, we can proceed with the bootstrap test. Graphically, if the solid line falls above the dashed line, which is representative of the measures performed on the shuffled sample, we say that the measure is valuable. In Figure 6.2 we plot the TE and TEF for the pair (VW, DLK) as m varies for a fixed value of d. The same plot is presented in Figure 6.3 for the pair (POP, DLK). In these cases we fix $d = 0.06$ and we see how the signal (solid line) is strongly distinguishable from the signal that is obtained by shuffling the time-series. Once the flow of information is measured for all the two direction, one can easily estimate the net flow Φ.

In Figure 6.4 and 6.5 we plot for different values of the diameter d the TE in both the directions for two pairs of time-series (CMM, DLK) and (DLK, LK). The values of TE that do not result significant according to the Z-score test are represented with empty markers. We find that in a range from 0.05 to 0.08 every choice of the diameter leads to significant results for this first test.

In this range the value of TE doesn't remain constant as it strongly depends on the value of d, and generally speaking the direction of the net flow doesn't change, i.e., the sign of Φ remains constant. Thus we choose arbitrarily $d = 0.08$ and we test via bootstrap all the measurements. We expect to find consistent results with the first method based on the Z-score.

In Figure 6.6 and 6.7 we show for each of the 16 couples of time-series the distribution of 100 bootstrapped estimates of TEF under the null hypothesis of independence between the time-series, and the actual measured value of TEF (red line). In almost all the cases the estimation of TEF is in magnitude greater than all the elements in the bootstrapped sample, which means that the significance of the measure is confirmed. The whole analysis is repeated taking into account the environment YouTube+Spotify, i.e. implementing measure of partial TE. In Figure 6.8 we show the heat-map where the TEF ($m = 11, d = 0.08$) are reported for each pair of time-series. The rows are meant to drive the columns, e.g. $\Phi_{DLK\to POP} = 0.270$. In Figure 6.9 the same result is reported using a directed graph representation, where the thickness of the edges is set to be proportioned to the TEF between the connected nodes.

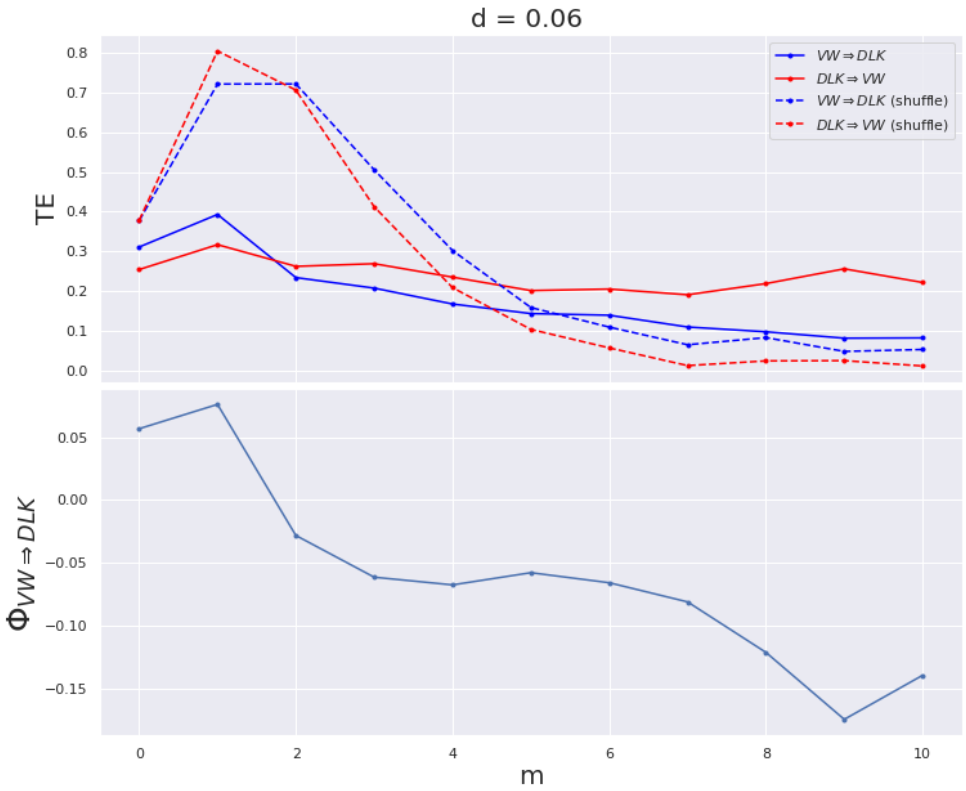

Figure 6.2: TE and TEF for the pair (VW, DLK) as function of the time window size m. The dashed lines represent the value of TE averaged on 10 measures performed on shuffled time-series ($d = 0.06$).

We find that the Spotify's popularity index is completely driven by the statistics from YouTube, mostly from the time-series related to the dislikes. This confirm our intuition that this index is just an indicator of popularity that summarizes the richer dynamics of the index of YouTube (where popularity is explicitly measured as number of views). We explain the great informative role of the dislike time-series claiming that "it is easier to express a positive feedback than a negative one". This *social behaviour* - for which users are more oriented to leave a positive feedback to a video rather than a negative one - would penalize the information associated to the likes time-series in favour of the dislikes time-series. In the graph in Figure 6.9 we also recognize the centrality of the followers time-series inside the network. It is an expected result and it perfectly matches the ever more imperative attention of

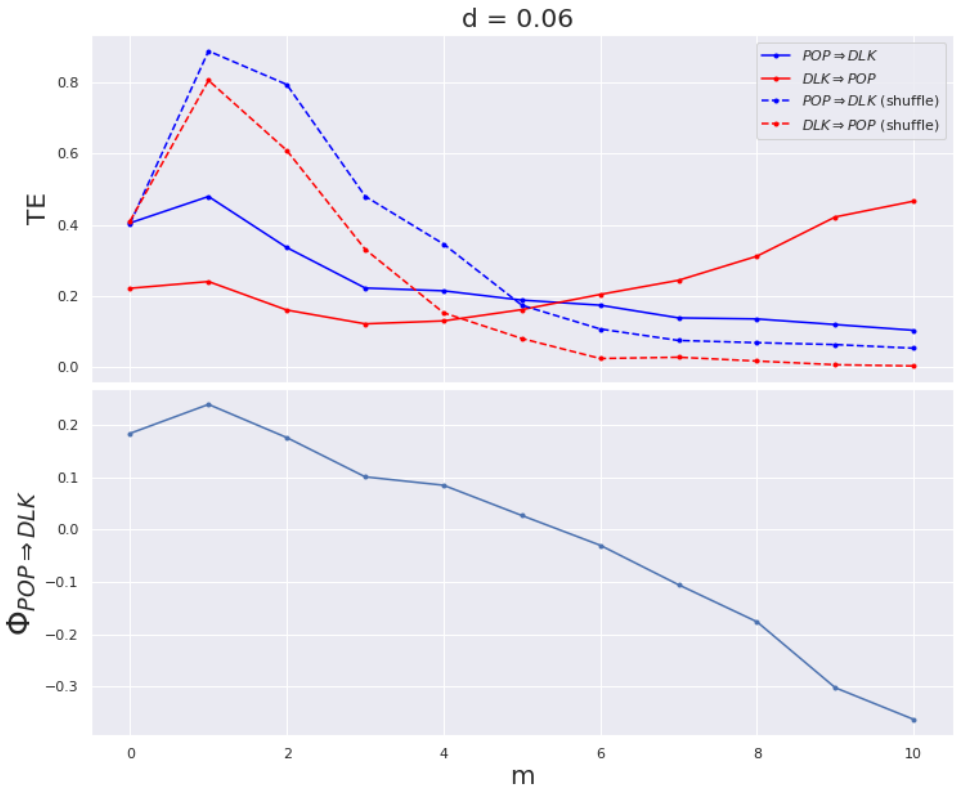

Figure 6.3: TE and TEF for the pair (POP, DLK) as function of the time window size m. The dashed lines represent the value of TE averaged on 10 measures performed on shuffled time-series ($d = 0.06$).

the markets in monitoring this index.

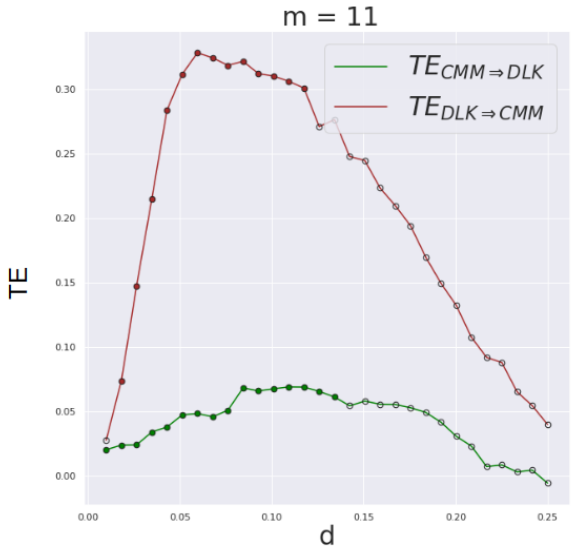

Figure 6.4: TE for different values of the diameter d the TE for both the directions (CMM, DLK). The values of TE which don't result significant according to the Z-score test are represented with empty markers.

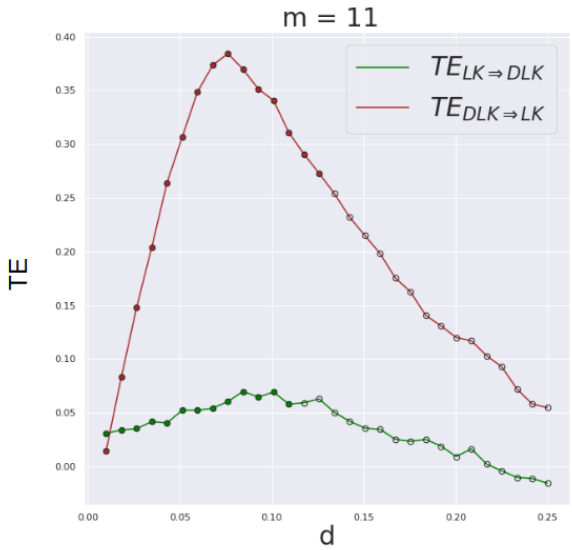

Figure 6.5: TE for different values of the diameter d the TE for both the directions (LK, DLK). The values of TE which don't result significant according to the Z-score test are represented with empty markers.

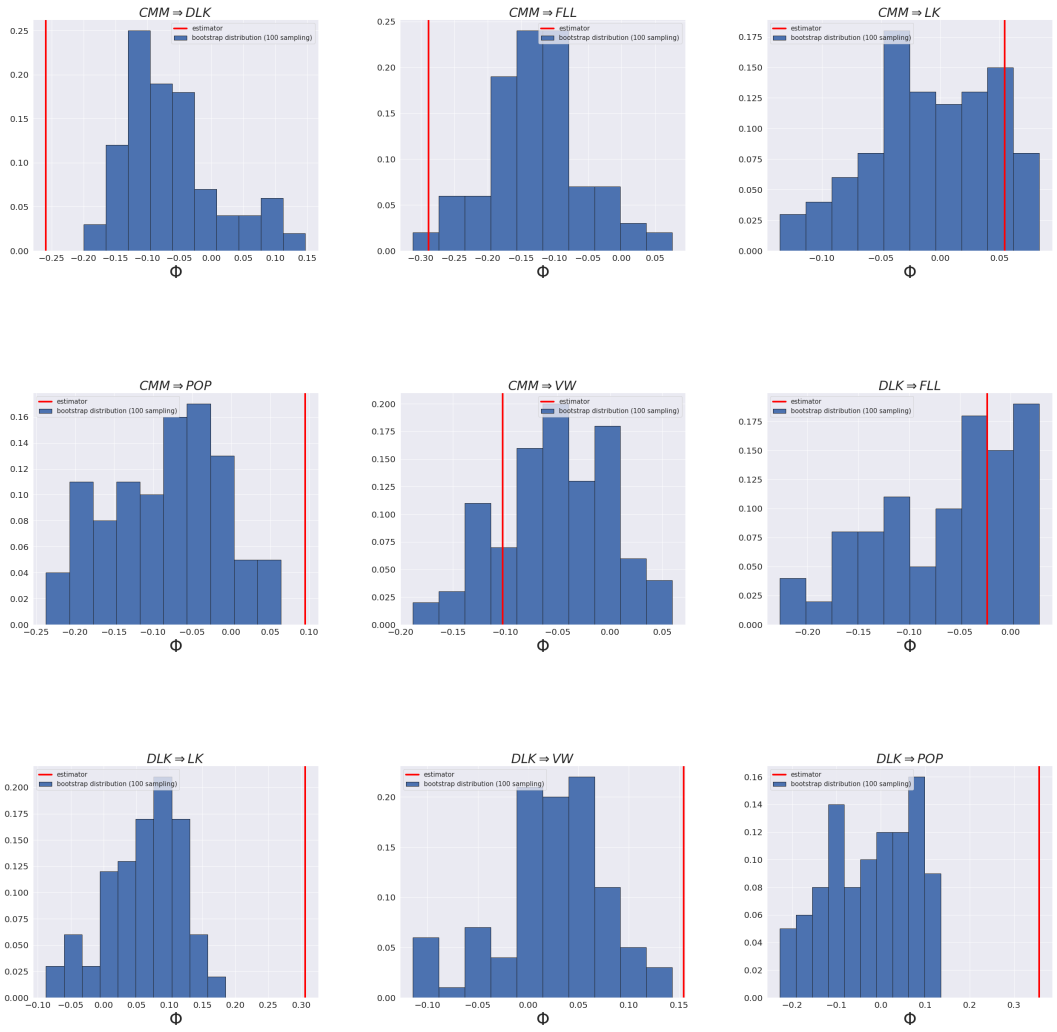

Figure 6.6: Distribution of 30 bootstrapped estimates of TEF under the null hypothesis of independence of the time-series, and the actual measured value of TEF (red line).

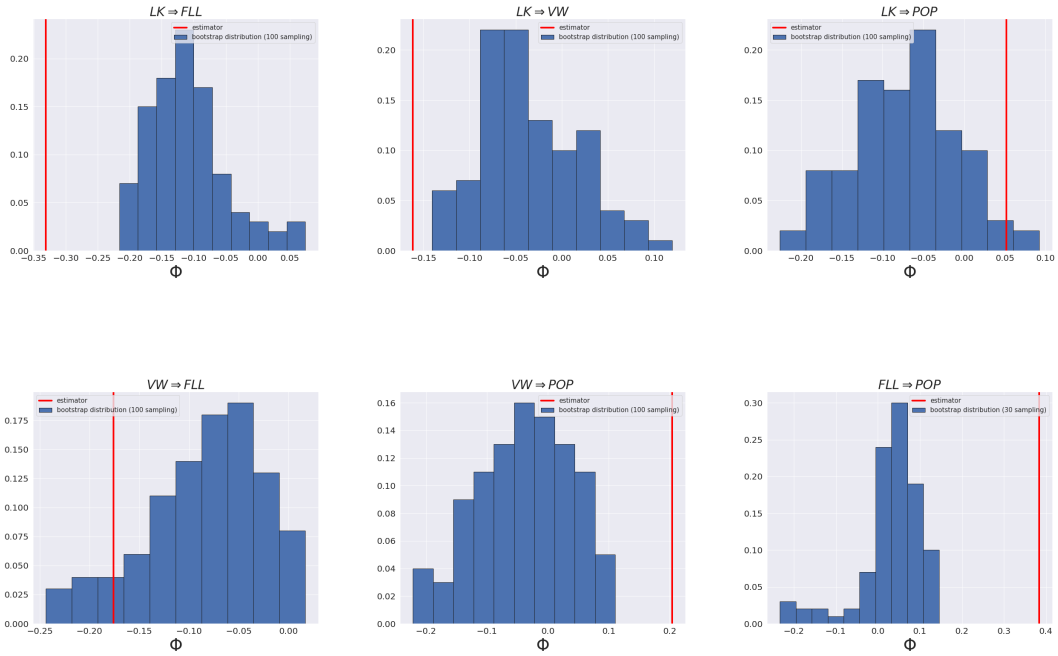

Figure 6.7: Distribution of 30 bootstrapped estimates of TEF under the null hypothesis of independence between the time-series, and the actual measured value of TEF (red line).

pair	ASL
(FLL, POP)	0.00
(VW, POP)	0.00
(VW, FLL)	0.09
(LK, POP)	0.02
(LK, FLL)	0.00
(LK, VW)	0.00
(DLK, POP)	0.00
(DLK, FLL)	0.65
(DLK, VW)	0.00
(DLK, LK)	0.00
(CMM, POP)	0.00
(CMM, FLL)	0.028
(CMM, VW)	0.20
(CMM, LK)	0.13
(CMM, DLK)	0.00

Table 6.1: Achieved Significance Level Table

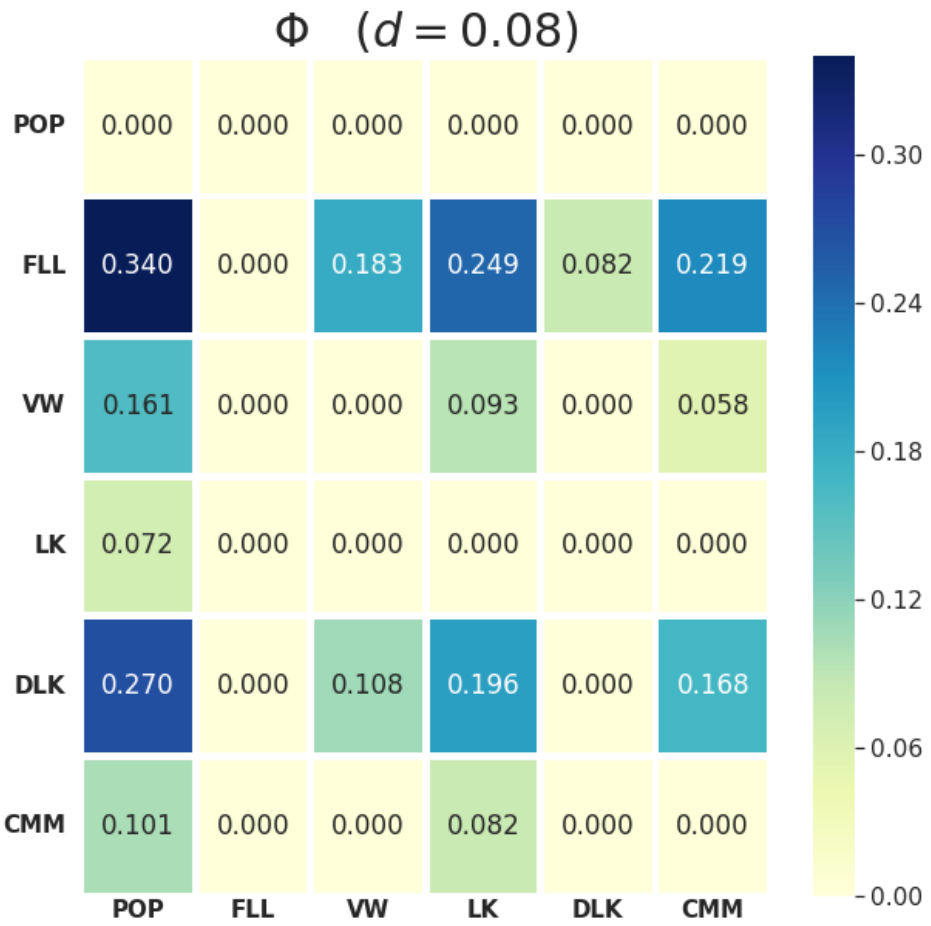

Figure 6.8: Heat-map of the TEF ($m = 11, d = 0.08$). The rows are meant to drive the columns, e.g. $\Phi_{DLK \to POP} = 0.270$.

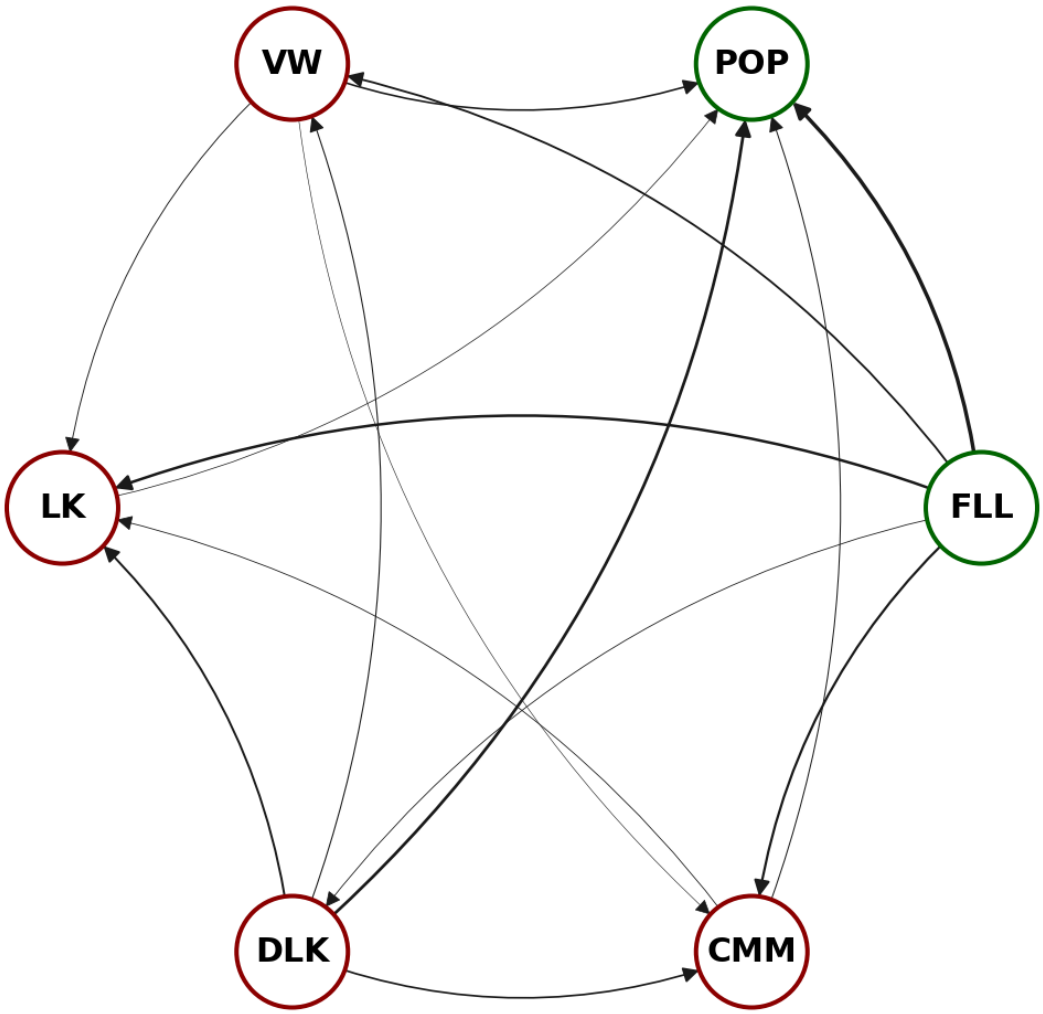

Figure 6.9: Directed graph built from the Heat-Map (Fig. 6.8). The nodes represent the time-series and the directed edges indicate the direction of the information. The magnitude of the flow of information is visually represented with the thickness of the edges.

Chapter 7

Conclusion

In this work we have shown that by analyzing track data released between January and March 2020 it is possible to train predictive models for classification and regression tasks. We saw how Random Forest algorithm efficiently classify the tracks of the data-set into the two classes "success" and "flop", just taking as input the static features that Spotify assigns to each track. As one can expect, the popularity of the artist resulted to be the most important feature when performing such classification. Also, the regression performed with the MP provided quite good results even if the fit between the predictions and the actual values of the time-series can be further improved. High frequency data rather than daily data could be useful in this sense. The most interesting results came from the analysis based on information theory tools. In this framework we were able to detect and quantify the strength and the direction of the drives between the time-series. We found that the Spotify's popularity index is completely driven by the statistics from YouTube, mostly from the time-series related to the *dislikes*. This confirm our intuition that the Spotify's popularity index is just an indicator of popularity that summarizes the richer dynamics of the index of YouTube (where popularity is explicitly measured as number of views). Moreover, the failure of Granger Causality in detecting causal relations allows us to conclude that these time-series are not linearly coupled. Although it is not possible to test it, we explain the great informative role of the dislike time-series claiming that "it is easier to express a positive feedback than a negative one". We also found the great informative role of the follower time-series, an expected result that corroborates our measurements. Moreover, the data do not allow to measure transfer entropy between the the VW time-series and the time-series of the number of plays on Spotify: in this case we could have inferred how users move from a platform to the other to listen to music contents and if there are different music consumption patterns on the two platforms depending on the music genre. Since Spotify lends itself better than YouTube for streaming audio tracks - as it is not possible to reproduce a video keeping YouTube

running in background and video streaming consumes more mobile internet data- we expect that, again, YouTube views would drive Spotify listening. More explicitly, we expect that the user may first watch the video-clip on YouTube, and after - assuming that the product satisfies his expectations - he may keep continuing to listen to the same track and other from the same artist on Spotify. In this way the information would flow again from YouTube to Spotify. Finally, we conclude that this kind of analysis could also have an important economic impact. Indeed, one day these kind of data will not be freely available anymore through API procedures, and tools such as the one developed here can help in two ways: on one hand, they can help setting the right value (and consequently the correct price) that a platform should charge for customers buying it, being the series with higher impact on others (e.g., their *weighted out degree* in the TE network). On the other hand, advertisement companies may optimize their data provisioning expenses by selecting only the relevant features to buy or scrape in order to have a real-time description of the features leading the tracks' dynamics. The latter point also holds for the feature selection of Machine Learning models. Indeed, being able to select redundant or completely dependent variables may help in building efficient predictive models that rely on a smaller set of feature for their functioning.

To conclude, our analysis reveal a rich and non-trivial network of interactions between different online streaming platforms and shade some light on the mechanisms driving the success (or failure) of a track being published on the music market. We also introduced an improved method to statistically test the significance of the Transfer Entropy information flow in time-series using a bootstrap-inspired method.

Appendix A

Back Propagation

Here we report a paragraph from the book *Neural Networks - A Comprehensive Foundation* by Simon Haykin [8], in which back-propagation algorithm is explained extensively.

The error signal at the output of neuron j at iteration n (ie., presentation of the n-th training example) is defined by

$$e_j(n) = d_j(n) - y_j(n), \tag{A.1}$$

We define the instantaneous value of the error energy for neuron j as $\frac{1}{2}e_j^2(n)$. Correspondingly, the instantaneous value $\mathcal{E}(n)$ of the total error energy can be written as

$$\mathcal{E}(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n) \tag{A.2}$$

where the set C includes en the neurons in the output layer of the network. Let N denote the total number of patterns (examples) contained in the training set. The *average squared error energy* is obtained by summing k(n) over an n and then normalizing with respect to the size N, as shown by

$$\mathcal{E}_{av} = \frac{1}{N}\sum_{n=1}^{N} \mathcal{E}(n) \tag{A.3}$$

The objective of the learning process is to adjust the free parameters of the network to minimize $\mathcal{E}_{av}$. To do this minimization, we consider a simple method of training in which the weights are updated on a *pattern-by-patter* basis until one *epoch*, that is, one complete presentation of the entire training set has been dealt with. The adjustments to the weights are made in accordance with the respective errors computed for *each* pattern presented to the network. The arithmetic average og these individual weight changes over the training set is therefore an *estimate* of the true change that would result from modifying the weights based on minimizing the

cost function $\mathcal{E}_{av}$ over the entire training set. Let neuron j being fed by a set of function signals produced by a layer of neurons to its left. The induced local field $v_j(n)$ produced at the input of the activation function associated with neuron j is therefore

$$v_j(n) = \sum_{j=0}^{m} w_{ji}(n) y_i(n) \tag{A.4}$$

where m is the total number of inputs applied to the neruon j. Hence the functional signal

$$y_j(n) = \phi_j(v_j(n)) \tag{A.5}$$

The *back-propagation* algorithm applies a correction $\Delta w_{ji}(n)$ to the synaptic weight $w_{j,i}(n)$, which is proportional to the partial derivative $\frac{\partial \mathcal{E}}{\partial w_{ji}(n)}$. We may express this gradient as:

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{A.6}$$

The partial derivative $\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)}$ represents a *sensitivity factor*, determining the direction of the search in weight space for the synaptic weight w_{ji}. Evaluating all the terms on the r.h.s. of Eq. A.6, we can write

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = -e_j(n) \phi_j^{'}(v_j(n)) y_i(n) \tag{A.7}$$

The correction $\Delta w_{ji}(n)$ applied to $w_{ji}(n)$ is defined by the *delta rule*:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}}{\partial w_{ji}(n)} \tag{A.8}$$

where η is the *learning-rate parameter* of the back-propagation algorithm. The use of the minus sign in Eq.A.8 accounts for the *gradient descent* in weight space. Accordingly, the use of Eq.A.7 in A.8 yields

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{A.9}$$

where the *local gradient* $\delta_j(n)$ is defined by

$$\delta_j(n) = e_j(n) \phi_j^{'}(v_j(n)) \tag{A.10}$$

According to Eq.A.10, the local gradient for output neuron j is equal to the product of the corresponding error signal $e_j(n)$ for the neuron and the derivative $\phi_j^{'}(v_j(n))$ of the associated activation function.
From Eq.A.10 we note that a key factor involved in the calculation of the weight adjustment is the error signal $e_j(n)$ at the output of neuron j. In this context

we may identify two distinct cases, depending on where in the network neuron j is located. In case 1, neuron j is an output node. This case is simple to handle because each output node of the network is supplied with a desired response of its own, making it a straightforward matter to calculate the associated error signal. In case 2, neuron j is a hidden node. Even though hidden neurons are not directly accessible, they share responsibility for any error made at the output of the network. The question, however, is to know how to penalize or reward hidden neurons for their share of the responsibility. This problem is solved in an elegant fashion by back-propagating the error signals through the network.

Case 1: Neuron j is an Output Node When neuron j is located in the output layer of the network, it is supplied with a desired response of its ow. We may use Eq.A.1 to compute the error signal $e_j(n)$ associated with this neuron. Having determined $e_j(n)$, it is a straightforward matter to compute the local gradient $\delta_j(n)$ using Eq.A.10.

Case 2: Neuron j is a Hidden Node When neuron j is located in a hidden layer of the network, there is no specified desired response for that metros. Accordingly, the error signal for a hidden neuron would have to be determined recursively in terms of the error signals of all the neurons to which that hidden neuron is directly connected; this is where the development of the back-propagation algorithm gets complicated. We may redefine the local gradient $\delta_j(n)$ for hidden neuron j as

$$\delta_j(n) = -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} \phi_j^{'}(v_j(n)) \tag{A.11}$$

To calculate the partial derivative $\frac{\partial \mathcal{E}(n)}{\partial y_j(n)}$, we may proceed as follows. Let k be the index for indicating output nodes

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{A.12}$$

We note that $e_k(n) = d_k(n) - \phi_k(v_k(n))$, hence $\frac{\partial e_k(n)}{\partial v_k(n)} = -\phi_k^{'}(v_k(n))$. We also note that for neuron k the induced local field is

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n) \tag{A.13}$$

Differentiating Eq.A.11 with respect to $y_j(n)$ yields

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{A.14}$$

Putting all together we ger the desired partial derivative:

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = -\sum_k e_k(n)\phi_k^{'}(v_k(n))w_k j(n) = -\sum_k \delta_k(n)w_{kj}(n) \tag{A.15}$$

Finally, using Eq.A.15 in A.12, we get the *back-propagation formula* fot the local gradient $\delta_j(n)$ as described:

$$\delta_j(n) = \phi_j^{'}(v_j(n))\sum_k \delta_k(n)w_{kj}(n) \tag{A.16}$$

Bibliography

[1] Brandt Ranj and Brandt Ranj. *We Compared Popular Music Streaming Services – These Are the Three You Should Check Out*. Rolling Stone. June 8, 2020. URL: https://www.rollingstone.com/product-recommendations/electronics/best-music-streaming-services-compared-1011378/ (cit. on p. 3).

[2] Steve Knopper and Steve Knopper. *Why Did Taylor Swift Pull All Her Music From Spotify?* Rolling Stone. Nov. 3, 2014. URL: https://www.rollingstone.com/music/music-news/taylor-swift-abruptly-pulls-entire-catalog-from-spotify-55523/ (cit. on p. 3).

[3] Kabir Sehgal Contributor. *How Spotify, Apple Music, can pay musicians more-commentary*. Jan. 26, 2018. URL: https://www.cnbc.com/2018/01/26/how-spotify-apple-music-can-pay-musicians-more-commentary.html (cit. on p. 3).

[4] Steve Knopper and Steve Knopper. *How Spotify Playlists Create Hits*. Rolling Stone. Aug. 15, 2017. URL: https://www.rollingstone.com/music/music-news/how-spotify-playlists-create-hits-200277/ (cit. on p. 3).

[5] Victor Luckerson. *How the Playlist Finally Usurped the Album*. The Ringer. Dec. 21, 2017. URL: https://www.theringer.com/music/2017/12/21/16804632/playlists-spotify-apple-music-drake-cardi-b-more-life (cit. on p. 3).

[6] Claire Shaffer and Claire Shaffer. *Music Videos Were Facing Extinction – Then YouTube Happened*. Rolling Stone. Feb. 13, 2020. URL: https://www.rollingstone.com/culture/culture-features/music-videos-youtube-951945/ (cit. on p. 3).

[7] Vladimir Svetnik, Andy Liaw, Christopher Tong, J. Christopher Culberson, Robert P. Sheridan, and Bradley P. Feuston. «Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling». In: *J. Chem. Inf. Comput. Sci.* 43.6 (Nov. 2003), pp. 1947–1958. ISSN: 0095-2338 (cit. on p. 10).

[8] Simon Haykin. *Neural Networks - A Comprehensive Foundation, Second Edition*. 2nd ed. Prentice Hall, 1998 (cit. on pp. 14, 69).

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. «Scikit-learn: Machine Learning in Python». In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 17).

[10] C. E. Shannon. «A Mathematical Theory of Communication». In: *Bell System Technical Journal* 27.3 (July 1948), pp. 379–423 (cit. on p. 28).

[11] William Bialek. *Biophysics: Searching for Principles*. Princeton University Press, 2012 (cit. on p. 29).

[12] T. Bossomaier, L. Barnett, M. Harré, and J. T. Lizier. *An Introduction to Transfer Entropy: Information Flow in Complex Systems*. 99th. Wiley series in telecommunications. Springer International Publishing, 1991 (cit. on p. 29).

[13] Robert M. Fano. *Transmission Of Information, A Statistical Theory Of Communications*. 1st ed. MIT Press, 1968 (cit. on p. 30).

[14] S. Kullback and R. A. Leibler. «On Information and Sufficiency». In: *Ann. Math. Statist.* 22.1 (Mar. 1951), pp. 79–86 (cit. on p. 30).

[15] Joy A. Thomas Thomas M. Cover. *Elements of information theory*. 99th. Wiley series in telecommunications. Wiley, 1991 (cit. on p. 31).

[16] Thomas Schreiber. «Measuring Information Transfer». In: *Phys. Rev. Lett.* 85.2 (July 10, 2000), pp. 461–464 (cit. on pp. 33, 35, 36).

[17] L. J. Moniz, J. D. Nichols, and J. M. Nichols. «Mapping the Information Landscape: Discerning Peaks and Valleys for Ecological Monitoring». In: *J Biol Phys* 33.3 (June 2007), pp. 171–181 (cit. on p. 33).

[18] Seung Ki Baek, Woo-Sung Jung, Okyu Kwon, and Hie-Tae Moon. «Transfer Entropy Analysis of the Stock Market». In: *arXiv:physics/0509014* (Sept. 2005) (cit. on p. 33).

[19] R. Marschinski and H. Kantz. «Analysing the information flow between financial time series: An improved estimator for transfer entropy». In: *Eur. Phys. J. B* 30.2 (Nov. 2002), pp. 275–281 (cit. on p. 33).

[20] Escalona-Morán, M., Paredes, G., Cosenza, and M. G. «Complexity, information transfer and collective behavior in chaotic dynamical networks». In: *arXiv:1010.4810 [nlin]* (Oct. 2010) (cit. on p. 33).

[21] Clive W J Granger. «Time Series Analysis, Cointegration, and Applications». In: *The Econometric Society* 3 (Aug. 1969), pp. 424–433 (cit. on pp. 33, 34).

[22] K Hlavackovaschindler, M Palus, M Vejmelka, and J Bhattacharya. «Causality detection based on information-theoretic approaches in time series analysis». In: *Physics Reports* 441.1 (Mar. 2007), pp. 1–46 (cit. on p. 34).

[23] Thomas Schreiber Holger Kantz. *Nonlinear Time Series Analysis*. 2nd ed. Cambridge University Press, 2004 (cit. on p. 36).

[24] David W. Scott. *Multivariate density estimation: theory, practice, and visualization*. Wiley, 1992 (cit. on p. 36).

[25] I. Ahmad and Pi-Erh Lin. «A nonparametric estimation of the entropy for absolutely continuous distributions». In: *IEEE Trans. Inform. Theory* 22.3 (May 1976), pp. 372–375 (cit. on p. 37).

[26] V.A. Vakorin, O.A. Krakovska, and A.R McIntosh. «Confounding effects of indirect connections on causality estimation». In: *Journal of Neuroscience Methods* 184.3 (Oct. 2009), pp. 152–160 (cit. on pp. 39, 50).

[27] Daniel W. Hahs and Shawn D. Pethel. «Distinguishing Anticipation from Causality: Anticipatory Bias in the Estimation of Information Flow». In: *Phys. Rev. Lett.* 107.12 (Sept. 2011), p. 128701 (cit. on p. 41).

[28] G. Györgyi and P. Szépfalusy. «Calculation of the entropy in chaotic systems». In: *Phys. Rev. A* 31.5 (May 1985), pp. 3477–3479 (cit. on p. 41).

[29] Patrick Boba, Dominik Bollmann, Daniel Schoepe, Nora Wester, Jan Wiesel, and Kay Hamacher. «Efficient computation and statistical assessment of transfer entropy». In: *Front. Phys.* 3 (Mar. 2015) (cit. on p. 51).

[30] R.J. Tibshirani Bradley Efron. *An introduction to bootstrap*. 1st ed. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Chapman & Hall, 1994 (cit. on p. 52).