UNIVERSITÀ DEGLI STUDI DI GENOVA

Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Dottorato in Matematica e Applicazioni
METODI MATEMATICI PER L'ANALISI DATI

Academic Year 2020/2021

PhD Thesis

Mathematical models for selling process optimization

Candidate:
Filippo Rossi, cycle XXXIII

Advisor:
Prof. Alberto Sorrentino

External advisor:
Dr. Anna Codispoti

"You can't connect the dots looking forward; you can only connect them looking backward. So you have to trust that the dots will somehow connect in your future."

Steven Paul Jobs, Stanford commencement address, 2005

Contents

Introduction

General premises

This work has the goal of summarizing what done during my PhD path which has been pursued in collaboration with an important company based in Genoa (Italy); due to specific company policy, I'm not allowed to share data and processes when these could be seen as properties of the company in accordance with the *Non Disclosure Agreement* in being between the University and the company itself.

It will be my effort to refer to processes and methodologies from a general perspective and to appropriately hide data, when this is possible, or simulate them when not, depending on the different scenarios.

Business framework

The following aims at presenting the reader an overview of the business context in which the Thesis work has been developed, also providing some information related to the context, necessary to better understand what done.

The research work was carried out within an important company operating in the tourism sector also active in the territory of Genoa.
The operational context has been very *agile* and throughout the path I very often interfaced with the company team of *Data Science* for the definition and the development of innovative projects. In collaboration with the team, update meetings have been held for the duration of the collaboration, aimed at better defining the different projects and understanding the theoretical aspects related to them.

An important and well-structured multi-departmental knowledge management of the company has also made it possible to efficiently organize the work; many different update meetings have been set with different teams from various departments, adding each one a specific kind of *know-how* for maximizing the outcome of each individual project, always having clear what

the scientific, the *business* and the operational needs were.
In defining, organizing and managing the different projects, I have always found extreme availability in each of the people involved, even indirectly, making sure that we always operated in a stimulating context.

The developed projects led to the release of prototypal products that have been tested and evaluated by different departments and which have produced feedback and suggestions, in order to be able to improve the very same prototypes in the future, if needed.

Focusing now on the followed projects, they can be seen as belonging to the *Destination Management* branch, that is the study and the implementation of actions aimed at better managing the company offer related to touristic experiences broadly. In particular, the first project has been related to *Revenue Forecasting* and has dealt with a revenue forecasting problem related to a specific corporate asset while the second, *Destination Discovery*, aimed at the high-level analysis of tourism opportunities in different geographical areas.

In addition to the intrinsic differences linked to their very nature, the two projects also differ in their high-level purposes; in fact, while the first is a project aimed at optimizing processes actually in progress, linked to destinations already present in the company's offer, the second is aimed at exploring new possibilities to assess the presence of opportunities, quantify them through the use of properly defined metrics and potentially lay the foundations to start exploiting them.
In the discussion, where deemed appropriate, examples relating to other possible business scenarios will be used.

What follows represents a brief summary of the content of Chapter 3, aiming at providing the reader an introduction of the chapter itself in order to approach the problems faced and have a preliminary intuition of them.

Destination Management

In a tourist company, managing destinations is clearly related to defining its offer and the revenues deriving from it. In such a light, this topic is referred to the much broader one of the *revenue management* which represents, according to [70], the set of the wide range techniques, decisions, methods, processes and technologies involved in demand management, that is the process of analyzing the demand in order to influence it thanks to the usage of different parameters. Activities like demand analysis, price and discounts definition and customer segments identification represent only a few of the whole set of actions involved in the branch.
It is clear then how analyzing the offer of a tourist company, linked to the destinations involved in its business, is greatly related to many of the topics above cited, being considerable as part of a company revenue management process.

Revenue Forecasting

Problem description

When referring to a *Revenue Forecast* problem, we mean the definition of a methodology, based on mathematical and statistical techniques, aimed at forecasting the revenue streams linked to specific items of a company.

Before approaching the topic more deeply let us make a specification: in a business framework, when talking about revenue management, there are two concepts related to revenues which are very often confused and interchanged like they represented the same thing, altough being very different, namely the *Target* and *Forecast*.
For the sake of clearity, in an organization the first is that the company would like to achieve, if all goes as expected and, generally, it pushes the activity in order to improve the current performances; on the contrary, the *forecast* represents an honest assessment about how the future performances will be like based on the most current data.
Connecting these concepts and according to these not formal definitions, the forecast should tell the organization whether it is on track to meet targets or not, representing a *status check* against targets, confirming their accuracy or providing early warning signals of incoming issues or even unveiling unexploited opportunities.

Having therefore to schematize the entire business process at a chronological level, first the *target* towards the company tends, in the time window of interest (usually the fiscal year), is defined; later, the *forecast* of the different sales is carried out on different time horizons, first long and then short, in

order to monitor the situation; finally, at the end of the sale, real revenues, forecasted revenues and the target ones are compared in order to evaluate both the corporate strategy and the forecast model performances by the mean of an *ex post* methodology; this with the goal of improving the whole process.
Other discussions about the impact of revenue forecasts in business will be provided in the next chapter.

Idea formalization

Let us now introduce the necessary formalization in order to tackle the problem of forecasting from a mathematical point of view.
Here the goal is to provide an overview that can facilitate the reader in understanding what will follow in the next chapters.

From a business point of view, revenues are the earnings that one or more items, sold by a company, generate. These can be physical products or services that the company offers to other companies (in the case of a B2B business model) or directly to customers (in the case of a B2C business).[1]
The prices of the items sold are defined by the company on the basis of many variables and logics whose analysis at this moment is beyond the scope of the work and therefore will not be addressed.

From a mathematical point of view, revenues can be formalized through a real variable $y_{\text{REV}} \in \mathbb{R}$ function of many other variables $v_1, \dots, v_n$.
Having said this, it is clear how crucial it is to identify these variables which, are obviously different depending on the business target, trying to find a suitable trade-off between simplicity of the model and realism of the assumptions.
Once this step has been carried out, it is therefore assumed that the revenues y_{REV} are function of the chosen variables only, i.e.

$$y_{\text{REV}} = G(v_1, \dots, v_n) \tag{1}$$

with G being a suitable operator.

It is important to point out that, while the revenues will always be represented by a real number[2], the variables of which they are function could be categorical, sometimes viewable as belonging to a space with implicit ordering (as an hypothetical *social status* variable) and other times without this (like a *gender* variable).

[1] B2B stands for *Business to Business* while B2C stands for *Business to Consumer*.

[2] It will be also assumed to be non negative, as the revenues are intended to be the money gained, not the ones lost.

Therefore, through this simple formalization, assuming that the variables of which y_{REV} is function are known, the object to be estimated in order to make the prediction is the functional G and as we'll better see in the next chapter, many could be the possible approaches to do that.

Destination Discovery

Problem description

If the project in the previous section is linked to the optimization of a process by the mean of an anticipated estimate of revenues, the one that will be introduced in this section aimed at researching and evaluating new possibilities for the company, in particular related to tourist interest.
In fact, for the company itself, which operates in the tourism sector, it could become of enormous value to have new possibilities of choice for the locations to be offered in its catalog.
It is specified that the term *new* mainly refers to what is the current offer of the company even if, in some cases, the term could represent novelty also in relation to what is the current competitors' offer.

The goal is to provide a tool for assessing the tourist attractiveness of specific locations; to do that the first operative step has been the definition of a metric which allowed to compare and rank different locations based on their current tourist attractiveness, once properly defined it; the second step would have been the implementation of a forecasting technique for the variables involved in the tourist attractiveness definition; the combination of the two projects would have represented the outcome of the overall process. However, the SARS-CoV-2 pandemic outbreak of the 2020 interrupted the company workflow allowing us to accomplish only the first part of the project. Because of this reason, what follows regards only the developed pilot.

Idea formalization

Conversely to the previous project in which the use of historical data within the company linked to the commercial habits of customers was crucial, in this project we relied on the use of external and public data in order to create a methodology valid both for localities already known to the company, and therefore of which the company already had some data in the history, and for new localities, for which data is certainly missing inside the company.
In this way, by relying on a representation based on external data, the data themselves have been made consistent for any location, known or unrelated to the company business.
More detailed discussions about the data and its nature are postponed to next chapters.

A set of cities of interest has been chosen as well as the source of tourist data used to represent any chosen location. The search for such a datasource has been a long and laborious process, requiring lots of efforts and time resources, having decided to resort to a single datasource because of the need for consistency of the data itself: if in fact integrating data deriving from different sources together has the advantage of increasing the mass and the completeness of the data itself, at the same time it introduces lack of coherence to the final structure, with missing and inconsistent fields between the different samples of the dataset.
Then, a search for a source that could provide a *complete* and *clean* data structure[3] has been implemented. Furthermore, the source has been sought to be subject to data reliability checks, in order to be sure of the truthfulness of the information contained in it.

Once chosen the data, it has been processed in order to build the comparison and ranking table required, defining suitable metrics to evaluate and compare the listed cities, in order to highlight their tourist importance with respect to the others.

Thesis organization

This Thesis is organized as follows.

Chapter 1 will provide preliminaries about what a *forecast* is and the many techniques aimed at accomplishing the task, giving a general theoretical framework for the topic.

Chapter 2 deals with data and the set of more practical operations needed in order to extract information from them in a numerical manner, eventually building a forecasting model on it.

Chapter 3 is related to the application of the techniques previously introduced to the different projects; for each one of them, the methodologies followed and the analyzes carried out will be provided, as well as the obtained results.

A final section containing a summary analysis of what has globally been done in the work along with different comments, the obtained results and some possible future work and improvements will conclude the Thesis.

[3]The term *complete* refers to a data structure which is almost entirely entered; using the term *clean* instead we mean a structure in which the data entered make sense in almost all cases.

Chapter 1

Preliminaries

In this chapter we provide a brief summary of the the mathematical techniques used throughout the Thesis. Because the number of techniques and methodologies involved is relatively large, a detailed mathematical treatment of each and all of them would have taken a significant amount of space, without introducing substantial novelty with respect to available books, such as [2]. For this reason, in this Thesis we made the explicit choice of providing a complete overview of the techniques, with the mathematical detail sufficient to give the idea and the intuition behind them; for more details on the formal aspects, the interested reader will find several prompts in the bibliography.

1.1 The basics of forecasting

In order to fully understand the meaning of the term *forecasting*, let us introduce the following scenario, using an approach similar to the one used in [59]:

> On March 20, 2021, the temperature will still be pretty low, something between 0 C° and 5 C° here in Genoa, northern Italy; that day will be the spring equinox and in the night many people in the hills nearby the city will be looking at the clear and shining sky.
> That Saturday afternoon instead I'll be sitting on the sofa watching the soccer match of my favourite team, drinking a hot tea and talking with some friends; it will be a very important game and, if we lose, the trainer will be for sure exonerated and my friends will certainly be happy, being fans of the opponent team; they also say that our attacker will get injured before the game ends.

This pretty short statement is very useful to highlight how the term *forecast* can be interpreted in different ways, with respect to the different

type of predictions that can be done; let us analyze some of their types:

a. **Time related:** the date indicates when the events will take place; Saturday instead is only the name given by cultural agreement to that day, in the hypotheses of using the Gregorian calendar.
From a higher perspective, a prediction is also made assuming the sequence of events will occur in that specific order and in those specific moments.
Also, the fact that there will be the equinox is related to the well known movement of the Earth around the Sun during the year.

b. **Place related**: different predictions have been made in specifying the location the events will take place in; forecast about being in Genoa or lying on the sofa are examples of such types; of course, some of these predictions can be pretty solid, like assuming to be on the planet Earth (unless you are not *Perseverance*), while other could be less probable.

c. **Behaviors related**: while forecasting more punctual *things*, such as specific people behaviours, it becomes harder and harder to be confident in what is being predicted; people drinking tea or looking at the stars are events pretty hard to anticipate, being related to boundary conditions quite impossible to forecast.

d. **Guess related**: this kind of forecasts are purely a bet on what it could be, in the sense of not being impossible; still, they are comparable to gambling; predict that a player will get injured is an example in such terms.

Observing how, from such a simple scenario, many forecasts at a different level can be done, shows how variegate can be the task of forecasting; prediction about different variables, of different type, and with different amount of uncertainty can be object of study, and the accuracy of the predictions, when possible, can vary pretty much depending on the assumptions made; also, the very same variable, in the same hypotheses, can be forecasted with different results, depending on the chosen approach.
All these facts make the topic of *forecasting* very difficult and, at the same time, very interesting to study.

1.2 From the past to the future

The concept at the core of every forecasting technique is pretty simple and is used by everyone in everyday life, very often: we try to anticipate what it will be, based on what has been in the past. In other words, we study the past, trying to identify patterns which can suggest what it will be; or again, from another perspective, we study variables, on an arbitrarily large scale, in

order to find models which can relate them by the means of equations which can be later used to study other variables with the same approach, trying to model the world, at different levels, by linking variables by the mean of *causality principle*[1], in the most reliable way.
In doing this at first, we rely on hypotheses and assumptions that in some way simplify the complexity of the world, and then we use some *priors* which represent everything has been already proved in the past about the domain of application, if possible; then the data collected, among the available ones, in the past are analyzed and used to extract other information about the relations between the variables of the domain; finally, the forecast is produced by defining a model using what obtained from the previous steps, defining a starting point representing the situation as is now and starting from it to make the forecast.

To better understand what is being said, let us consider the topic of weather forecasts which we all hear about at least once a day in our everyday life and let us try to schematize, in a very simplistic way, one possible approach in dealing with such a task in the following operative steps:

I. **Make hypotheses**: the first step is making assumptions about the domain; in a weather forecast scenario, some of these assumptions are related to the atmosphere and the particles it is constituted by. For example, the basic assumption could be assuming that it is a discrete space, instead of a continuous one; this is because of the fact that the built model will be runned on a calculator, which only works with discrete spaces.

II. **Use a priori knowledges**: the atmosphere is a fluid[2] and its state, in certain conditions, can be represented by the mean of thermodynamics and fluidodynamics equations; these models contain all the knowledges acquired and built through the years about a specific domain and let us model the world by the use of math. Their usage let us introduce the presence of specific relations between the *actors* of the domain.

III. **Extract more knowledge from the past**: depending on the scenario, this study can be done by looking data in the far past, in the near past, or both; in weather forecasts however, most of the information related the variables defining the problem are already considered in the model equations previously introduced; for this reason, because of the rapid changing of physics conditions which characterize the domain, only the near past is interesting; in fact the weather state in

[1] *Causality* is the relation between *causes* and *effects*; it is a concept widely studied and debated both in physics and philosophy.

[2] It is a state of the matter which includes liquids and gas; a fluid continually deforms under an external force.

the moment of making the forecast is considered as the initial state to be used inside the built model. Nevertheless, this step relies on the assumption to be able to have the data we need; in this scenario, this assumption is satisfied because of the huge presence of satellites which provide us with all the data we need about the atmosphere.

IV. **Wrap it up and forecast**: the last step is making the forecast; considering the hypotheses and the equations representing some relations of the variable chracterizing the problems, given the data acquired from the past, a model is built; then, given that this morning the weather is sunny, we are now able to say the the weather in one hour will still be sunny.

Now that it is clear the basic idea of forecasting, one question should arise from the previous example scenario; what if we do not only want to know the weather in one hour, but also in 2 hours? And what will the weather be like in the evening? And tomorrow? And the next month?
These questions lead to another concept that is crucial when facing the topic of forecasting, that is its *time-horizon*; in other words, every forecast can be done to estimate variables at different time distances from the moment in which the prediction is done and, usually, the reliability of the results decay more rapidly the more far we predict in time; for example, when talking about short term weather forecasts, they become pretty unreliable with a time-horizon of 7/10 days.
Still, there are more global patterns that can be found in a much larger time scale when talking about weather; for example we know that every Fall the precipitations in the northern emisphere will be much frequent and heavier than in Summer season.
Again, the observation of the weather forecast scenario, that is pretty simple when related to forecasts in other scenarios, shows how complex could be the topic and how it could be faced in different ways and at different levels.

1.3 Different approaches

Now that it is clear the meaning of the term *forecasting* and some observations have been made about the complexity of the topic, let us introduce some possible approaches that can be chosen to face the task of forecasting variables, depending on the scenario.

Imagine you are sitting on the couch, planning your next business travel; your company has set you a meeting in Milan for tomorrow, Tuesday, at 8:30 am, and you are planning to reach your destination by car. You remember that, in your amazing years at High School, your Physics teacher taught you

about the linear motion equation

$$s(t) = s_0 + v_0 t \tag{1.1}$$

and told you it would somehow and somewhen be helpful for you in your future; of course you didn't trust your teacher. Now however, it turns out that she was right; in fact, if you *put* inside the equation the fact that you live in Genoa, that the pure distance from Genoa to Milan is approximately 120 km and the fact that you know you'll probably drive with an average speed of 120 km/h, than the equation becomes:

$$120\text{km} = 120\text{km/h} \cdot t$$

which leads to the estimated travel time of 1 hour.

Such an approach shows how, in some hypotheses and conditions, no data related to the past are explicitly used to make a forecast, but only data representing the initial state; in the previous example, given the initial state of leaving from home at 7:00 am, the destination will be reached at 8:00 am. In fact, the equation in 1.1 represents itself the relation between space and time in certain hypotheses. However, it is necessary to make an observation related to the apparent lack of observed past data in making the forecast; even if no data is *explicitly* used in the equation to make the prediction in fact, the equation itself is the product of the application of the Galilean scientifical method, representing the formalization of the causality principle found existing between *space* and *velocity* by the mean of the observation of past empirical data; under this light, the data observation is *implicit* in the definition of the equation itself.

Coming back to your business meeting, now that you know that it'll take 1 hour to reach Milan, you are setting the alarm on the phone in order to be able to leave at 7:00 am, when you remember that you are going to use the A7 highway, which has a lot of speed cameras; this makes you realize that the previously estimated time-distance is very rough and doesn't take into account the fact that you'll have to slow down in different streets (in order to avoid sanctions) and many of them are not exactly straight; these considerations make much more difficult to compute the estimated time-distance by just one simple equation, like the one in 1.1.

This fact leads to our second consideration: depending on the precision we want our forecast to have, some assumptions could be done; they allow the choice of some models and approaches and make others not usable; in general, the more hypotheses we make and the stronger they are, the simpler the model equation becomes; in the above scenario, removing the hypotheses

of constant speed and linear distance leads to a first problem reformulation.

Going back to your alarm, after you considered the real distance to be 150 km (based on online search) and your average speed to low down to 100 km/h, you decide to set it in order to leave at 6:30 am, having estimated the time distance to be 1,5 hour; in this way you are more confident to reach your destination in time. Nevertheless, it's when you are almost sleeping that you rememeber that tomorrow is Tuesday and there is a strike scheduled by the transporters in Italy. This fact makes all of your previous estimations almost useless, being tomorrow not an ordinary day.

Such a scenario represents again, in a very simple way, the link between the reliability of a forecast and the assumptions at its basis. In particular, removing the hypotheses that assumes that *tomorrow is an ordinary day*[3], makes the previously built forecast, before reputed as reliable, totally unreliable. Still, it is to be defined the meaning of not being ordinary; differently speaking, tomorrow will be not an ordinary day because it will be Tuesday and in someway we expect the traffic to be different depending on the day of the week? Or is it because tomorrow will the transporter strike take place? Or maybe both?
While removing hypotheses, the theoretical model representing the problem grows harder and harder, becoming almost impossible to find and mathematically formulate.

At this point your head is literally exploding trying to consider all the variables to include in your model and the equations which link them all; then you remember about your Math teacher telling you that, when facing a difficult problem, changing the point of view could be the solution sometimes. At that very moment you realize that one dear friend of yours usually has the same business travel every week, on Tuesday, approximately at the same hour, since more than a year now; you call him and he tells you that it takes approximately 2 hours from Genoa to Milan by A7 highway on Tuesday morning, but they become almost 3 in the case of a strike, even if this scenario occured just twice in his experience.
With this information, you then decide to consider the worst case scenario as the expected one and eventually set the alarm in order to leave at 5:00 am.

This final approach represents a scenario in which a forecast has to be done with the necessity of a particular degree of precision and reliability, in the absence of many simplifying hypotheses; of course, it is almost impossible not to make any assumption in a real case scenario, but changing them

[3]In the sense that no extraordinary events occurs.

has a dramatic impact on the forecast[4].
In such a scenario the approach changes, leaving the need of formalizing the model equation which describes a phenomenon, for a similarity-based approach, in which we observe the past looking for similar cases to ours, and use them to forecast our future.
If from the one hand we have a purely equation based approach, on the other we do not even care about finding a model, and we assume that our scenario is approximately equal to others in the past (in the terms of the variables defining it) and that there is no reason to think that the outcome will be different, when the inputs are the same. In between these opposite approaches, there are hybrid methods in which we could still be trying to represent a phenomenon by the mean of a model algorithmically built upon the information obtained by looking at a huge quantity of past data. Some hypotheses on the mathematical form of the model are usually assumed but other than that, we often give up in finding the equation of the model in a closed form, as the price to have a numerically built model ready to be used to make forecast.
At the basis of such an approach lies statistics, which gives us the theory to represent the foundations of every data based modelization and let us introduce and formalize in our results the *uncertainty* which has to characterize, in many forms and at different levels, a forecast.
Secondly, the availability of huge quantities of reliable data is required; finally, technological means able to compute heavy algorithms in an effective and time affordable way are required as well.

Under this light, it appears clear the motivation behind the growing importance of the *Machine Learning* techniques recently; in fact if on the one hand statistical methods, more recently generally referred to with the term *Machine Learning*, are part of mathematical literature since one century or more, the technological advances let us suddenly be able to acquire and deal with huge quantities of data in an efficient way. This technological revolution brought a new fresh breath in the field and many other statistical techniques have born and avanced for this motivation in the recent past.
The following section aims at giving an overview of the Machine Learning and some mathematical and statistical techinques used to make forecasts at different levels.

We conclude this section briefly talking about the importance of the reliability of the data; in fact, it's not only important that the data is available, but also it needs to be reliable. While in theory these two contraints are always satisfied, in practice it is pretty common to work in contexts where

[4]Find an accettable tradeoff between assumed hypotheses and simplicity of the model is one of the most crucial part of the task.

the data needs to be *double checked* before even starting the modelization phase. The importance of data and the procedure to deal with it requires a specific chapter which will later follow; for now, let us consider the previous scenario and conclude by saying that, after having awaken in the very early morning, and left the house in hurry, it would be very annoying to realize that your meeting was on Wednesday and not on Tuesday.

1.4 The Machine Learning

The term *Machine Learning*, first introduced by the American computer scientist Arthur Samuel in the mid-twentieth century, represents a set of techniques based on statistical methods aimed at identifying patterns within data through a procedure that is defined as *training phase*.

In the common conception, each algorithm/technique of the aforementioned set is generally classified as belonging to one of the following categories:

- *Supervised Learning*: represents a set of algorithms, each often aimed at a single specific task, which are trained through a *trial and error* procedure (hence the term *supervised*) based on the observation of historical data.
  In particular, a mathematical model is defined thanks to a fine-tuning procedure in order to best represent the information contained in the historical data, to be then evaluated in terms of real time performance.
  Decision trees, *Artificial Neural Networks (ANN)* and *Linear regressors* represent algorithms of this type and can be applied, for example, to build forecasting trend models (e.g. revenue streams).
  Another example of supervised technique is represented by *kNN* (*k Nearest Neighbor*) even if, as we will see, this algorithm differs from others already mentioned for being non-parametric.

- *Unsupervised Learning*:: models in this category are used to identify patterns within data.
  A characteristic of this set of techniques is that the *ground truth* variable on the historical data is not present as it is unknown, and therefore it is not possible to use a trial and error training procedure, hence *Unsupervised Learning*.
  Techniques of this type are those which are generally referred to when speaking of *clustering* with which, often thanks to some prior data, patterns are sought within this in order to be able to divide it into a predetermined number of clusters.
  Example of this class of tecniques is represented by *k-Means* algorithm.

- *Reinforcement Learning*: the algorithms belonging to this class aimed at the creation of *autonomous agents* which are able to relate to the environment they are immersed in, through a training phase based on the use of penalties and rewards that reinforces or discourages specific behaviors.
  An algorithm trained to play *Go*[5] represents an example for this last category of algorithms.

Within this Thesis, given the particular contexts of application, the work will mainly deal with some techniques belonging to the first category.
It is specified that the purpose of this chapter is to introduce some mathematical techniques subsequently used, without however going into the details of each of them. For further information, please refer to [1][2].

1.5 Introduction to supervised learning

This section will provide the preliminaries and basic concepts to then be able to present some supervised learning techniques in the following parts.

1.5.1 Preliminiaries

In the *Supervised Learning* the objective is to relate input variables, which can be referred to indiscriminately in the literature with the term *features* or that of *predictors*, with one or multiple output variables. As mentioned, this relationship is inferred thanks to a *training* or *fitting* procedure in which a fine tune of a particular model is carried out thanks to the historical data, minimizing a certain error function, usually named *Loss function*.
Through this procedure it is necessary to pay attention on how well the constructed model represents the historical data. In other words, a model that manages to achieve extreme accuracy on the historical data will hardly be able to adequately generalize about the future, leading to poor performance; this phenomenon takes the name of *overfitting*.
The phenomenon, characterizing the opposite scenario in which a model exploits historical data so little that it is too generic, takes the name of *underfitting* and still the performances of such built model would be insufficient.
The key is therefore represented by the identification of an appropriate trade-off between the two scenarios, different for each application context.

Going back to talking about the output variables, each of these can have a quantitative or qualitative nature; in the latter case these variables are defined in this context as *categorical*.
For the problems in which the output is quantitative we speak of *regression*,

[5] *Go* is a two-player strategy board game with origins more than 2500 years back in China. Despite very simple rules, the game is very complex strategically.

while for those in which it is qualitative we speak of *classification*.
An example for a regression problem is the construction of a forecast model for the EUR/USD exchange rate.
Profiling a customer is instead an example of a classification problem and is the practice of classifying customers behaviour, using historical data, in order to decide which market strategy pursue, for example choosing which product to show them by ADVs.

1.5.2 Statistical Learning Theory

Before presenting some supervised learning techniques, an introductory formalization of some concepts related to statistical learning is provided.

Let us introduce the *input space* X defined by the so called predictors; in our and in many other practical applications X is a vector space of the form $X \subseteq \mathbb{R}^D$, but depending on the scenario it could be the space of the probability distributions on a generic probability space Ω, a collection of graphs or, given an alphabet Σ, it could be the finite space of strings of p letters $X = \Sigma^p$, with $p \in \mathbb{N}$. Be then Y the *output space*; again, depending on the problem the nature of the set Y can vary: $Y \subseteq \mathbb{R}^B$ in a *regression* problem, in which if $B > 1$ it becomes a *multivariate regression* one; on the other hand when $Y = \{Y_i\}_{i=1:T}$ we talk of a *classification* problem, with the particular case of $T = 2$ that takes the name of *binary classification* and Y is commonly represented by $Y = \{1, -1\}$.
The purpose of supervised learning is to infer the relationship that exists between input and output by defining a model f such that

$$f(x_{\text{obs}}) \sim y_{\text{obs}}$$

with $x_{\text{obs}} \in X$ e $y_{\text{obs}} \in Y$, where $X \times Y$ takes the name of *data space*.
The *training set* is defined as a set consisting of N input-output samples

$$S = \{(x_1, y_1) \dots (x_N, y_N)\}$$

with $x_i \in X$ e $y_i \in Y$.
The objective of a supervised model is to define a possible training set[6] and find a model that *learns* the *best* input-output relationship for the chosen training set.
It remains clear that it is therefore necessary to assume the existence of such a theoretical model that takes into account the *uncertainty* intrinsic to the reliability of the data.
For this reason it is postulated the existence of an unknown distribution

[6]It is specified that depending on the methodologies chosen, it is possible and often indicated to choose different subsets of the historical data to represent different training sets; these procedures are often used to limit the overfitting problem.

$p(x, y)$ according to which the data is i.i.d. [1]

By the following factoring

$$p(x, y) = p_X(x)p(y|x)$$

the different types of uncertainty modeled by the distribution are highlighted. Indeed:

- $p(y|x)$: represents the nondeterministic relationship between input x and output y.
- $p_X(x)$: represents the sampling uncertainty from the input space X.

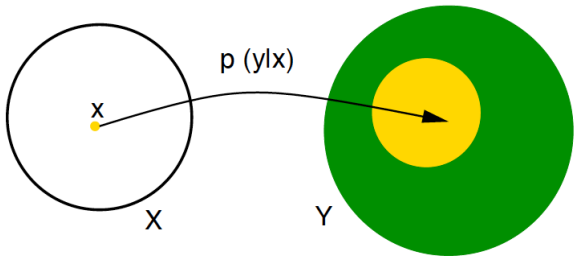


Figure 1.1: The green zone represents all the possible outputs, while the yellow zone represents the outputs obtainable from the input x.

1.5.3 Loss function & expected risk

Let us consider the sample $(x_j, y_j) \in X \times Y$ and $\hat{f} \in \mathcal{H}$ as a possible operator approximating the desired input-output relation f, in which $\mathcal{H}$ takes the name of *hypoteses space*; depending on the choice of $\mathcal{H}$ many different $\hat{f}$ can be obtained for the very same problem, forcing the solution to be bound to some constraints (an example of this is constituted by a *linear regression* in which, as it'll be soon clear, $\hat{f}$ is forced to be linear) or others.
Then the error committed by this operator can be measured by a suitable *Loss function*

$$L : Y \times Y \to [0, +\infty)$$

defined by the model requirements; in particular the error committed will be represented by

$$l(y_j, \hat{f}(x_j))$$

Given a loss function L is denoted by f^* the objective function which represents the best approximation of the operator that relates input and output. Specifically, we have:

$$f^* = \arg\min_f \mathbb{E}[l(y, f(x))] = \arg\min_f \int p(x, y)l(y, f(x))dxdy$$

where the operator $f^* : X \to Y$ minimizes the expected risk $\mathbb{E}[L(y, f(x))]$. We can therefore say that the goal of a supervised learning approach is to find a $\hat{f}$ as similar as possible to f^* and that, therefore, generalize well, that is, that it behaves well on both historical and new samples.

1.6 Linear models

Also known as *regression models*, they represent one of the pillars of statistical data analysis. Introduced for the first time by Legendre and Gauss at the beginning of the XIX century, they received a great boost from the 1970s with the advent of the first computers which allowed to drastically reduce data processing times.
As the name itself suggests, they are based on the assumption that the relationship between predictor variables and output variable is, in fact, linear, or equivalently $\mathcal{H} = \{f | f \text{ is linear}\}$[7].

Following [2], the array $\hat{\mathbf{x}} \in X$ of the input space of p features (i.e. $X = \mathbb{R}^p$) can be defined and y the corresponding output variable[8]. The following expression for the output variable can be then written

$$y = \hat{\boldsymbol{\beta}} \cdot \hat{\mathbf{x}} + \hat{\beta}_0 \tag{1.2}$$

in which $\hat{\boldsymbol{\beta}}$ represents the weights of the linear combination and $\hat{\beta}_0$ the intercept or, using ML terminology, the *bias*.
Defining then the following arrays

$$\begin{aligned} \mathbf{x} &= (\hat{\mathbf{x}}, 1) \\ \boldsymbol{\beta} &= (\hat{\boldsymbol{\beta}}, \hat{\beta}_0) \end{aligned} \tag{1.3}$$

a more compact version of the equation (1.2) is obtained

$$y = f(\mathbf{X}) = \boldsymbol{\beta} \cdot \mathbf{X} \tag{1.4}$$

Switching to the matrix notation the *Residual Sum of Squares (RSS)* are defined as

$$RSS(\beta) = \|\mathbf{y} - \mathbf{X}\beta\|_2^2 \tag{1.5}$$

which is the variable to minimize in order to *fit* the model.
This approach is known as *Least Squares* and the solution β is found as the

[7] This assumption can be quite *strong* and simplistic in some application contexts; on the other hand, the simplicity of the model and the interpretability of the result make it a widely used method, even only in the early exploratory stages of a problem.

[8] For easiness it'll be assumed to deal only with scalar y but almost exactly the same discussion stands for multivariate ones.

array which minimizes (1.5) as follows

$$\boldsymbol{\beta}^* = \arg\min_{\beta} \mathrm{RSS}(\beta) = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - \beta \cdot \mathbf{x}_i)^2 \tag{1.6}$$

in which N is the number of samples in the training set.
Differentiating on $\boldsymbol{\beta}$ leads to the so called *normal equation*

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X} \cdot \beta) = 0 \tag{1.7}$$

where $\mathbf{X}$ is an $N \times p$ matrix representing the samples in the training set and $\mathbf{y}$ is the N-dimensional vector of the corresponding outputs.
the quadratic form of (1.6) in respect to β leads, when $\mathbf{X}^T\mathbf{X}$ is non singular, to the unique minimum of the form

$$\boldsymbol{\beta}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{1.8}$$

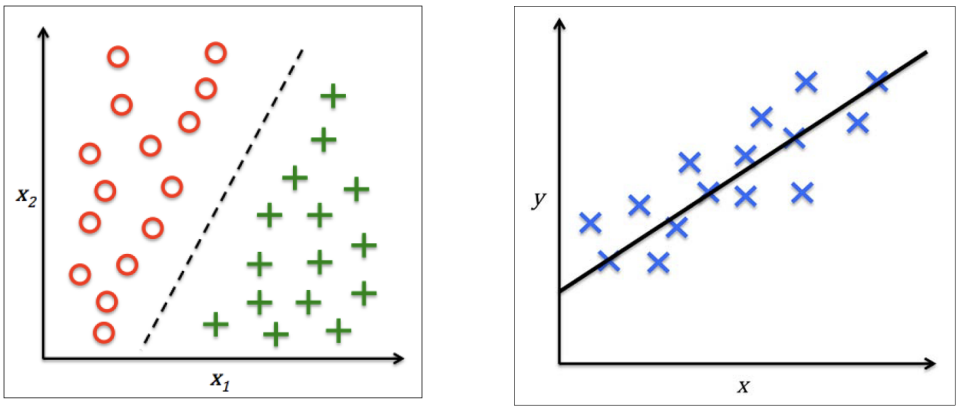


Figure 1.2: Examples of linear regression for classification (left) and for continous output (right).

1.6.1 Regularized least squares

In order to control overfitting and build a model able to generalize well, a slightly different version of the model is defined as follows

$$\arg\min_{\beta} \left(\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \|\beta\|_2^2 \right) \tag{1.9}$$

in which the scalar parameter λ controls the regularization. This model is known as *Tikhonov-regularized least squares* or *Ridge regression.*
Following a similar approach to the previous one, the solution can be written as

$$\boldsymbol{\beta}^*_{Reg} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{1.10}$$

where $\mathbf{I}$ is the $p \times p$ identity matrix.
Notice that, in the case of $\lambda = 0$, eq. (1.10) becomes eq. (1.8) i.e. the non regularized one.

1.6.2 Beyond linearity

More generally, the *Tikhonov-regularized* problem can be defined and the solution can be computed as

$$\arg\min_{\beta \in \mathbb{R}^p} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f_\beta(x_i)) + \lambda \|\beta\|^2 \tag{1.11}$$

being ℓ the chosen loss function.
The discussion above follows if it is assumed $f_\beta(x)$ to be linear, i.e.

$$f_\beta(x) = \sum_{j=1}^{p} \beta_j x_j.$$

It can be now introduced the concept of *feature map* as an opportune operator

$$\Theta : X \to F$$

from input space X to a new space F (possibly infinite), called *feature space*, in which a scalar product, denoted with the notation $\langle \Theta(x_1), \Theta(x_2) \rangle_F$ where $x_1, x_2 \in X$, exists.
For simplicity, in order to provide the intuition of the importance of this kind of formalization, let us assume $F = \mathbb{R}^k$;then

$$f_\beta(x) = \sum_{j=1}^{p} \beta_j \Theta(x)_j \tag{1.12}$$

With this simple trick, linear relations in the feature space can be found even if the relations in the input space was not linear at all.
For further details please see [1].

1.7 Nearest Neighbors methods

1.7.1 1-Nearest Neighbor

Another simple yet, in many contexts, useful and efficient class of methods, other than linear regression, is represented by a class of local and non parametric algorithms called *Nearest Neighbors*.
They belong to the class of the, so called, *memory based* algorithms which don't require a model to be fit and are based on the assumption that similar points in the input space *should* behave similarly in the output space, i.e. it stands the principle of continous dependence from data.

Formalizing, called

$$T = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

the training set, if considering an input $\hat{x}$, follows

$$\hat{i} = \underset{i=1,\dots,n}{\arg\min}\, \mathcal{D}(\hat{x}, x_i) \tag{1.13}$$

i.e. the index of the closest sample point in the training set to the new input $\hat{x}$ in respect to the metric $\mathcal{D}$.
For example, in the case of regression, it is very common the choice of the Euclidean metric and in this case eq. (1.13) becomes

$$\hat{i} = \underset{i=1,\dots,n}{\arg\min}\, \|\hat{x} - x_i\|^2.$$

Using this formalization, the *Nearest Neighbor* estimator can be defined as

$$f(\hat{x}) = f(x_{\hat{i}}) = y_{\hat{i}}$$

that is, in words, defining the output of a new sample equal to the output of the closest (in respect to the chosen comparison metric) input to the new one in the training set.

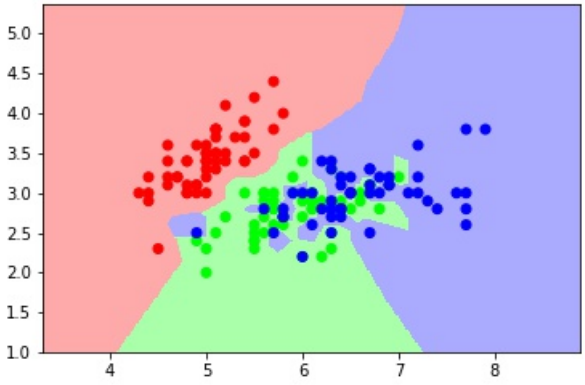


Figure 1.3: A 3-class classification example performed on Iris dataset using a 1-NN model.

1.7.2 K-Nearest Neighbors (KNN)

A slightly different approach could then be choosing not only one point in the training set to estimate the output (as it happens in the *NN* estimator), but using k data points.
In this perspective, again calling $\hat{x}$ a new input sample, with the notation

$$\mathcal{D}^i_{\hat{x}} := \mathcal{D}(\hat{x}, x_i) \tag{1.14}$$

then

$$\boldsymbol{\mathcal{D}}_{\hat{x}} = \left\{\mathcal{D}^i_{\hat{x}}\right\}_{i\in\{1,\dots,N\}} \tag{1.15}$$

is the ordered array (in an ascendent order) collecting all the distances (in terms of the chosen metric $\mathcal{D}$) of the new input sample point $\hat{x}$ from all the others in the training sample.
Defining as

$$\mathcal{I}_{\hat{x}} := (\hat{i}_1, \dots, \hat{i}_N) \tag{1.16}$$

the sorted array containing the indices of (1.15) the set

$$\mathcal{K}_{\hat{x}} = \left\{\hat{i}_1, \dots, \hat{i}_K\right\}$$

collects the first K indices of the array in (1.16).

Follows that

$$\mathcal{D}^{\hat{i}_1}_{\hat{x}} \leq \mathcal{D}^{\hat{i}_2}_{\hat{x}} \leq \dots \mathcal{D}^{\hat{i}_K}_{\hat{x}}.$$

Finally, the *K-Nearest Neighbor* estimator is defined as

$$f(\hat{x}) = G((y_{\hat{i}_1}, \dots, y_{\hat{i}_K})) \tag{1.17}$$

where G is an aggregation functional, with the notation $y_{\hat{i}_j} = f(x_{\hat{i}_j})$.
For example, chosing G as a weighted average function it follows

$$f(\hat{x}) = \frac{\sum_{j=1}^{K} \mathcal{D}_{\hat{x}}^{\hat{i}_j} y_{\hat{i}_j}}{\sum_{j=1}^{K} \mathcal{D}_{\hat{x}}^{\hat{i}_j}} \tag{1.18}$$

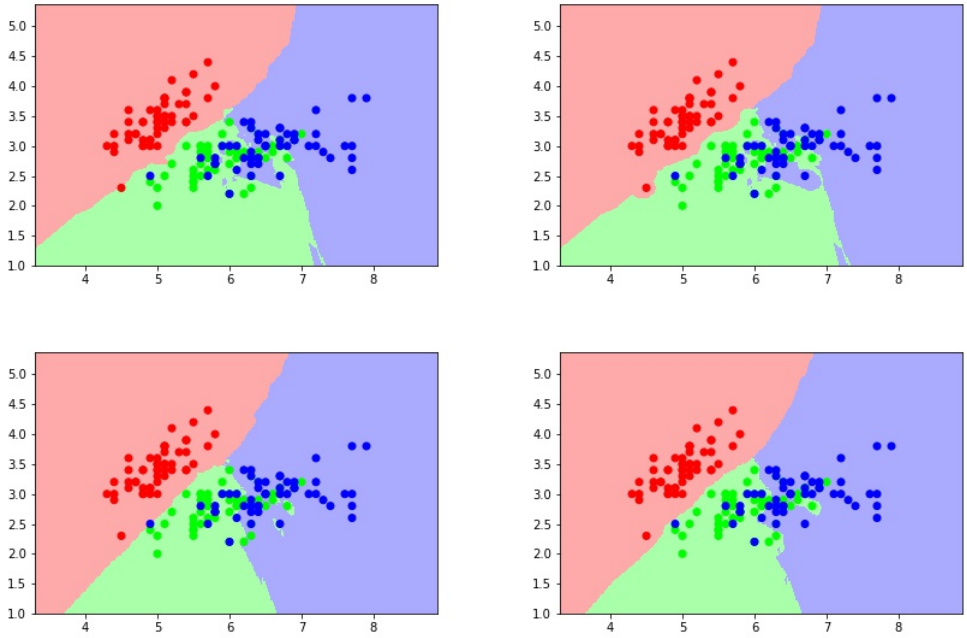


Figure 1.4: Examples of 3-class classification on Iris dataset using kNN:
(top-left) 5 neighbors with uniform weights.
(top-right) 5 neighbors weighted on distances.
(bottom-left) 15 neighbors with uniform weights.
(bottom-right) 15 neighbors weighted on distances.

1.7.3 Curse of dimensionality

As already said this class of methods, despite their conceptual simplicity, it is often very efficient.
This fact breaks down when working with a high dimensional space where this class of algorithm starts to perform poorly; this phenomenon is famous as the *curse of dimensionality* (term coined by Richard Bellman in [4]) and is related to the concept of neighbors this class of methods stands upon.
Without going in details (to have more see [2]) and with the aim to give the intuition of the problem, let us consider a unit cube in a p-dimensional

space.
Suppose we want to find a hypercubical neighborhood of a point in this hypercube to capture a r percentage of the volume of the global hypercube. How long should it be the edge e of this neighborhood hypercube?
The answer is provided by the formula

$$e_p(r) = \sqrt[p]{r} \tag{1.19}$$

Supposing the goal is to find the 10% of the total volume. Using (1.19) it can be observed that in 2 dimensions the edge e should be long $\sqrt[2]{0.1} = 0.31$, while in 3 it should be $\sqrt[3]{0.1} = 0.46$. Growing the dimension it is clear how e needs to be enlarged to find the same percentage of global hypercube reaching, in the case of 100 dimensions, the value of $\sqrt[100]{0.1} = 0.97$.

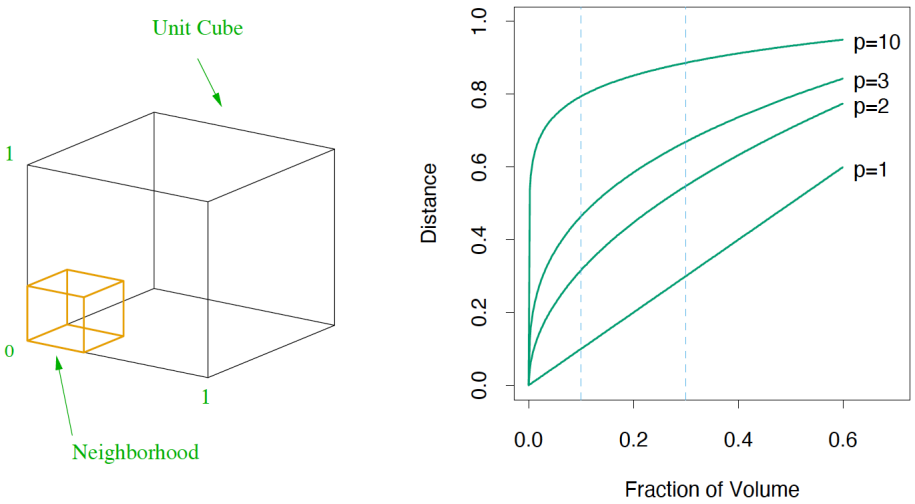


Figure 1.5: In this figure it is shown how the length of the edge e (*Distances*) and the fraction of volume r are linked as the dimension of the space p changes.

This simple example shows what the problem is and how the concept of neighborhood changes when in high dimension.

1.8 Tree-based methods

1.8.1 Introduction

In this section a very popular class of methods will be introduced; based on the concept of decision tree, they represent a widely used class of methods in the Machine Learning community and in the business world.

The basic idea constituting the foundations of the final methodology has been introduced for the first time in 1963 by Morgan and Sonquist with their *Automatic Interaction Detector (AID)*; it started to evolve and change in the years until it reaches the maturity with formalization of *CART* by *Breiman et al.* [5] in 1984.
The most important features of this class of methods are that:

a. They are extremely easy to interpret; this feature has made this class of algorithms very popular in business applications where it is crucial that even non technical people can clearly understand what is being pursued.

b. They can deal with numerical and categorical variables; in fact, if for many methods the introduction of categorical variable is pretty hard to manage, with trees this feature comes free because of the method itself.

c. They can model arbitrarily complex and non linear relations in data.

The further step, starting from the concept of decision tree, is represented by what is called *Random Forest*, a set of enseble methods which are based on the concept of decision trees.
For simplicity, what follows is the formalization in the case of a classification problem; the case of regression can be derived from the classification scenario (for details please see [6]).

1.8.2 Decision trees

In a classification problem, the output space is the collection of all possible outputs of the model $Y = \{c_1, c_2, \dots, c_k\}$. With this formalization, such a problem can be seen as finding a model ϕ which partitions the input space as

$$X = X^{\phi}_{c_1} \cup \cdots \cup X^{\phi}_{c_k} \tag{1.20}$$

where $X^{\phi}_{c_i} = \{x \in X | \phi(x) = c_i\}$.

Before moving on, a *rooted tree* is defined as a directional graph in which every pair of vertices is connected by only one path and there exists a node that is designated as root, which every edge moves away from. The terminal

nodes of it are usually called *leaves*.
When all the nodes (but the terminal ones) have exactly two splits, the tree is called *binary*.
In this framework, a *decision tree* model $T : X \to Y$ is a rooted tree (usually binary) which splits the input space X sequentially into subspaces until the partition in (1.20) is reached.

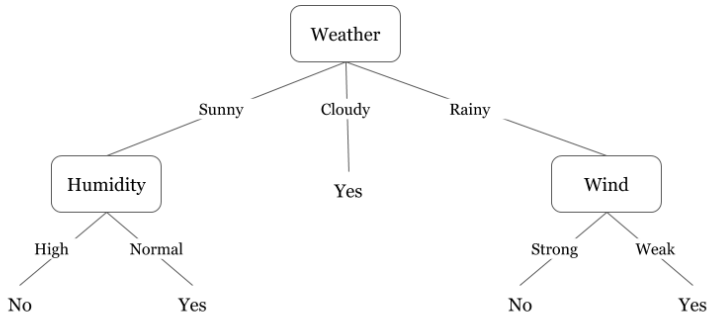
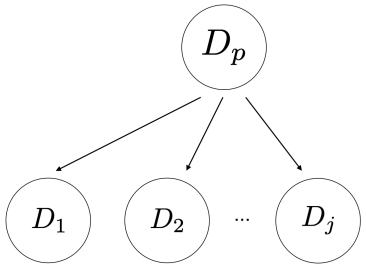


Figure 1.6: Example of an extremely simple decision tree.

Now that it is clear what a decision tree is, what remains is to understand how the training process works.

Starting from the root node t_0 it starts an iterative process which involves every parent node t_p and the choice of a splitting criterion s in order to maximize the so called *Information Gain* function at every parent node, defined as



$$IG(D_{t_p}, s) := I(D_{t_p}) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_{t_j}), \tag{1.21}$$

where D_{t_p} represents the data in the parent node t_p, D_{t_j} the one in t_j-th node, N_p and N_j the number of samples belonging to D_{t_p} and D_{t_j} respectively.
The function I is known in literature as *Impurity measure* and is the one which tests the *goodness* of the split.

In the case of a binary tree, eq. (1.21) becomes

$$IG(D_{t_p}, f) = I(D_{t_p}) - \frac{N_{\text{left}}}{N_p} I(D_{\text{left}}) - \frac{N_{\text{right}}}{N_p} I(D_{\text{right}}).$$

where *left* and *right* labels denote the two partitions obtained from the parent node.
In this framework, usually the most common possibilities for the choice of the Impurity function are the following:

- *Gini index*
$$I := I_{\text{G}}(t_j) = 1 - \sum_{i=1}^{c} p(i|t_j)^2.$$
- *Entropy*
$$I := I_{\text{H}}(t_j) \ = -\sum_{i=1}^{c} p(i|t_j) \cdot \log_2 p(i|t_j).$$
- *Misclassification error*
$$I := I_{\text{E}}(t_j) \quad = 1 - \max_i \{p(i|t_j)\}.$$

where $p(i|t_j)$ represents the probability of a sample in the t_j-th node of belonging to i-th class over the total of c classes.

Feature Importance

Introduced for the first time by Breiman et al. in 1984, *Feature* or *Variable importance* is, in the context of decision trees, a metric with the purpose to evaluate how much every predictor used by the tree is important in order to discriminate in the model itself or, in other words, how much predictive power it has with respect to the ground truth.
Defining X_j the j-th variable and being φ the decision tree model, feature importance of φ is defined as

$$\text{Imp}(Xj) = \sum_{t \in \varphi} \Delta \text{I}(\tilde{s}_t^j, t) \tag{1.22}$$

where $\tilde{s}_t^j$ is the best surrogate split for s_t, that is the closest split defined on variable X_j that can mimic the actual split defined at node t. For more details see [5].

Dealing with overfitting

From an intuitive point of view, it is natural to think that the deeper the tree, the better the results (i.e. the less the residual error on training set). This fact is supported by many results (see Section 3 in [6] for more details) but the problem here is that, even if the error on training set is almost zero, that doesn't mean that the results of the algorithm in the general case on new samples will also be very good; on the contrary it is very common to fall in the case of an *overfitted* model in which the noise in the data is captured by the leaves of the tree.
In order to deal with this problem the stopping criterion in the training procedure is usually defined by some heuristics that are based on the fine tuning of many hyper parameters; the most common are the following:

- β: represents a threshold used to control the decrease stepwise of the impurity and set as leaf node every node in which the decrease is less than it.
- d: it is the maximum depth reachable during the training of the tree model. In the libraries `max_depth`.
- L_s: it is the minimum number of samples node have to contain in order to be a leaf node. In some data science libraries it is called `min_samples_split`.
- S_s: if used, it represents the minimum number of elements every node from a split has to contain; if this condition is not verified, the node that generated the split is set as a leaf node. In the libraries `min_samples_leaf`.

In order to tune these (and possibly other) parameters different procedures can be pursued such as an extensive *Grid search* or a *Bayesian search.*

The tuning of the parameters above can help to prevent overfitting in the training procedure of a decision tree.
On the other hand, there is another method that could be useful, often if it follows an optimization of the yet introduced parameters, for dealing with overfitting and takes place at model *trained*; it is called *pruning* and this name comes from the fact it basically consists in pruning some of the *branches* (this is how are called the edges in a decision tree) of the tree when built.
This procedure leads to a leaner structure with the removal of less informative splits and edges and can be perform *ex post* compared to the training procedure.

While these techniques can improve the results of a model smoothing the problem of overfitting, decision trees fall in the category of the **weak learners** because of their propension to high variance due to the presence of noise in the data.
From this fact borns the idea of *ensemble methods* which consists in combining different weak learners to build a new model, based on the others but more stable and less sensitive to noise; such a model would fall in the category of the **strong learners**. In the next sessions some models belonging to this categorywill be introduced.

1.8.3 Ensemble learners

As anticipated, this class of models is based on the simple idea of combining more models which may suffer of instability problem with the aim of the creation of a more stable one.
This goal can be pursued by using various approaches and implementing different logics; the following two sections represent an introduction to some of the most famous ones.
The principle at the basis of the following methods is the idea of *bagging* or *bootstrap*.
This technique is very suitable for decision trees that suffer of high-variance; in fact, it has the goal of reducing the variance of an estimated prediction function by building K models, each one on a different training set, and averaging the results[9].
In formula, calling $\Phi_\lambda^{X_i}(x)$ an opportune model (possibly dependent from a set of parameters λ), the new model based on bagging technique is defined as

$$\hat{f}_K(x) = \frac{1}{K}\sum_{i=1}^{K} \Phi_\lambda^{X_i}(x) \tag{1.23}$$

where K is the number of models averaged and X_i represents the i-th training set bootstrapped from the global one which the model $\Phi_\lambda^{X_i}(x)$ is built upon.
With this approach every model is *i.d.* (identically distributed) and averaging the results should lead to better results with less variance.

Random Forest

The algorithm, first introduced in 1995 by Tin Ka Ho [7], takes its name from the fact that it could be visualized by a set of decision trees, a *forest* indeed, in which another grade of *randomness* aimed to variance reduction is introduced.

[9]This is to give the intuition; in practice it is an average for a regression model, while is a majority vote in a classification scenario.

With bagging each tree is i.d. and the expectation of an average of K trees is the same of anyone of them. More, averaging K i.i.d. random variables, each one having variance σ^2, we obtain another variable with variance $\frac{1}{K}\sigma^2$. If the variables are simply i.d. with positive pairwise correlation ρ, the variance of the average is

$$\rho\sigma^2 + \frac{1-\rho}{K}\sigma^2. \tag{1.24}$$

From eq. (1.24) it is clear that growing K let the second term disappear but has no effect on first term that depends on the basic variance σ^2 and on the correlation ρ that could partially smooth the benefit of averaging.
Made this consideration, the idea of random forest is to act on correlation factor ρ in order to reduce it without changing the variance σ^2 too much. This result is achieved by adding, at every split inside the building procedure of every decision tree of the forest, a random selection of m variables among the total p variables of the bootstrapped data.
In this scenario the eq. (1.23) assumes the form

$$\hat{f}_K^{\mathrm{RF}}(x) = \frac{1}{K}\sum_{i=1}^{K} T^{\Theta_i}(x)$$

where $T^{\Theta_i}(x)$ represents the tree in the forest, defined on the i-th bootstrapped training set and characterized by specific samples of m features at each inside node.
Being built on decision trees, random forest algorithm inherits all their hyperparameters to tune when in training phase; more, there are to tune two other very important parameters:

- K: represents the number of decision tree that will form the forest. We showed that the bigger the value, the smaller the value in (1.24). In the libraries it is called `n_estimators`.
- m: represents the number of features to sample from the total inside every node in every tree; common choices for this parameter, being p the total number of features of the data, are
  - $m = \sqrt{p}$
  - $m = \log_2 p$

  The tune of this parameter is useful to low the correlation between trees.

It has to be reminded that, if there are just a few meaningful features F among a much bigger total, if m is chosen too small the performance will be poor.
In fact the probability to extract exactly the F features on the total p is

$$Prob = \frac{1}{\binom{p}{F}}$$

and, assuming that $F \leq m \ll p$, it is clear that *Prob* is likely to be very small; in this scenario the random forest will be built with trees that won't easily capture the information in the data and this will lead to poor results.

Boosted decision trees

Another popular class of ensemble methods is the one based on the concept of *boosting* the search of the optimal estimator.
One of the first method of the category was *AdaBoost* algorithm introduced by Freund and Schapire [9] in the mid 90's.

As presented in the section above, the idea behind random forests is to build a set of independent trees to reduce the high-variance which affects decision trees; on the contrary, the idea of boosting is to build a set of dependent sequential trees in which each one has the goal to minimize residual error of the precedent one.

In general, given a training set $\{(x_i, y_i)\}_{i=1}^N$ of N elements, we can define a boosted algorithm as a sequence of dependent *weak* models f_i where usually the first term is defined as

$$f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} \mathcal{L}(y_i, \gamma) \tag{1.25}$$

where $\mathcal{L}(y, f(x))$ represents the chosen *Loss* function with the constraint to be differentiable.
With f_0 obtained, the so called *pseudo-residuals* are computed

$$r_{i1} = -\left[\frac{\partial \mathcal{L}(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_0(x)} \quad \text{for } i = 1, \ldots, N. \tag{1.26}$$

Now a *decision tree* model[10] $T_1^{\Theta}(x)$ is fitted on the *residual* training set $\{(x_i, r_{i1})\}_{i=1}^N$ and, finally we define the estimator at the next step of the sequence as

$$f_1(x) = f_0(x) + \gamma_1 T_1^{\Theta}(x) \tag{1.27}$$

in which the factor γ_1 is found as

$$\gamma_1 = \arg\min_{\gamma} \mathcal{L}(x_i, f_0(x_i) + \gamma T_1^{\Theta}(x_i)) \tag{1.28}$$

by a gradient based minimization technique.

[10] The discussion holds for any *weak* learner.

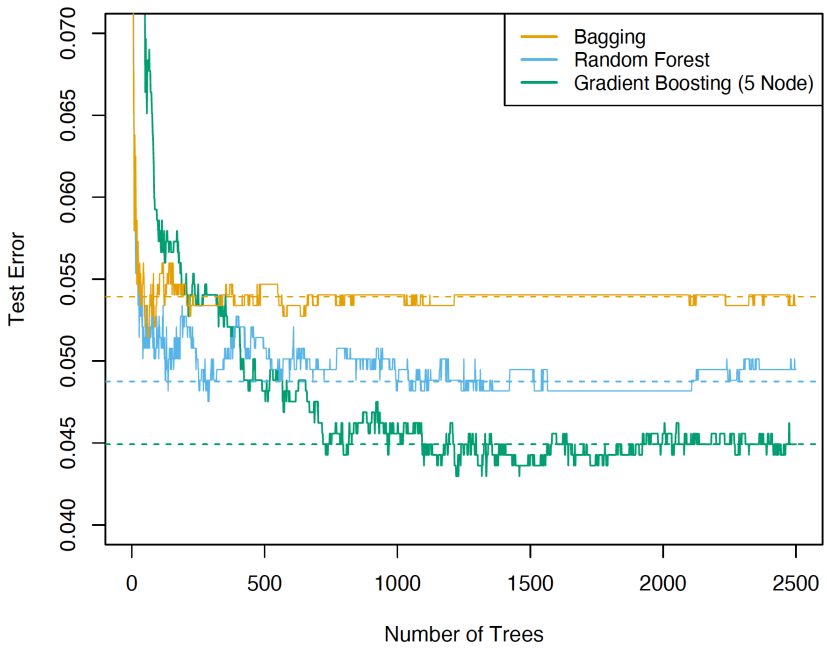


Figure 1.7: A comparison between Random Forest and Gradient Boosting performances in a SPAM classification problem (source: [2]).

1.9 Artificial Neural Networks

1.9.1 Introduction

With the increase of computational power in the last decades, this architecture has probably become the most famous and utilized among all the others nowadays.
Historically however, it is common to identify in the 1958 the year in which was formalized the first neural newtwork in the history, named *perceptron* [46].

As the name itself suggests, the idea that drove the construction of such an architecture is founded on emulating how biological (human) neural networks work.
Without the will of deepen in, what follows aims to give the reader a mere overview of what a biological neural network is, in order to comprehend how an artificial one is constructed and how similar it is to its biological twin; for some more details see [49] and [36].

In biology, a *neural network* is a very complex network made of connections of specific cells, called *neurons*, in which electric signals constantly flow to interact with the whole organism.
As said, the unity of a neural system is the *neuron*, which is constituted by a central body and various ramifications exiting and entering in it. The *dendrites* collect signals incoming from other neurons while a longer ramification, called *axon*, is responsible for sending the impulse to other neurons thanks to its terminal arborizations.
During the process, if the external electric spikes or the signal deriving from them is too low, the impulse is suppressed.

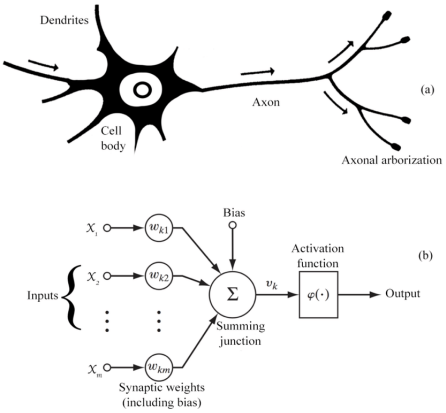


Figure 1.8: Similarity between biological and artificial neural networks (fig. a: [40]; fig. b: [33].)

Trying to mimic this architecture, a neuron in an *Artificial Neural Network (ANN)* is a unity constituted by some *synaptic weights* w_{ki} (the dendrites), a *summing function* with some implicit unknown bias representing the cell body, an *activation function* φ which represents the axon with its threshold for the strength of the signal v_k, and the output/s y_k, mimicking the final arborization/s of the biological neuron.
Generally, an Artificial Neural Network (ANN) is a combination of many different artificial neurons (that may be linked in different ways) built following a training procedure on historical/training data.

While similar in the goal and for many different aspects, artificial and biological networks are however very different under many lights because of the extreme complexity of the biological one, which is impossible (at least today) to reproduce.
The most evident and impacting differences are:

- **Size**: Altough they are still estimates, the number of neurons in the brain is estimated to be higher than 120 billions [50] with more than 1000 trillions of synapses [52]; with such numbers, it is clear that, even with the *state of the art* technology today it is impossible to reach not even close this magnitude.

- **Topology**: Human neural network is a very complex net of neurons, linked together by connections which activate in an asynchronous way and in which signals flow without a privileged direction, in a way that is far too complex to be modeled.
  On the other hand, artificial networks can be clustered into two different categories, namely *feedforward* and *recursive* ones.[11]
  While the networks belonging to the first category are organized in sequential layers of neurons in which the signal flows only *forward* and every layer is directly connected only to its neighbours, the latter category contains networks, like *Long Short Term Memory (LSTM)* architecture, in which the signal can flow in both directions, even though only in the way the architecture itself let.
  In both scenario however, it is evident that the architectures are pretty different, in term of complexity, from the biological networks.

- **Power consumption**: On average, human brain consumes approximately 20% of the body's energy [30] estimated, in a typical adult, to be around 100W [41], which corresponds to a consumption of 20W. If we compare this number with the average wattage needed to light a bulb, that is between 15 and 60W depending on the bulb itself, it can

[11] A lot of research work is being done in the field and this clusterization refers to the most classic and consolidated architectures at the moment.

be concluded that, despite its complexity, the human brain is impressively efficient.
On the other hand, the most sophisticated and powerful artificial networks usually need a *GPU (Graphic Processing Unit)* to train and run on; this particular hardware has an average consumption of more than 100W for the less powerful units, touching peaks of more than 250W for the most recent and performing ones.

- **Learning process**: When talking about how the human brain learns and how it stores the information, the unknowns are still infinite. What we call *learning* is the process of adding information, often finalized to a goal, above all the other information we stored in the past which are almost always used as basis in the learning process. During the human life cycle the brain grows, in terms of neurons and connections, and change in its morphology.
  This simple set of facts, that does not reflect the real (and mostly yet to uncover) complexity of the learning phenomena, shows the difference from every artificial neural network that can be built today in which is called *training phase*, as will be clearer later, the process of optimizing weigths within a static architecture by minimizing a specific operator.

1.9.2 Types of Neural Networks

As previously anticipated, the term *Neural Network* usually refers to some architecture based on the concepts of neurons, layers and activation functions.
Starting from these entities, different architectures have been formalized and implented through the years due to the many different scenario of application of these models.
Generally speaking, the most popular artificial neural networks can be splitted into different categories:

- **Feedforward Networks** In this type of network, neurons are arranged in *layers* with an input layer, which represents the input data, and an output layer, representing the outputs of the net. All the other layers are called *hidden* because they have no connection with the outer world.
  In this kind of network, every neuron in one layer is connected with each node in the next layer (for this reason they are also called *fully connected* layers) but there are no connections within the same layer.
  Finally, the links between the neurons are one directional only. For more details see[11].
  It has to be remembered that working computational cost grows with the number of layers and neurons in it while, usually, improving in

performances.

When there are not hidden layers this net takes the name of *simple perceptron*; the first neural network in the history has been of this kind. On the opposite, when the number of hidden layers and neurons in each of them is very high this architecture is called *Deep Neural Network*. Variations of this architecture are represented by the so called *Autoencoders* (see [11] and [53]) very useful for *Feauture Extraction*.

The field of application of such an architecture is mostly related to *Predictive Analysis* with *Pattern Recognition* and some *Computer Vision*.

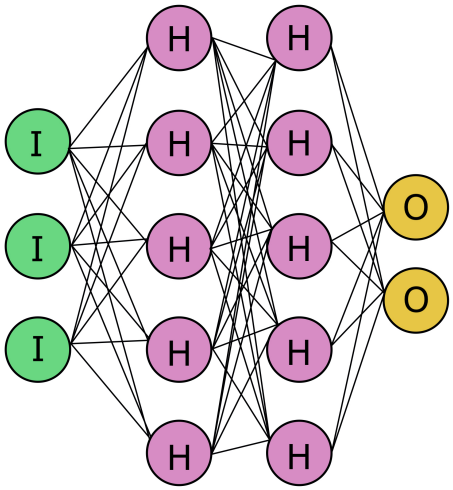


Figure 1.9: The architecture of a feedforward neural network with 2 hidden layers, with 5 neurons each, and 2 output neurons.

- **Recursive Networks** This kind of networks allows connection not only between sequential neurons but from multiple direction and not only forward.

  Examples of this kind of networks are *Recursive Neural Networks (RNN)* or *Long Short Term Memort (LSTM)* (see [11] and [55]).

  This kind of networks is used whenever an ordering relation between the inputs wants to be kept into account when generating outputs; examples are *Time series forecasting*, *Speech analysis* and *Music generation*.

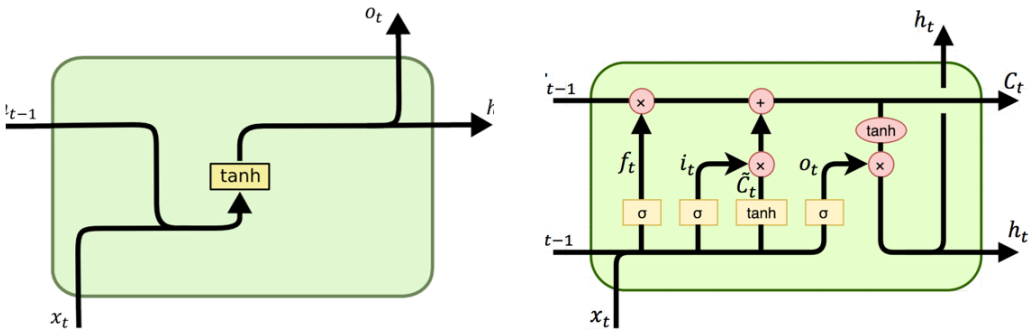


Figure 1.10: On the left, a unit in a RNN; on the right a unit in a LSTM.

- **Convolutional Networks** Also known as shift invariant artificial neural networks, because of the translation invariance characteristic that add to feedforward nets, they represent a variation of this type of network in which some convolutional blocks are added before a series of fully connected layers. Between convolutional layers some *pooling* procedure to reduce the size of the data is usually performed. To get more details see [11] and [10].

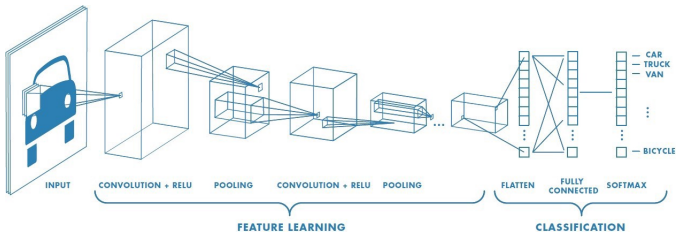


Figure 1.11: An example of convolutional neural network used in an image classification task.

Belonging to this class of networks are *Generative Adversarial Networks (GAN)* (see [51]) used to generate content based on previously learned features.

Convolutional networks are mostly used in *Computer Vision* to classify images, perform *Object Detection* tasks and some *Natural Language Processing (NLP)*.

In the next sections some efforts will be spent only on giving the idea of how a feedforward network works, being this type of architecture the one utilized in the work.

1.9.3 Feedforward Artificial Neural Network

As previously said, the fundamental unit in such a network, also called *Multi Layer Perceptron (MLP)*, is constituted by the *neuron*. From a mathematical point of view, it represents a node of a graph (the network) in which information from the nodes of the previous layer is collected, combined, and finally the result of this elaboration is given to the nodes in the following layer. Let us now focus on how the information flows through the neuron ([46][48]).

Let us consider the first layer after the one of *inputs* in an artificial network and the generic k-th neuron in it. What follows stands for every generic neuron in the network.
Be $X^p = (x_1^p, \dots, x_N^p)$ the p-th sample in the training set where N is the total number of features of every sample. Let us then define the signal threshold variable as x_0, that can be seen as representing the bias. It can be then defined $\hat{X}^p = (x_0, x_1^p, \dots, x_N^p)$.

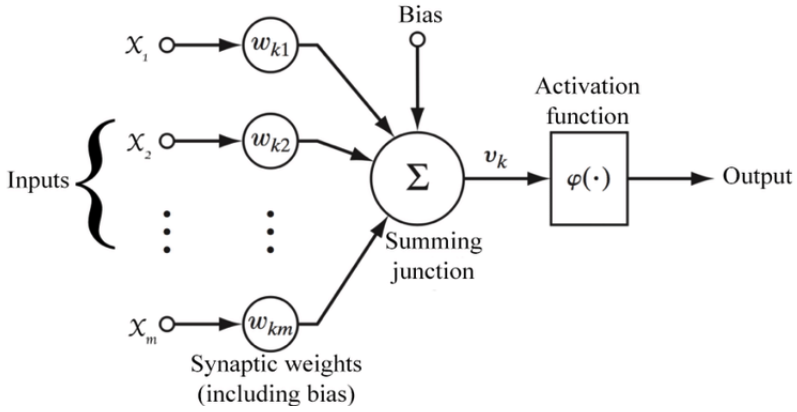


Figure 1.12: The structure of a neuron in a feedforward artificial neural network.

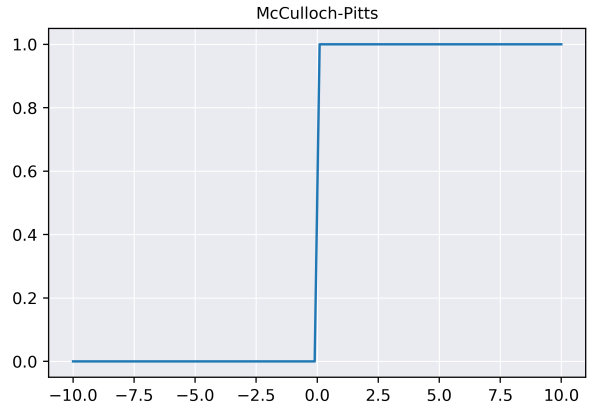
Let us then define as $\hat{W}^k = (w_{k0}, w_{k1}, \dots, w_{kN})$ the weigths defining the k-th neuron.
With this formalization, the signals collected by the neuron are defined as

$$v_k = \hat{W}^k \cdot \hat{X}^p = \sum_{i=0}^{N} w_{ki} x_i.$$

Finally, on this quantity the so called *activation function* φ which emulates the signal threshold is applied; this function is usually differentiable because of the optimization algorithm used in the training phase.
Historically ([54]) the first chosen function was the *McCulloch-Pitts* function,

named in honour of the authors of the paper, defined as

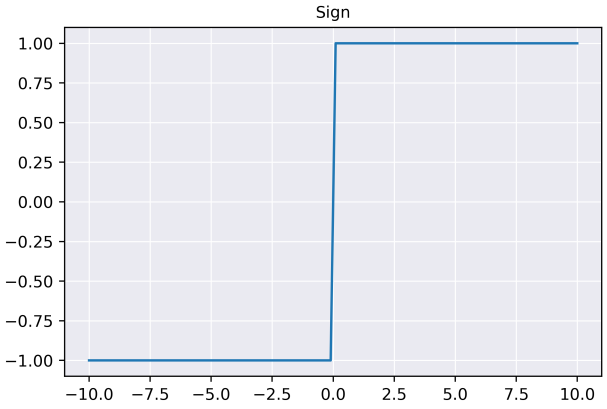
$$\varphi(v_k) = \begin{cases} 0 & \text{if } v_k \leq 0 \\ 1 & \text{if } v_k > 0 \end{cases}.$$



Nowadays, the most common choices are:

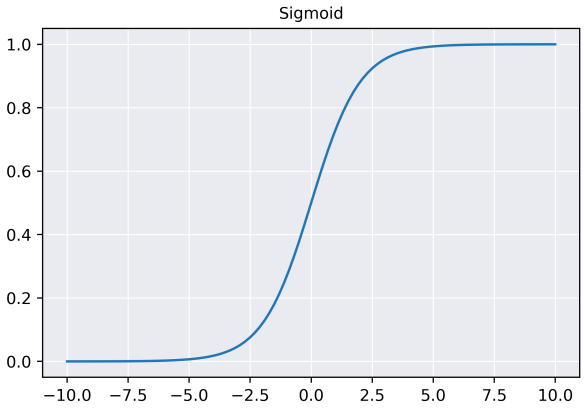
- *Sign* function:

$$\varphi(v_k) = \begin{cases} -1 & \text{if } v_k \leq 0 \\ 1 & \text{if } v_k > 0 \end{cases}.$$



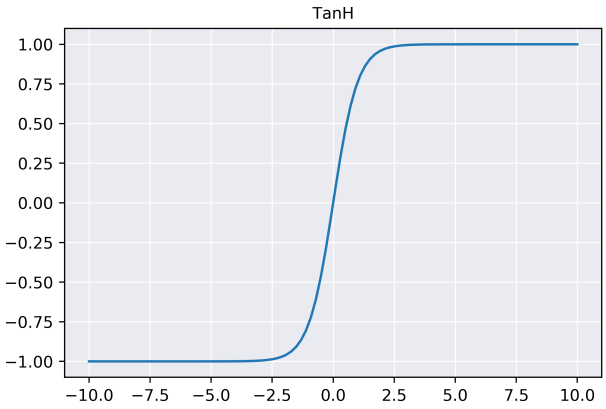
- *Sigmoid* function:

$$\varphi(v_k) = \begin{cases} 0 & \text{if } v_k \leq 0 \\ 1 & \text{if } v_k > 0 \end{cases}.$$



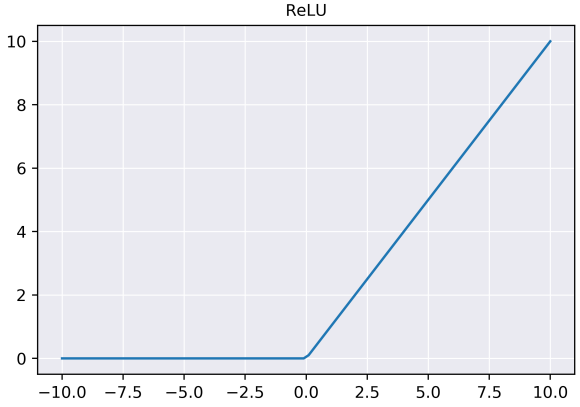
- *Tanh* function:

$$\varphi(v_k) = \begin{cases} 0 & \text{if } v_k \leq 0 \\ 1 & \text{if } v_k > 0 \end{cases}.$$



- *ReLU* function:

$$\varphi(v_k) = \begin{cases} 0 & \text{if } v_k \leq 0 \\ 1 & \text{if } v_k > 0 \end{cases}.$$



It has to be said that today the *Sigmoid* and *TanH* are less and less used because of a problem that can occur during the training phase called *Vanishing gradient*[12] while *ReLU* is becoming the most diffused choice.

Once chosen the activation function, the output of the neuron is defined as

$$o_k = \varphi(v_k)$$

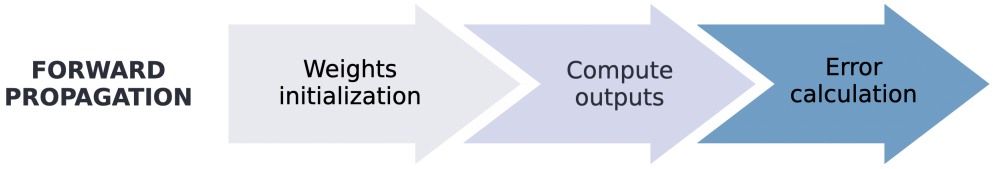
and this will be provided to the next layer neurons or compared with the ground truth samples, depending on the position inside the network of the considered neuron.

The learning process

When talking about the learning process of a neural network is is usually referred to the *backpropagation* phase. The name derives from the fact that the learning phase can be thought as splitted in 2 different phases, one moving *forward* along the net and the other moving *backward*.
Specifically, the first step in the forward phase is an inizialization of the weigths of every neuron of the network; the easiest way to initialize is by a random approach, but there are many other techniques that sometimes could lead to better performances in terms of both solution quality and execution time (see for example [57] and [58]).
Then, a set of samples, called *batch*, from the training set is used to get

[12]Some details will be given in the next section.

the output of the network, given the values of the weights set in the step before; also in this case, there are multiple approaches possible in relation to the number of training samples in the batch, depending also from the minimization algorithm used (see [11], [60] and [61]).
At the end of the forward phase, the error between the ground truth value and the output, in respect to a chosen metric[13], is computed for every sample of the batch.



With the error computed, the backward phase starts in which a gradient based algorithm[14] allows to minimize the error by finding the optimal set of weights for the ntework. This process takes the name of *backpropagation* and it was first formalized in 1986 (see the original paper [62]).
As clearly stated in [43] – *At the core of backpropagation is a method for calculating derivatives exactly and efficiently in any large system made up of elementary subsystems or calculations which are represented by known, differentiable functions* – and, specifically, is a smart and efficient way to use a very simple concept, as the one of the *chain rule* for differentiate.
The chain rule in differentation simply states that, having

$$f(x) = (g \circ h)(x) = g(h(x))$$

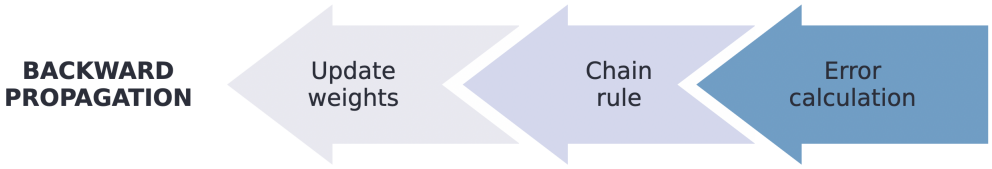
with g and h differentiable operators, then the derivative of f in respect to x can be computed as

$$\frac{\partial f}{\partial x} = \frac{\partial g(h(x))}{\partial x} = \frac{\partial g(y)}{\partial y} \cdot \frac{\partial y}{\partial x}.$$

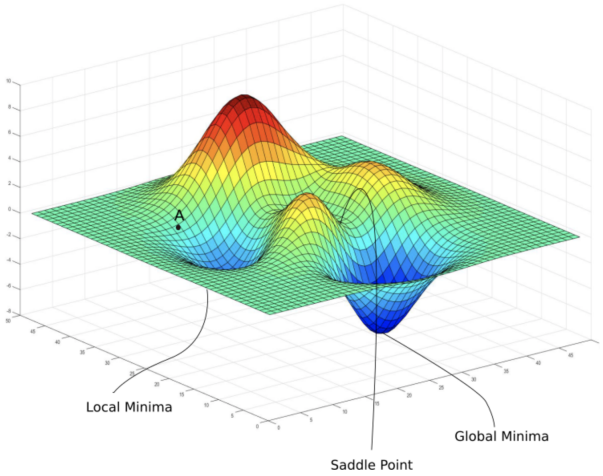
This very simple, yet very powerful tool, allows to move backward in the net, from the errors to the weights, guiding in the optimization of them in order to minimize the error.

[13] The choice of the *Loss* function to compute the errors can be done, also in this framework, in order to add some regularization; also, another technique called *dropout* [69] can be useful to achieve that goal.

[14] There are different gradient based techniques, like *batch gradient descent* or *stochastic gradient descent* that can lead to computational differences and different results; for some details see [35].



This procedure usually converges to a local minimum being a gradient based algorithm, but there is no guarantee that it is also the global minimum.



More, as previously mentioned, depending on the choice of the activation functions, during this procedure the value of the gradient could tend to 0, causing issues in the weights update phase. This problem, known as *vanishing gradient* ([63],[56]), is mostly related to the properties of the derivative of the activation function, fact that lead to the choice of other functions different from the classical ones, such as ReLu function and its *Leaky* version.

Before concluding the section, let us wrap up the most important parameters that need to be chosen in order to train a neural network, again using the popular library *scikit learn* as reference, where such a model takes the name of *neural_ network.MLPClassifier*:

- **Number of hidden layers** : it represents the number of hidden layers to use inside the nework.
- **Number of neurons for every layer** : self explanatory; in `sklearn` the combination of this parameter and the previous one is represented by `hidden_layer_sizes` which is a tuple where the number of entries is the number of hidden layer and the value of the entry i represents the number of neurons relative to the i-th layer.

- **Solving algorithm** : it defines the minimization algorithm chosen to train the net and is called `solver`.
- **Activation function** : self explanatory; in the library is called `activation`.
- **Number of samples before updating** : it is called `batch_size` and it is related to the gradient method chosen as solver.
- **Regularization** : it is a value, called `alpha` that is responsible for introducing regularization inside the network.

1.10 Forecasting in business

Now that a quick overview of some of the mathematical tecniques that can be used to make forecasts has been done, let us go back to the application scenario in which some of them have been applied in the work.

When it comes to forecasting, it is particularly interesting to do it in a business framework, because anticipating issues and opportunities can have a dramatic impact on the business itself; to present some possible scenario, following what done in [59], some typical forecast situations in business are proposed:

- A company has to forecast revenues deriving from the sales of different items belonging to its offer in order to keep track of the business, evaluate productivity and control the likelihood of being unable to meet orders.
- A VC found has to decide if, when, where and how much capital to invest into a new possibility; to do that, it has to forecast the return that the investment is likely to produce in the next years.
- A government has to forecast the value of many different economic variables in order to decide if taking some actions or not.

These few examples, are just some over the billions that we could provide, and made clear some of the situations in which forecasts can be implemented in business frameworks; the most important point we want to highlight is the common feature of them all, that is that the forecasts lead to decisions. In fact, being it a political decision like in the case of the government, or related to the long, mid or short term strategy in the other cases, the results of a prediction are always the basis of future actions that will impact on the life of the business itself with different form and magnitude.
Here we need to specify the role of the forecaster, that is the one who provides the information the decision maker will use to take business actions.
The goal and the impact the forecast has on the business strictly depends on

the nature of the business itself, but it is always the starting point in taking actions at different levels.

1.10.1 Intepretability

Citing our previous phrase, we see the forecaster as the one who provides the information the decision maker will use to take business actions. It is then obvious that, in order to be useful, the information provided has to be clear and understandable for the decision maker; he or she has to be able to interpret both the results of the analysis and the process which produces them, understanding all the operative steps from an high level and interpret all the variables that took part in the process, along with the motivations behind the different choices.
We refer to this as *interpretability* and it covers a crucial role in order to obtain the trust in the process by the decision maker; in other words, a perfect forecaster not capable of gaining the trust of the decision maker, will be useless because it won't be used.

As we'll see, there are different ways that try to move in this direction, aiming at making comprehensible for non technical people, being often the decision maker more focused on business than on math and statistics, all the instruments that take part to the forecast definition.

1.11 To sum up

Summarizing what presented in the chapter, we started by introducing the concept of forecasting, we made some basic observations by using different examples in order to clarify the meaning of forecasting, the complexity of the task and some approaches that can be used in facing such a problem; we then introduced the Machine Learning as a class of algorithms and techniques used to forecast in different scenarios, later introducing different examples of the possible techniques usable; we then concluded by providing some examples of business situations in which forecasts can be used in order to understand the reasons which motivate such an approach.

In the following chapter we'll deal with the importance of data in statistical techniques and with different actions we need to take care of while using data in algorithms.

Chapter 2

Data processing and ML algorithm training in a production scenario

2.1 Introduction

In the previous chapter we introduced the problem of forecasting along with the introduction of some algorithms of Machine Learning; we highlighted how they are based on the wide usage of a huge amount of data which is related to the specific domain of application.
We said that, in order to use data based techniques, data must be of course available and also reliable for what concerns the information represented and must be seen as *truly* representing the specific application domain.

While in the previous chapter we focused more on theory behind the algorithms and the procedures by which using data to build models, in this one we'll focus more on the data and the operations we need to manage when dealing with it in real case scenario; we'll also give the overview of the operative steps to take in order to train a Machine Learning algorithm with all the different tasks to perform before reaching the deployment phase of the model built.

2.2 The training process

As we already introduced in the previous chapter, we call *training phase* of a model the set of actions to implement in order to algorithmically define an instance of that mathematical model by fine tuning its parameters in order to represent the *best*, with respect of one or more metrics, functional which links some input variables, named *predictors*, with one or more outputs.
By saying *training process* instead, we also consider the set of operations

that we need to perform before, while and after training a specific model; these operations are related to data and to an overall analysis that needs to be done in the whole process before considering the model as ready to be used and deployed.

The different steps of such a process can be identified in the followings:

a. *Analysis of the application domain*

b. *Problem definition*

c. *Data gathering*

d. *Choice of the technical tools*

e. *Data cleansing*

f. *Data processing*

g. *Model choice*

h. *Model validation*

i. *Model deployment*

j. *Model update*

The previously introduced and later explained *training phase* is part of *g.* being involved in the process of chosing the most suitable model for the purpose.
It is specified that, even if the operations presented in the above list are in theory well defined and separated, in the practice they often overlap in many ways; for example, the choice of the tools to use in the process could be influenced by the data that is used, or again, some steps in the *data processing* can be done keeping in mind of many different aspects of the models that are going to be tested in the *model choice* phase. However, for the sake of simplicity aimed at educational purposes, we consider the steps as well defined and separated.

The next sections will cover every point of the previous list of steps in details.

2.3 The domain

Steps *a.* and *b.* are related to the need of dive in the application scenario before starting to build the model; this phase is very important because it is in it that we acquire every global information related to the domain

Name	Surname	Sex	Age	Employed
Filippo	Rossi	M	29	False

Table 2.1: Example of a data sample which could represent a person in terms of some of his/her features; it is showed how many of them could be numericals while other categoricals or even alphabeticals.

which will be crucial in order to chose the data, interpret the meaning of the different variables and extract conclusions from the results.
Even if it is simple in the concept, it is one of the most time consuming parts of the whole process, because it requires study of the context and talking with experts of the field.
The second phase is the one in which the problem is defined; the goal is made clear as are the available resources, the possible data sources and the evaluation metrics.

2.4 Data gathering

Once the application scenario is approached and the problem defined, it is time to evaluate all the available data; however, first, we need to understand exactly what is *data*. According to [39], data can be defined as *the measurements or observations that are collected as a source of information*; of course, depending on the purpose of the data collection, there are different ways to represent data and for this motivation there is a variety of different types of data; however, due to the computer based data analysis methodologies, we implicitly consider the data to be digital, or deriving from a digital transposition.
Citing only a few examples, the collection of environmental measures made by analogic or digital sensors, the images acquired by an optical camera, the posts that we share on social networks, the sound recorded by domotical speakers or the answers we provide to customer satisfaction questionnaries are *data*.
Depending on the application, the data could be *private*, as it happens for data recorded within a company, or *public*, like in the case of social media and government statistics.
In its basic form data can be intended as one recorded instance of a particular variable, but in the application we refer to data as samples, each one representing a certain entity, described by the mean of many variables, called in this context *features*; we already said that, in Machine Learning frameworks, the feature that we want to forecast is called the *output* variable and the ones that are used to predict it are called *inputs* or *predictors* sometimes.

Going back to our pipeline definition, once a problem has been defined and the goals set, we need to chose the data we are going to use inside the project, by scraping all the available data and evaluating the way it will be involved inside the operative flow. For example, if we are forecasting weather conditions, we'd probably search for atmospherical data provided by orbital satellites.

It is underlined that, in a production scenario, it is not obvious that the data needed in order to succeed with project goals exists and in such case, we would need also to implement the data collection methodology; in the previous example, if no atmospherical measurements were available, there would be the need to position satellites around the Earth in order to retrieve the data; it is clear that, depending on the scenario, this operation could be very expensive in terms of time and resources.

2.5 Technical tools

While making numerical analysis, it is today the standard to perform it by using computers and digital calculators. In fact, due to the technological advances, they are today able to perform extremely heavy computations in small amount of time; their capabilities increase year over year at a tremendous rate and this fact opens to computations simply not possible before.

The tools for the analysis are then constituted by the *hardware* from one side, and the *software* from the other; the term *hardware* refers to the physical parts constituting the calculator/s which are involved in the analysis, while the term *software* refers to the code containing the instructions to perform algorithms.
Until some years ago the most important part of a computer aimed at performing computations with algorithms was the *CPU (Central Processing Unit)*; today, with the advent of much heavier algorithms, like *Neural Networks*, more computation power resources are needed and has been involved in the computations also the *GPU (Graphics Processing Unit)* which was hystorically developed with other purposes, more related to the rendering of images and graphical objects on the monitor screen.
Talking about the software, we mainly refer to the programming languages available and suitable for the purposes; the choice of a particular programming language is related to people's knowledges, preferences and attitudes, but some of the most common possibilities are represented by C, C++, Python and R, even if many of them are defined one up the other (for example, Python is written in C).
Another interesting software, relatively new in the Machine Learning panorama, is represented by Tensorflow and it is mainly focused on the Neural Networks

and the Deep Learning.

2.6 Data pre-processing

In this section we'll deal with one of the most important parts of the training process, that is the so called *data pre-processing* phase.
It represents the operative steps in which, once the data has been collected, we perform operations in order to prepare it for the model training phase; these operations aim at, from a high perspective, choosing the best data representation in order to maximize the information that can be extracted from it. Even though this definition could be not completely clear, we'll make examples in the following sections.
The popular phrase *garbage in, garbage out* represents probably at best the impact this phase has on the whole training process; in fact it represents the true concept saying that, if you put *bad* data inside a model, you cannot expect that the model will produce acceptable results. To make an example, if we use corrupted data from orbital satellites in our weather forecasting, the resulting model will be built upon incorrect data and, as a result, it may be forecasting 45C° for a winter December night!
The following sections represent all the operative tasks usually needed to *build* a suitable version of the data before feeding it to the model.

A final note before diving in each different topic regards the very first phase usually performed right after the data collection, that is the so called *EDA (Exploratory Data Analysis)*, which represents the first deep observation of the data we just collected; the data distributions are analyzed and visualized thanks to graphical tools like histograms, density plots, heatmaps and so on. This phase is crucial because it makes us able to acquire precious knowledge and find hints in order to chose what the following approach will be, representing the lens thanks to which we find the issues we are going to face in the operations we'll define in the next sections.

Given the framework, let's move to the presentation of the different operative steps.

2.6.1 Data cleansing

The first step represents a *double check* on the quality of the data and its reliability. If we consider all the possible sources of data, we realize that they can come both from human and technological inputs: paper made personnel satisfaction questionnaries collected within a company, which will later need a digital transposition made by human resources, are an example of the first category, while temperature data from a thermometer are examples of the latter.

Provided this, it is clear that both the acquisition methodologies can reflect some errors of the process itself; a person who has to transpose thousands of paper questionarries into digital tables could make some errors in inputing values, both in the form as in the concept (typos, age which become date of birth, etc.).
On the other side, an uncalibrated instrument could provide incorrect measurements from the beginning, or if someway some of its components breaked it could start to provide incorrect data suddenly.

While the first class of errors could be easier to detect while looking at the data, the second would be much more difficult, being it a systematic error; in such a scenario a deep knowledge of the domain, acquired at the first step of the training process, is important.

However, it is pretty common to have a huge quantity of data, so a manual approach would be simply not feasible; for this reason different global automatic checks are usually implemented on the whole set of features representing the data.

The most common procedures regard the removal of empty samples, in toto or in some of their constituting variables, and the check over the range of the possible values of every feature; while the first check is related to the calculator need of numerical value as input variable (empty values are not numbers), the second checks the consistency and the coherence of the representation.
In some cases, depending on the scenario, missing and incorrect values are filled and replaced by others using different techniques, among which also statistical ones, as presented in [66], while in others they can just be removed; this choice depends on the need of maintaining an equilibrium between real and edited data, also having enough samples in order to be able to train a model on them.

Sex	Age	Weight (kg)	Height (cm)	Employed	Empl. on
M	18	67	168	True	01-02-21
F	15	54	156	False	NaN
M	30	85	182	True	13-01-20
NaN	1991	88	177	True	05-09-17
NaN	NaN	66	179	1	NaN
...	...	...	...	...	...

Table 2.2: Set of dummy data samples representing the result of a survey aimed at studying the employment situation in a certain city, performed on 01-03-21.

To make an example and clarify what is being said, let's look at the dummy table in 2.2. If we look at it, we notice that some values are missing[1] and others seem incorrect. In fact, blue cells represent missing inputed value and red ones instead show some different types of error in which, for example, the age becomes the date of birth; the green cell also seem the result of a moment of confusion of the inputer who indicated a purely boolean value with another very common notation for booleans in which *False* is represented by 0 and *True* by 1; for the moment we assume that the value '1' represents a boolean *True* and move on.
Analyzing the last feature we see how missing values populate it; in this case however, a missing value could be interpreted not as an error but as a consequence of *Employed* feature being *False*; in fact, a person that is not employed, has no employment date.
This final observation leads to another intepretation of the green cell previously discussed; if what earlier supposed was true, then the value of feature *Empl. on* of the same sample should be populated with a date; on the contrary, if we assume the value inside feature *Empl. on* to be true, then we expect the value in the green cell to be '0'. The two scenarios contradict each other and make it difficult to understand the last sample.

In such a situation, the best cleaning approach would probably be discarding the last sample, because of the poor information it would add to the model, changing the red cell into the corresponding age, and fill the value inside the feature *Sex* of the 4th sample by observing that, statistically, a 29 years person, 177 tall who weights 88 kg is probably a men, according to the last demographics surveys of the city. This would lead to the *cleaned* version of the table 2.2

Sex	**Age**	**Weight (kg)**	**Height (cm)**	**Employed**	**Empl. on**
M	18	67	168	True	01-02-21
F	15	54	156	False	NaN
M	30	85	182	True	13-01-20
M	29	88	177	True	05-09-17
...	...	...	...	...	...

Table 2.3: Cleaned version of table 2.2

2.6.2 Feature engineering

With this name we refer to the process of creating new features starting from the original ones which represent the data.
The motivations behind this task are to be found on one side in the way

[1]Missing values are usually presented with *NaN* in a numerical framework.

computers deal with data; in fact, we already know that computers can only deal with numerical values and all the data provided to them has to strictly comply with this rule. Examples of such a scenario are the transformation involved in dealing with textual data usually in a *NLP* framework[2], where the words are represented by numbers or arrays by *Bag of words* embedding or *TF-IDF* representation, as done in [18].
The second driver in the feature engineering phase is the sensibility to scale and distribution of data of many models; as shown in [22] and [26], architectures like *Linear Regressors*, *Nearest Neighbors* methods, *Radial Basis Function (RBF)* kernels, and anything that uses the Euclidean distance are sensible to scale, and for this reason in these cases it is often a good idea to normalize the features so that the output stays on an expected scale.
Other types of models instead are not sensible to this feature of the data; examples are represented by Tree based methods.

To better understand some concepts at the basis of the feature engineering process, let's consider now the first sample of the previous example in table 2.3

Sex	**Age**	**Weight (kg)**	**Height (cm)**	**Employed**	**Empl. on**
M	18	67	168	True	01-02-21

The information in it contained are essentially the following:

i. The person is a male.

ii. His age is 18.

iii. His weight is 67 kg.

iv. He is 168 cm tall.

v. He is currently employed.

vi. He got the job on February, the 1st, in the 2021 year.

vii. He has been employed for 28 days at the moment of the survey is implemented.

The thing is that, if not adequately treated, the data in such a form would not be manageable by a computer, being not numeric in many of its features.

[2] *Natural language processing* or *NLP* is a subfield of linguistics, computer science, and Artificial Intelligence, concerned with the interactions between computers and human language; in particular, it studies how to program computers to process and analyze large amounts of natural language data.

More, the information in vii. is implicit in the sample.

The goal of feature engineering phase here is to translate the very same information in order to maximize the capabilities of a computer to use it within the model training phase.

In the proposed example this operation would involve the last feature only and would consist in the transformation of the information (implicit and explicit) represented in the date into new, numerically manageable features; this would produce:

Sex	Age	Weight (kg)	Height (cm)	Employed	Empl. day	Empl. mth	Empl. yr	Empl. duration (days)
M	18	67	168	True	1	2	2021	28
F	15	54	156	False	NaN	NaN	NaN	0
M	30	85	182	True	13	1	2021	15
M	29	88	177	True	5	9	2017	1273
...	...	...	...	...	...			

Starting from the date in *Empl. on* feature, the day, the month and the year are extracted and represented in dedicated new features; more, the time distance between the date of employment and the date in which the survey took place has been explicited.

A final note regarding the *NaN* values that are still present: since they are a consequence of the representation choice and representing days, months or years simoultaneously to the value "0" of the feature *Empl. duration (days)*, it seems legit to fill these values with "0" value, finally obtaining the resulting data

Sex	Age	Weight (kg)	Height (cm)	Employed	Empl. day	Empl. mth	Empl. yr	Empl. duration (days)
M	18	67	168	True	1	2	2021	28
F	15	54	156	False	0	0	0	0
M	30	85	182	True	13	1	2021	15
M	29	88	177	True	5	9	2017	1273
...	...	...	...	...	...			

Table 2.4: Data with new numerical features engineered from the old feature *Empl. on*.

2.6.3 Feature encoding

Altough sometimes the *Feature encoding* phase can refer to the process of mapping set of numerical values into others (for example processing pixels of images in order to improve the performances of an image classifier as done in [29]), in this framework we refer to this phase as the process of transforming not numerical variables into numerical values, preserving as much as we can the information the original variables bring.
At this purpose there are different techniques which can assign a numerical value to a categorical one (the *encoding* phase precisely) as presented

in [22], like integer encoding (sometimes referred to as *dummy* encoding) or frequency encoding.
Depending on the nature of the variable to encode and the model that will be trained upon the data, the techniques vary a lot; examples of feature encoding when using ANN model is presented in [31] while different considerations about encoding in a tree based model framework can be found in [42].

In the previous example an appropriate choice of encoding strategy would be a binary encoding for *Sex* and *Employed* features producing

Sex Enc	Age	Weight (kg)	Height (cm)	Employed Enc	Empl. day	Empl. mth	Empl. yr	Empl. duration (days)
0	18	67	168	1	1	2	2021	28
1	15	54	156	0	0	0	0	0
0	30	85	182	1	13	1	2021	15
0	29	88	177	1	5	9	2017	1273
...	...	...	...	...	...			

Table 2.5: Data with non numerical features encoded.

After this operation, as shown in Table 2.5, data is defined by only numerical features and it is now in the correct form in order to be used by a computer in a model training phase. Still, there are some other operations that could be impactful on the quality of the model that will be built and the future forecasts.

2.6.4 Feature aggregation

The first of these potentially useful operation is the *aggregation* of the instances of some features into macro categories; this can be suggested by the presence of too punctual instances which can create a data space so *sparse* that the model could struggle in finding relations between data.
From a more abstract point of view, this mapping procedure is equivalent to introduce an *a priori* knowledge into the model, if there is one, in order to help the model to observe the correct information brought by some features. Again, as it happened for the encoding phase, this one also can be viewed as a characterization of a *feature engineering* phase.

If we observe the data in Table 2.5, depending of the goal of the analysis, *Weight* feature could be informative not in the punctual value of its instances, but in the macro category they can be split in; in other words, we can assume that there is no reason to treat as different people weighting 88 or 85kg, but probably it is much more useful to cluster the instances in macro categories like *underweight*, *normal weight*, *overweight*; the same approach could be used for the *Height*, the *Age* and the *Empl. mth* feature, mapping the months into the season they belong.
While doing this, we are conceptually mapping a numerical variable, like a *100 kg* weight instances, into a class, like *overweight*; to make this class

numerically treatable, we need to encode it by the choice of a suitable map. This is usually done by chosing a map which preserves the ordering in the original space.

Considering the weight feature, we could chose the following map:

$$\begin{aligned} \text{underweight} &\to 0 \\ \text{normal weight} &\to 1 \\ \text{overweight} &\to 2 \end{aligned}$$

by which, being $x, y \in \mathbb{Q}$ so that $x \leq y$, two generic instances of the original weight feature, then $f(x) \leq f(y)$, where f is the aggregation map; in this way the original ordering is preserved.
Again, in order to properly find an opportune tradeoff between aggregating features and not doing it, a deep understanding of the problem goal and the application domain is required.

2.6.5 Feature selection

While the previous sections aimed at creating the best, in terms of provided information, representation for the data, this one aims at doing it, in the optimal way.
If in the first scenario the link with an output variable (all the operations performed would be approximately the same regardless the output variable) was not crucial, the feature selection phase is strictly linked to the output variable. In the practice in fact this phase regards getting rid of as many features as we can, without impacting on the accuracy of the forecast of the output variable; given this, it is obvious that the very same features could be incredibiliy informative with respect to an output variable, and completely useless with respect to others: using the weight of a person as predictor of hearth diseases could be extremely useful, while trying to use the very same variable in a weather forecaster would be totally ridiculous!

This need is justified by different aspects, most of which are linked to numerical and algorithmical matters. As shown in [71] and [2], reducing the number of features impact on the training time of the model, as less computations are required to converge to a solution; more, it allows to reduce the variance and hence reduce the potential overfitting (see Chapter 1 for the definition); in general, selecting *all and only* the most informative features, benefits both the model training phase and the accuracy of results.
Other than that, we have to always remember that such a forecaster is likely to be used in a production framework, in which decisions will be taken upon the obtained results; under this light, as already specified in Section 1.10.1, it is crucial to make both the results and the process used for obtaining

them the most clear and interpretable as we can, especially for non technical people.
Provided this need, feature selection phase allows also to reduce the number of variables involved in the process, keeping only the most informative, helping in making everything simpler to interpret for the decision maker.

Now that the motivations have been made clear, let's focus on the methods that allow to reach such a goal.
There are different techinques which apply in different assumptions and scenarios in order to reduce the number of features; while some theoretical backgrounds related to the topic can be found in [34], the different methods can be globally clusterd, as suggested in [32], into *Wrappers*, utilizing the learning machine of interest as a black box to score subsets of variable according to their predictive power, *Filters*, which select subsets of variables as a pre-processing step independently of the chosen predictor, and *Embeddings*, which perform the selection while training the model. Belonging to the last category are methods like *PCA (Principal Component Analysis)*; with this class of techniques the task becomes not to select the most informative feature between the provided ones, but to combine them into a smaller globally more informative number of features. Examples of such an approach can be found in the Neural Networks and in the process which leads to the definition of the *fully connected layer*, obtained by embedding the information previously obtained into a lower dimensioned space, namely the layer itself.

Other methods regard the study of linear correlation between features, as shown in [37] and [38]; these methods are pretty simple to implement and their risults can be easilly interpreted thanks to the usage of visual tools like the *correlation matrix* like the one in Fig.2.1

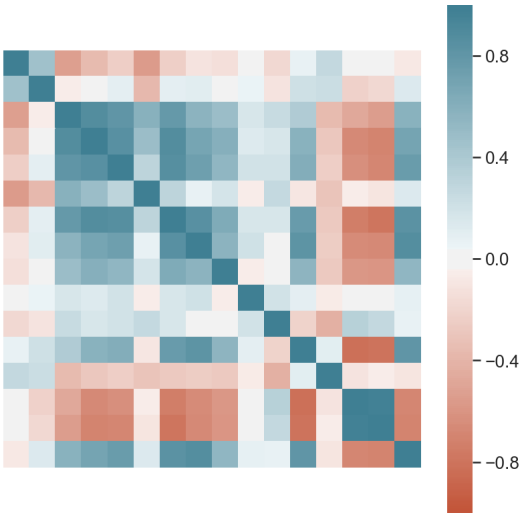


Figure 2.1: Example of a correlation matrix computed on a dummy dataset; the values, namely the *Pearson index*, vary between -1 and 1, with -1 representing high inverse linear correlation, 1 high direct linear correlation and 0 no linear correlation at all.

However, by definition, such a process can find only linear relations between variables which may in some cases not be enough or not informative at all.

Considerations about the variance of the features can also be viewed as ways to discard some of them; in fact, if a feature has low, or even 0 variance in its instances, it means it brings poor or no information at all in forecasting any output.

Such a scenario can be better understood when considering the example data table in the previous sections. Imagine you are performing your EDA and finding that the feature *Sex Enc* assumes the value *1* in only one sample, while in the others is always *0* (meaning the table contains only 1 female sample); this situation makes the feature almost constant and, for this reason, considering it in a forecasting model would probably bring no information at all, not introducing any variance into the forecast, with respect to the feature itself. Discarding the feature *Sex Enc* in this case would probably not affect the precision of the forecast, on the contrary affecting the complexity of the training phase reducing it and simplifying the reading of the model.

Finally, as we already highlighted many times before, we have to introduce and exploit the most *a priori* knowledges information as we can within the whole process; here this could be done by analyzing the link that could

exist between the variables and the output of the model from an high perspective.
Imagine for example, always referring to data in Table.2.5, that such a data is used to build a model which forecast if a person will get a professional promotion in the next year, using the data presented as predictors; in this view, even before starting the process of algorithmically selecting the most informative features, we can assume that variables like *Height* and *Empl. day* or *Empl. mth* do not bring information as predictors in order to say if a person will or not be promoted; a slightly different discussion could be made about *Weight* feature that can be seen as a personality indicator.
Given these considerations, with the goal of predicting if a person will or not likely receive a promotion in the next year, *Height*, *Empl. day* and *Empl. mth* feature could be probably discarded, with this approach, keeping instead as predictors the variables we expect, or only suspect, to have an impact on the output variable.

This concludes the presentation of the operations needed before approaching the model related tasks.

2.7 Modelization phase

While the previous sections were mostly related to data and the process to make it more useful for extracting information, even though always keeping in mind the final goal, this one is related to the model choice and definition; this phase will lead to the definition of a model which can perform the forecast in the interested scenario.

As we presented in Section 2.2, we can find different phases within the modelization, related to the evaluation of different candidate models and their validation in a simulation environment.
While we already talked about the training phase in the previous chapter, we want to spend a few words about the process from an high perspective.
The first step is surely the evaluation of different mathematical models in being the forecaster; this means to select an appropriate type of architecture among different ones (Decision Tree, ANN, etc.) and eventually, once chosen it, to find the most suitable set of hyperparameters (in the case the model needs them) in order to achieve the best possible solution, in the sense of forecast accuracy.

While the second phase is almost always performed by exploring the space of the hyperparameters in many ways, as briefly presented in the next section, a choice made in the first phase could be influenced by very different factors, some of which more based on theoretical assumptions, while others

on more practical needs.
The need of simplicity could force to use a Linear Regressor, while the categorical nature of the feature could lead to the choice of a tree based method like a *Random Forest*; the goal of the model itself could force the hand in some direction: a time series forecasting problem would probably need the use of a *LSTM* or *RNN* while an image classification problem would require the use of a *Convolutional Neural Network*.
To conclude the section, there is not the perfect model in absolute, but instead there is a suitable model depending on the needs of interpretability and easiness, the type and the amount of available data and the nature of the problem itself.

2.7.1 Hyperparameters choice

As previously said, a very important aspect to pay attention to in order to build a well performing model is the choice of the hyperparameters that define it.
It is clear that, depending on the considered model, the set of hyperparameter changes.
Looking for the best set of parameters is usually not an easy task; depending on the problem, on the amount of available data and the model, they vary a lot in a possibly not finite range. For this reason, methods to accomplish the task are based on a research of the hyperspace which the parameters live in.
This search could be performed in a *greedy* methodology, extensively exploring the hyperspace looking for the best combination of parameters, or could be approached based on Bayesian Optimization as done in [67] and [65].

At the end of this phase, we'll hopefully have our set of best performing parameters for every candidate model.

2.7.2 Cross validation

When training a machine learning model it is crucial to define an appropriate validation strategy in order to be sure, or at least confident, the performance of the model in production will approximately be the one estimated in the development phase.
To reach this goal there is the so called *cross validation* technique, which is fundamentally a way to split data and compute the results starting from the same historical dataset.

There are different approaches depending on the specific choices made on the splitting criteria, but generally we can split the methods in 2 main categories:

- Holdout
- K-Folds

Holdout cross validation

It is probably the most common and used approach and the first tried during the training phase.
With this approach the available data is splitted, usually randomly, in 2 different parts, the *train* set and the *test* set; then, the model is trained on the *train* set and later evaluated on the *test* set by using some chosen metric; the size of the train and test set is commonly chosen $2/3$ of the total dataset for the first and $1/3$ for the test set.
The results obtained by using this method are really dependent on the data that are put in the train and test set, thing that could lead to high variance in evaluations. Attempts to deal with such a problem are represented by the following methods and involve the usage of a multiple splits.
In *sklearn* python library the function which prepare the data with this method is called `Train_Test_split`.

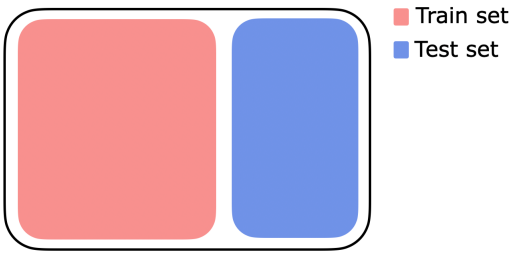


Figure 2.2: *Holdout* cross validation schema.

K-Folds

This set of techniques are called *K-folds* because the dataset is splitted in K parts (precisely *folds*) of which $K-1$ are used as train set and the remaining one as test set; then this procedure is iterated for the K parts previously obtained and the results are averaged.
This method should decrease the variance but in return it becomes heavier, in terms of computing resources needed, the larger K is.
In *sklearn* python library the function which prepare the data with this method is called `KFold`.

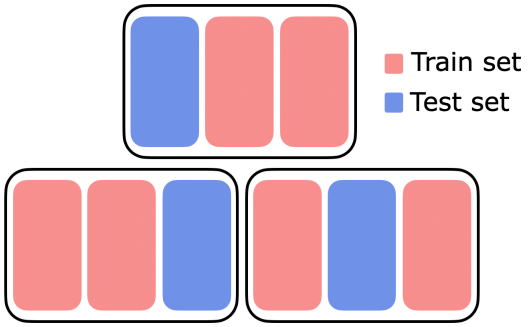


Figure 2.3: *K-Folds* cross validation schema.

When $K = N$, where N is the number of datapoints in the set, this method gets the name of *Leave On Out Cross Validation*.
In *sklearn* python library the function which prepares the data with this method is called `LeaveOneOut`.

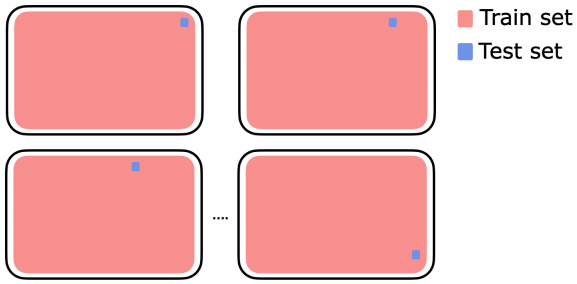


Figure 2.4: *Leave One Out* cross validation schema.

While training the different models in the evaluation phase, a cross validation methodology is implemented in order to get the performances of every model in a production simulated environment and get and indicator of what the real performances would be when in production.

2.8 Performances evaluation

This phase represents the conclusion of the development phase and provides the resulting model which will be implemented in the deployment phase.
Depending on the different scenarios, different metrics can be chosen to evaluate the results provided by the models, each one based on the distance, according to the chosen metric, between the real value of the output variable and the one forecasted by the models.
Let's remind that, being in the development phase, the training phase of a supervised learning approach is based on the presence of real values of the

output variable for the historical data, which the models are trained on.

Common metrics used in this phase are

- *mre (Mean Relative Error)*

$$mre_G^{(X,y)^{\text{Test}}} = \frac{\sum_{i=0}^{N} \frac{|G(x_i^{\text{Test}}) - y_i^{\text{Test}}|}{y_i^{\text{Test}}}}{N}$$

- *mae (Mean Absolute Error)*

$$mae_G^{(X,y)^{\text{Test}}} = \frac{\sum_{i=0}^{N} |G(x_i^{\text{Test}}) - y_i^{\text{Test}}|}{N}$$

in which G is the built model and $(x_i^{\text{Test}}, y_i^{\text{Test}})$ represents the test set of N elements which the error is computed on.

2.9 Model deployment

To conclude the chapter, we present the last phase of the process which is the deployment of the build forecasting model; depending on the application domain, the role of the model and the policy of the company (in the case of a business application), the ways to make the forecaster *operative* are really too many to count. These could involve the deployment on local or online servers and the implementation of dedicated software applicative.
Also, this phase comprehends usually a period of time in which the model is tested in production, called *beta test phase*, aimed at evaluating its performances before investing money in making it user accessible.

After the model deployment, it could be necessary to update it from time to time; in fact, changes in the data (reflecting changing in the application domain) which will be feed to the model could occur, with the need to make the model itself able to manage such new data.

2.10 To sum up

In this chapter we provided the notions at the basis of every training process in this framework; we explained how to process data in order to maximize the information from it extractable before feeding it to a model and we then talked about some operations needed in order to fight the most common issues when training a model and in order to chose the best candidate model. We concluded providing some information which follow the development phase, moving to the deployment phase which represents, after some beta tests, the conclusion of the process.

Chapter 3

Forecasting in Destination Management

3.1 Introduction

In this chapter we'll present some applications of the previously introduced methodologies and techniques aimed at making forecasts in a real company scenario.

As previously anticipated, our work has been developed in a collaboration with a tourist company operating in a B2B and B2C framework. Their offer is mainly touristic and the projects we've developed together were related to the *Destination Management* branch, that is the study and the implementation of actions aimed at better managing the company offer related to tourism.

The overall goal has been to make forecasts about the sales of some items belonging to the current tourist offer to identify new possible opportunities to exploit in the company business.

As presented in Chapter 1 making forecasts, especially in business, is the first step of a much longer process which mainly involves two different categories of actors: the *forecasters*, represented by the people involved in the definition of the forecasting methodology, and the *decision makers*, the people who will actually use the results provided by the forecasters; in the company operative framework, our role has naturally been the one of the forecaster. However, in order to implement a useful methodology, we have always been in touch with the people who would have used our work in the future; we talked about the overall goals, defined different operative milestones and discussed about possible methodologies.
As a consequence of these meetings, together we have decided to focus our

attention on two different topics which could have given the company some hints about opportunities and possible issues about its business; two different projects have been defined, each one at a different level of details and related to a specific topic.

The first project has been related to the topic of *Revenue Forecasting* and its goal was to approach the sales of a particular tourist item belonging to the company offer, define a model which could predict the sales of the same items along the year in different scenarios and finally deploy it in order to make it a tool for the different teams within the company.

The second project was instead related to the tourism activity in general and aimed at *Destinations Discovery*; the goal was to collect a list of all the possible tourist locations around the globe and make forecasts about their tourist activities trend in the next years.

By just observing these two projects, we can immediately notice the big difference that can exist between two forecast related problems; even though, in fact, they both involve a process of forecasting at high level, they differ a lot both in the target of the project and in the implemented methodologies, as we'll later see.

The next sections will introduce the two different topics, explaining implemented procedures and obtained results within the two projects, always according to the NDA we must comply with.

All the numerical computations and all the code produced have been implemented in Python; for more details about how to numerically implement Machine Learning model training procedures, please refer to [27] and [28].

3.2 Revenue Forecast in selling process

3.2.1 Introduction

The problem of forecasting revenues, sales and other variables is very common in many different business contexts: from estimating revenues in the grocery market with Artificial Neural Networks [13], to the forecast of sales in the fresh food market using regression [15] or using tree based methods to estimate the price of electricity in New York [14], companies are fastly approaching this kind of analysis; in fact while the data sources increase rapidly and the computational power exponentially grows along the years, many companies have become proficent in collecting, storing and using data in many forms and ways to accomplish operative tasks and creating tools of

strategical value.

In this framework, tourism sector makes no exception with different approaches implemented depending on the goal; many attempts have been done to forecast the tourist demand from a high perspective, representing the historical information into agglomerated feature data, such as *Monthly Number of Guest Nights in the Hotels*, by a time series structure and making predictions by the mean of different models like ARIMA [12] or Artificial Neural Networks [20]; also, many other basic indicators have been used in the literature like, for example, *Tourist Arrivals by Air* [23], *Tourism Employment* [24] and *Tourism Import and Export* [25].
While the above cited works deal with estimating the tourism demand from a very high level perspective, other analysis can be more focused on forecasting revenues produced by items related to the tourist offer of companies and privates.
To make a practical example of such a scenario let us consider airlines companies: studying historical flights data in order to make forecasts about the future business is one of the most important aspects in a proficient revenue management methodology; estimating flight passenger bookings, as done in [3], or forecasting cancellations, ticket fares, demand and *show-up* rate, like presented in [8], could represent a valuable asset for every airline company, allowing it to proactively implement actions in order to manage issues and exploit opportunities.

The work presented in this section falls inside the last scenario, in which the problem of estimating the revenues deriving from a specific asset inside the company offer, which is related to tourism sector, have been faced.

3.2.2 Cross-sales forecasts

Being more specific, the whole analysis was related to the sales prediction of cross-selled items by the company.

With the term *cross-selling* we refer to a specific business model in which the company offer is usually made by a basic product and different *add-on* possible items; in such a model, the company sells the customer the basic product and tries to convince him/her in buying other add-on items, in order to increase profits from the same customer by increasing the number of items he or she purchases.
Examples of such business models can be found in e-commerce websites ("You have added a new tower pc to your basket; why don't you also add mouse and keyboard?") or while having a lunch at fast food ("You spend 8.9€. Do you want a frappé by adding only 0.2€?"); knowing the behaviour of customers in relation to the basic offer often suggests what they are willing to

pay for add on items.
Under this perspective, our goal has been to forecast and estimate the sales of add-on items to the customers, in a specific time-window, given their behaviour on the basic offer and some personal and contextual information.

Following the notation previously introduced and considered what above said, let us consider T as the time window of the forecast and call E_k the k-th basic offer bundle by the company in the time window T, revenues can be formalized as follows

$$\begin{aligned} y^{T}_{\text{REV}} = \sum_{E_k \in T} y^{E_k}_{\text{REV}} = \\ \sum_{E_k \in T} \left(\sum_{i | C_i \in E_k} y^{C_i}_{\text{REV}} \right) = \\ \sum_{E_k \in T} \left(\sum_{i | C_i \in E_k} G(v^{C_i}_1, \ldots, v^{C_i}_n) \right) \end{aligned} \tag{3.1}$$

where C_i is the i-th customer who bougth the supplementary item E_k and v_l the l-th feature characterizing the customer and cosidered as predictor of the revenue by him generated.

Such formalization for the revenues makes clear that G is the operator to find in order to build a forecasting model; in fact the revenues follows as *a posteriori* sums and reaggregations of the punctual revenue produced by a specific customer which is formalized as a function, by the mean of G, of its features.

Having defined the goal, a supervised approach has been chosen to approximate the operator G; according to the usual pipeline formalized in the previous chapter, what follow are the operations implemented to reach the goal alomg with information about the data used and the observations made during the development phase.

3.2.3 The data

The first step of the project has certainly been the analysis, with the Data Science team, of the available data; the different features which possibly characterize the data have been studied in order to choose the most suitable ones for the purpose.
Being the goal to estimate the revenue $y^{C_i}_{\text{REV}}$ and doing that by using a supervised approach for estimating the operator G, the most logical choice has been to use data relative to every customer in which different features were

corresponding to a real produced revenue in the past, in order to use it as *ground truth* for model training phase.

In this perspective, historical data from 3 past fiscal years[1] of customers who bought add-on items (the assets we want to build the forecast model for) have been used; it must be specified that the data used is not from a singular source, but it is the result of a merging methodology between different sources. These data represent the customers by the mean of different features and collect the expenses they made (the y variable to use), which represent the revenues to forecast.

As previously specified, because of the *NDA* existing between the company and the University, it's impossible to share real data; from this moment on, any data sample shared must be considered, as partially edited, encoded or hidden.

Because of the nature of this kind of business problems, in literature the customer data usually consists in many different kinds of predictors:

- *Demographics*: this kind of features are usually taken into account as first level discrimination predictors.
- *Spending power related*: this set of predictors is related to possibilities of the customer, in terms of spending power.
- *Contextuals*: these represent the context in which the customers made the expense.

To better understand the meaning of each category of feature, let us consider the example of forecasting the revenues made by selling airflights tickets; in such a scenario, important features to consider, when observing a candidate passenger, to estimate his/her future expenses, could be demographic features, such as *Sex* or *Age*, features related to the spending power, like the *Average salary*, or contextuals, like the *services offered by the plane* he/she will fly on.

In a similar scenario, having such different types of features, it's pretty common that the instances of many of them are categorical, with implicit ordering or not.
This fact occurred in our project as well and had a great impact on the model choice, as will be later showed.

[1] A *fiscal year* or *financial year* is a twelve-month period used to calculate annual financial statements in businesses or organizations.

3.2.4 Preprocessing

Right after the extraction and the merging phase, the whole dataset counted approximately 3.5 millions of records, and every record was characterized by 86 features, selected to be candidate predictors for the model.
As often happens in production environment, and as presented in the previous chapter, the available dataset was affected by different issues and criticity and it needed to be preprocessed before being able to use it in the training process of whichever type of Machine Learning model.
Preprocessing phase consisted, according to the common approach as introduced in Chapter 2, in:

- *Cleaning data*: Empty values have been filled and wrong ones have been corrected; when no recovery procedure were usable, the samples have been discarded.
- *Feature engineering*: The whole set of techniques and methodologies in order to get the maximum amount of information in an optimal form has been implemented; new predictors have been obtained from the original ones and the encoded and aggregation phases have been implemented; also, a subset of features has been selected from the original set.

After this whole procedure, the data consisted in more than 3 million samples with 9 features, 7 of which were encoded version of categorical ones.

Finally, the data so obtained has been splitted into two different sets, the *Training* and *Test* respectively, in order to proceed with the training phase. In particular, following a very common approach in the literature, the training set is chosen two be $\frac{2}{3}$ of the dataset while the test set is represented by the remaining $\frac{1}{3}$. In this project they consisted in 2 millions and 1 million samples respectively.

3.2.5 Model choice and validation

Model candidates

Choosing a model means to select an appropriate type of architecture among different ones (Decision Tree, ANN, etc.) and eventually, once chosen it, to find the correct set of hyperparameters (in the case the model needs them) in order to achieve the best possible solution, in the sense previously specificied.

In this particular scenario the most important factor to consider was the rate of categorical features with respect to the total; specifically, with 7 categorical features on the total of 9 it was crucial to choose an architecture

which could handle them well.

Another driver in the choice in order of importance has been the need of a certain degree of intepretability of the results and the conclusions.
In fact, for working in a business context the model is required to be reasonably comprehensible, at least in the results, in order to let the decision makers be able to use field experience to see beyond numbers and find hints or threats in what emerges from the calculations and mathematical forecasts.

Considering these reasons and after some analysis a tree based model, specifically a *Random Forest Regressor*[2], has been chosen to be the ideal candidate. In fact, such a model is pretty easy to interpret, being based on tree structure and having a feature importance KPI, and can handle very well categorical feature, almost regardless the chosen encoding.

With it, a *Linear Regressor* and an *Artificial Neural Network* model have been chosen in order to have some benchmarks to compare the results with. The first is even simpler to interpret than a Decision Tree based model; on the other hand it's too much semplicistic from a modelistic point of view.
An ANN, or MLP equivalentely, on the contrary can model much more difficult relations but it is like a *black box* when it comes to intepretation; more, the architecture needed to reach good results (number of hidden layers and neurons) could make it computationally heavy.

Hyperparameters choice

In Chapter 2 we introduced this phase as a very important one to pay attention to in order to build a performing model, and we already introduced the concepts of some different techniques to perform it.

In this scenario the efforts have been spent in the research of the parameters regarding the *Random Forest Regressor*, the *Linear Regressor* and the *MLP*; in particular, for every model a *greedy* search within the hyperspace has been performed.
Regarding the *Linear Regressor* the attention focused on the research of the regularization parameter *alpha*; the *number of trees in the forest*, the *minimum number of samples required to be at a leaf node*, the *minimum number of samples required to split an internal node* and the *number of features to consider when looking for the best split* are fine tuned in building the *Random Forest Regressor*; finally, the *number of layers*, the *number of neurons per layer* and the *activation function* are fine-tuned by mean of such a greedy research regarding the *Multi Layer Perceptron*.

[2] *Regressor* because the target value is a real number and not a category.

Cross validation

In the training phase a cross validation strategy in order to find the best set of hyperparameters has been implemented. In particular a K-folds cross validation strategy has been implemented (refer to Section 2.7.2 for details). While performing the cross validation however, the sampling has not completely been random; in fact, particular attention has been paid in order to preserve the overall chronological structure of the data; in details, it has been considered a moving time window of 1 year and the data, regarding 2017 and 2018, has been splitted in the different folds according to the *train* fold always preceding the *test* one.

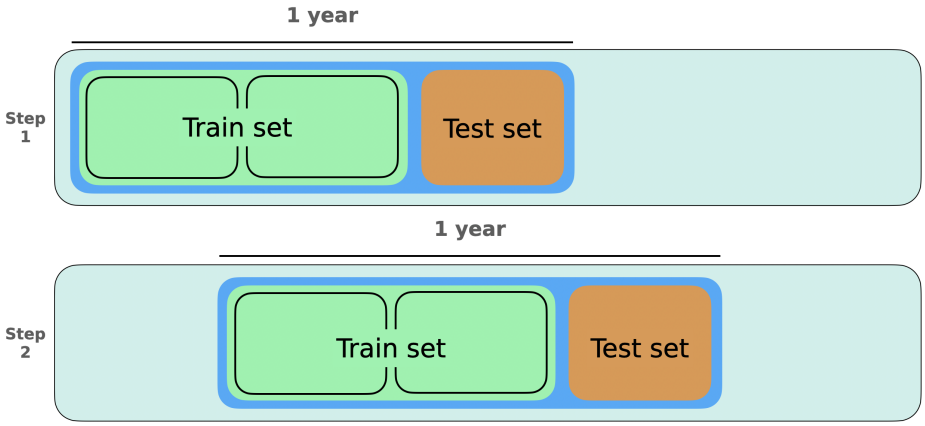


Figure 3.1: A custom cross validation strategy has been implemented

Performances comparison

As conclusion of the comparison phase, once found the best set of hyperparameters according with the methodology previously introduced, the three different architectures have been trained on the whole train set and tested on the *test set*.
The evaluation metric for this phase has been chosen as *Mean Relative Error (mre)* defined as

$$mre_G^{(X,y)^{\text{Test}}} = \frac{\sum_{i=0}^{N} \frac{|G(x_i^{\text{Test}}) - y_i^{\text{Test}}|}{y_i^{\text{Test}}}}{N}$$

in which G is the estimated operator and $(x_i^{\text{Test}}, y_i^{\text{Test}})$ represents the test set of N elements which the error is computed on.

Model typology	*mre* value (%)
Linear Regressor	21
MLP Regressor	18
Random Forest Regressor	**10**

Table 3.1: The different *mre* computed for each candidate model on the test set.

Table 3.1 shows the results obtained on the test set for the different models trained with previously introduced methodologies and shows how the Random Forest performed better than the others, becoming the chosen predictor for the project goal.

3.2.6 Before deployment evaluation

In the previous sections the objectives of the project have been introduced together with the different procedures used in preparing the data for the process and their use within a described, evaluation and training phase for a suitable predictor.
This section aims to show some specific results, according to some metrics, about the performances of the previously trained model obtained on historical data. These results represent the benchmarks to compare with the live performance data retrieved in the operative phase of the model itself.

To compute the above mentioned benchmarks, a Random Forest Regressor has chosen to be the model to use in deployment phase and it has been trained on the training data represented by 2 fiscal years samples and with the hyperparameters previously found; the test set has chosen to be constituted by 1 fiscal year data next to previoius ones.

Global error

While the *mre* showed in Table 3.1 was computed as a global error related to all the customers in the test set, it is now interesting from the business point of view to look at *mre* on specific aggregations of data; in particular, considering only customers in the test set which bougth a basic offer bundle E_k to be *consumed* in particularly stable markets, the relative error $re_G^{E_k}$ can be computed as

$$re_G^{E_k} = \sum_{E_k} \frac{\sum_i^{N_{E_k}} |G(x_i) - y_i|}{\sum_i^{N_{E_k}} y_i}$$

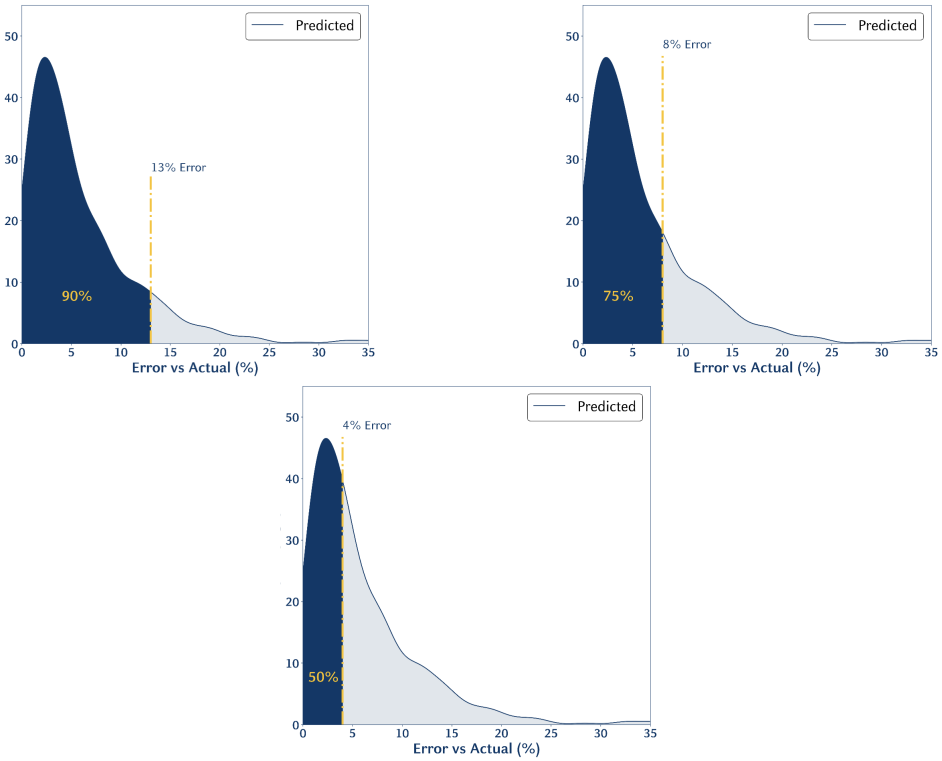
From it, it can be defined a *Weighted Mean Relative Error* (*Wmre*) over the bundles as

$$Wmre_G^{E_k} = \frac{\sum_{E_k} W_{E_k} \cdot re_G^{E_k}}{\sum_{E_k} W_{E_k}} = \frac{\sum_{E_k} W_{E_k} \cdot \frac{\sum_i^{N_{E_k}} |G(x_i) - y_i|}{\sum_i^{N_{E_k}} y_i}}{\sum_{E_k} W_{E_k}}$$

In order to compute a fair business analysis, knowing that different touristic bundles have different importance, weights representing this (quantifiable) importances have been introduced to calculate the *Wmre*.
With such weights, the *Wmre* computed is approx. 6%, reflecting an improvement with respect of the average computed on the whole test set. This fact comes from the choice of focusing on a more stable market; when observing the overall result instead, also less stable scenarios are considered in the analysis, leading to a lack of pattern and impacting on the affordability of predictions.

Percentiles analysis

The second interesting analysis that can be performed using the relative errors above defined is looking some percentiles of the relative errors as showned in the following figures



As shown in figures above, it can be seen how more than 50% of the touristic experiences have relative error, in respect to the real revenues, that is minor than 4% while the 75% of them have *re* minor than 8%.

Feature importance

Finally, having chosen a Random Forest Regressor as model, the Feature Importance data was available.

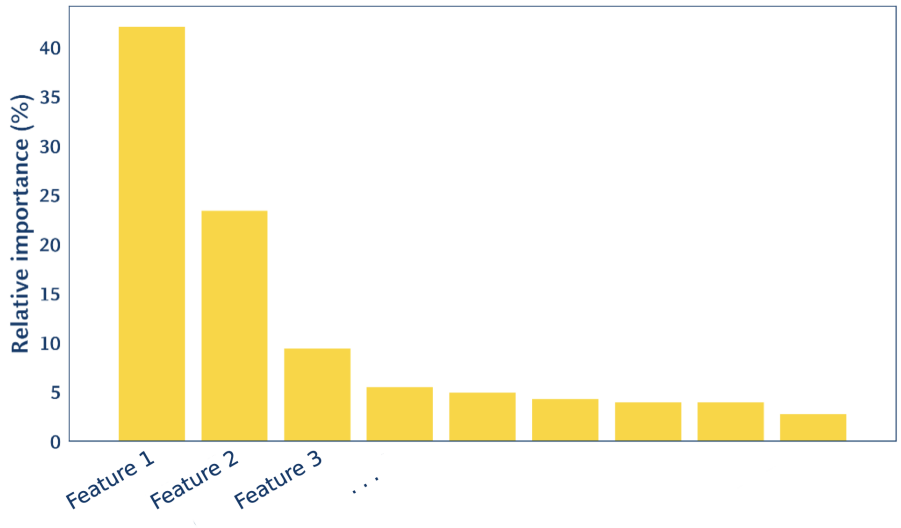


Figure 3.3: Here the plot of the feature importance, in percentage, of the Random Forest Regressor trained.

Feature Importance of the model, plotted in Fig. 3.3, shows the prominence of 3 features above all, suggesting the high predictive power of those 3 with respect to the others.
In particular, even if it is impossible to share the nature of the variables, the importance of the most impactful feature has been confirmed by a business expert in an *ex post* analysis.

3.2.7 In production

Previous sections gave the lecture the basis to understand the business framework in which the project took place, the goals of the project itself and the methodology adopted in order to reach such objectives. It has made clear the data driven approach and the algorithmic side of the matter, providing also interesting analysis on the performances of the built model in the pre deployment phase.
From a business perspective it is clear that all these phases remain almost

useless if the tools and knowledge which emerge from them are not inserted in a suitable deployment phase. So, in order not to confine all the work done only to the *R&D* branch, what followed was the study of an appropriate methodology by which making the built model accessible and usable by some users, who could provide us feedback about its performance in the production phase.

Model deployment

The methodology to make the model available to users has been chosen to be as a web platform, available in the company intranet. This platform, built with the help of pre existing libraries, has been designed to have a very simple and easily understandable user interface which allows to filter by time and other variables useful for the business.
Once selected the filter combination, it starts the prediction by the mean of the prebuilt model, using the data appropriately processed in order to deal with exceptions and missing inputs.
Finally, the results can be downloaded in the selected format (*.csv*, *.xls*, and others).

In this test phase of deployment, the window of prediction has been chosen to be the 2020 fiscal year.

Performances evaluation and conclusions

After the deployment started, a monitoring phase have been implemented, by which a performance evaluation has been made possible.
This section contains the data summarizing the performance of the model in the production phase relative to fiscal year 2020.

At this point it is necessary to make a specification: in fact, while originally the time window for the test phase should have been relative to the whole 2020 fiscal year data, due to the unpredictable and unexpected SARS-CoV-2 pandemic outbreak the performances have been monitored only for the fiscal months of December 2019, January 2020 and February 2020, being this timeframe the only ones business results were available of.
For this reason, it is possible to provide only a partial analysis on the results relative to a small number of events.

Global error

Following what done in Section 3.2.6, the *Wmre* in the considered scenario of deployment, using again the weights as done in the proposed methodology, has been approx. 14%.
Comparing this error with pre deployment one, it is clear how this improved

from training and evalutaion phase to the in production stage; nevertheless, this result was pretty good from the business prespective. It has to be underlined also that this result has been obtained in a 3 months time window and, for a fair comparison, a one year time window would have been needed.

Percentiles analysis

In this section the percentiles analysis proposed in Section 3.2.6 has been done on the data and their errors deriving from the production stage of the model.

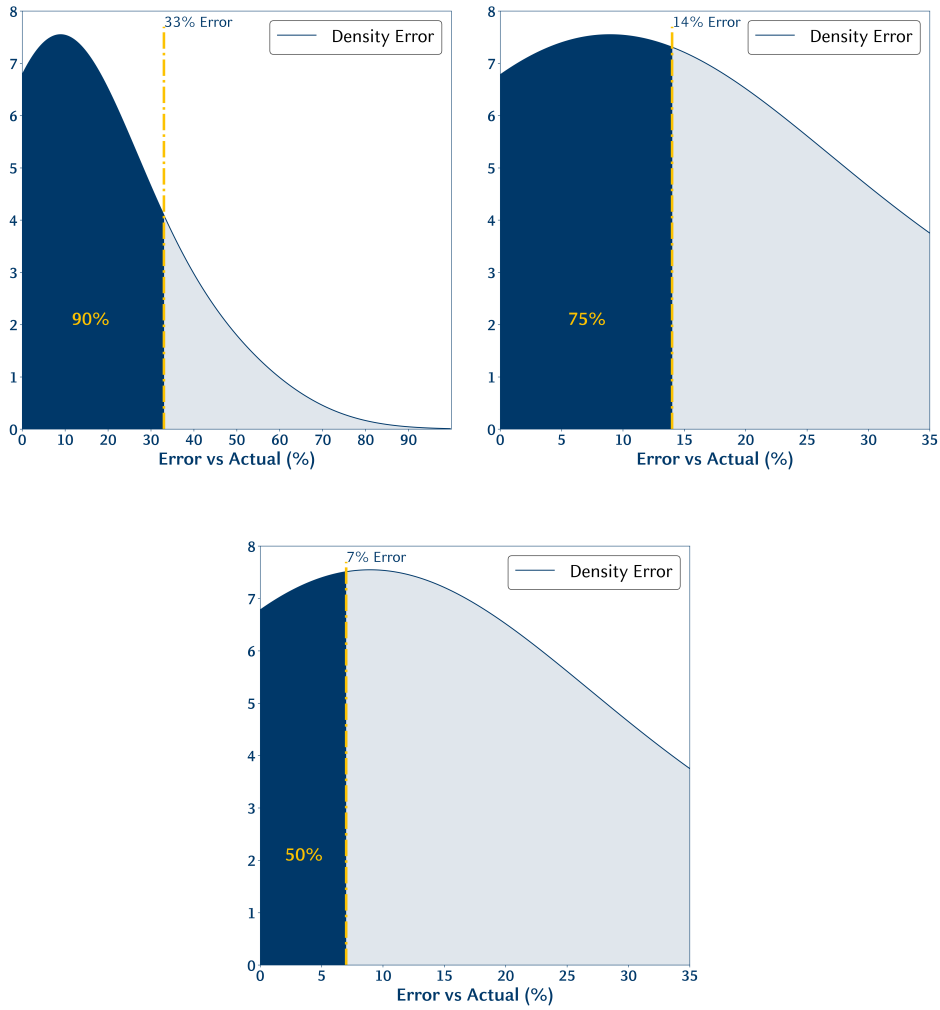


Figure 3.4: Different percentiles of the relative errors over the touristic experiences relative to deployment phase.

Again, as happened for the global error and in accordance with what expected, the performances in the production stage have been slightly worse

in respect of what noticed in the pre deployment phase; still, as already said, they were deemed extremely good.

3.2.8 From forecasts to actions

Before concluding the chapter, the last thing to be analyzed is the outcomes that a study like the one just presented provides.
In order to do that, let us assume to have built a perfect forecaster which always predicts the perfect value of different revenues; in such a scenario, until the built predictor confirms sales expectations for the future then no issues arise; however, if anomal forecasts begin to appear, then another branch of possible issues opens.
In fact, depending on the anomal forecast trend, the reason for the deviation of it from the expected one can be searched; while in theory sometimes it could be true, in the practice it is usually too much simplicistic to assume that the anomaly could be caused by one factor only and it is much more realistic to hypothesize that it is related to different determinants.
Assuming that, the safest way to proceed is to reduce the possible *causes* in order to find the set of candidates which could impact most on the forecast. To do this, some information and hints can be provided by the chosen model which constitutes the forecaster; for example, in our scenario this information can be extracted by observing the Feature Importante in Section 3.2.6 which shows how 2-3 features impact the most in making the prediction (by the model previously built); then, it is often usual to interact with experts and decision makers in order to use their sector experience to complete this step and identify the candidate responsibles of the trend deviation.
Then, on the basis of the *determinants* previously identified, some actions must be hypothesized and their impact quantified; to do this not only it is necessary to find the determinants, as already done, but also it is necessary to estimate the sign of their impact on the forecast (positive or negative) and quantify it.
To accomplish this task, in our work we spent some effort to build a simulator which allowed the user to, in fact, simulate the impact of some actions based on the hints provided by the model and its feature importances; by the way, no other details will be provided about this topic being it not the goal of the discussion.

Only when each one of the previous tasks has been accomplished then the correct (estimated) set of actions can be implemented to react to the change in the trend.

3.3 Destination discovery

This section regards the second project we developed in our work and, as previously mentioned in the introductive section of this chapter, it had the goal of estimating tourist attractiveness for a list of many different locations.
That is not an easy task and could be faced by using many different approaches depending on the hypoteses made and the available resources; different indicators and methodologies could be involved but, with respect of forecasting sales as developed in the previous section project, this is a completely different problem both in the approach and in the level of punctual information involved in it.
In fact, if estimating future expenses of customers in relation to different items and frameworks provides punctual *customer-centric* predictions which can be later aggregated into much more general values, dealing with tourism attractiveness has to be approached with more high leveled dynamics since the beginning, or at least it seems pretty reasonable to do it.

In order to properly approach the topic, we need to fully understand and define what is the *tourist attractiveness* of a location in our framework.

Most authors naturally agree on recognizing tourism attractions, defined as features which have an impact in motivating people to visit specific destinations, as key variables to define the touristic value of a location [44][45], which determine not only the direction of the tourism development of the location, but also the intensity.
For this reason, choosing a representation for tourist attractions becomes the first step in order to accomplish the main task; for doing that different data sources could be chosen, depending on the depth and the wideness of the analysis to implement, and different methodologies to estimate the tourist value of a location; for example in [19] people answers related to specific location are collected by questionnaires and used to define an *IDA*, or *Index Destination Attractiveness* of a location; in [21] the answers are instead combined to define a linear model and highlight the impact which different features have on the tourist attractiveness of a location.

In our scenario, however, a more high level approach has been implemented; in fact the willing to consider in the analysis an extremely wide range of localities, possibly very distant from each others, made it not practicable the usage of questionnaires to get information; on the contrary, it seemed reasonable to rely on online sources containing reviews made by travelers all around the globe in order to get tourist information related to the different locations and use them inside a custom methodology of analysis.
This approach let us create a methodology valid both for localities already

known by the company, and therefore of which it already has collected some data in the history, and for new localities, for which data is certainly missing inside the company itself.
In this way, by relying on a representation based on external data, the data have been made consistent for any location, known or unrelated to the company business.
From a business perspective, in which the company is always looking for new business chances to exploit and growing trends to follow in order to expand and improve its business and enlarge its offer, isolating and potentially exploiting brand new scenario and growing interest trends is surely the main interesting aspect.

Going deeper in the problem definition we, in accordance with company internal departments, initially organized it into two different operative steps, considered as project milestones: the first part of the project had to be a pilot aimed at defining a certain metric to rank locations according to their current tourist attractiveness; the second operative step had to be the extension of what done in the first part in the case of future tourist attractiveness of the locations.
Formalized in this way, the *forecasting* task would have entered the process in the second part, while the first would have been the definition of a suitable comparison metric for the tourist attractiveness as it is today; in other words, in the first part we would have needed to define the tourist attractiveness as it is today and then use this information to define a suitable metric to compare location on its basis, while in the second part we would apply the very same metric on the forecasted touristic attractiveness, once we obtained it.

Unfortunately, as it has been already said in the introductive chapter, we could only manage to complete the first phase of the project, due to the SARS-CoV-2 pandemic outbreak. However, the work produced would be surely useful for anyone in the company who would face the second part and complete the project.

Focusing now on the first part, the objective has been the study of the tourist appeal of different cities of the globe, being them related or not to the actual business of the company, and define not only a metric to sort them in order of importance, but also to compare them and to find similar cities in terms of tourist offer and typology.
In this direction the methodology defined and the conclusions that followed will be provided; in this work, a lot of efforts have been spent in data acquisition and processing, being these tasks essential to choose a data structure for representing information in the most efficient way with respect to the goals of the project.

Before going into details, it is necessary to provide the framework of the project, the assumptions made and some information required to better understand the whole process which will follow.

3.3.1 Preliminaries

Studying the tourist quantities related to a location is a very difficult task in general; this is because of the many different variables that are involved in the framework and their quantification.
This work moves in the direction to approach such task through a data driven approach based on different preliminary assumptions. The first and most important fact to note is surely that this work has been considered as the first evaluation step to a much bigger work; it had the goal to evaluate if such an approach could be valuable for the company and to highlight issues in the methodology, in order to clearly define different key points for the future project.

Having said that, for such a task the most important asset is the *tourist related* data used in the process; if until some years ago the most popular way to get this type of data was by interviewing people, nowadays the growing number of also growing online data banks (in particular constituted by platforms for tourist reviews or related) or social networks represent the major resource for such analysis. Along with the diffusion of digital tourist information, the research which involved has grown as well, with examples of attempts in using *SNA (Social Network Analysis)* [68] within tourism related researches.

Clearly, depending on the source, the data related to the very same location could be different in terms of mass and typology, possibly leading, inside the same analysis, to very different results; this fact forces to pay extreme attenction to the choice of the sources. Other than that, within the digital revolution which is overwhelming companies, data is becoming one of their most valuable assets; for this reason, it can be very difficult in different frameworks to access data that would be perfect for a certain task because it is simply not publicly available.
Finally, even though it has been certainly a necessary step in regulamentating data usage, the GDPR implementation of 2018 made it even more delicate for companies to deal with personal data.

Another important aspect that needs to be considered when dealing with this type of data and the tourism itself is the strong time dependence which characterize this framework.
Talking about data, especially the ones obtained from an online data bank,

it is clear that they are *time series*[3] in which different features and values are recorded at different times; examples of such data are the tourist reviews provided by users on different online platforms sequentially in time.
Thinking instead about tourist activity from a general perspective, it is also evident that it changes a lot for the very same location in the different moments of the year; for example, the tourist activity in the Costa Azzurra in the month of August is very different from the one in December.

Finally, it is underlined the amount of almost unlimited definitions that could be possibly given of *tourism* and *tourist activity*, depending on the variables considered and the assumptions made.
For this reason, instead of giving one at the begininning of the discussion, it has been decided to give the definition after the collection and the study of the available data using the variables that define it.

Because of different business related motivations, the analysis focused on cities located in the European territory.

Approaching the above mentioned task, it is clear that the concept of tourism is strongly related to locations and, in particular, their geography and tourist attractions.
More, it is usually common to consider tourism appeal of macro aggregations of geographic locations; for example, by analyzing the tourist activity of Genoa we usually refer to the tourism related to the city of Genoa itself and to a collection of sub locations of the city which are in the geographic neighbourhood of it, like Recco and Savona for example.

For this reason, the first step in the process has been the definition of the locations to study tourist activity of, thought as aggregations of geographically close locations. To this phase followed the choice of suitable data and the acquisition of such data from the suitable sources. Then the preprocessig, similar in the concept to what presented in Chapter 2, has been performed before the metric definition phase began.

The next sections represent a focus on all the different parts related to the operations and the data involved in the process.

3.3.2 The data

As previously mentioned, the data are the most important asset of the analysis; this is because everything that follows is strongly related to them and an eventual *bias* would drastically impact on the conclusions.

[3]In data science, it is inteded as a series of data samples indexed in time order.

In this work, different types of data, deriving from different sources were needed.

The representation

As anticipated, a hierarchical structure has been chosen to represent the data.

The goal has been to define a list of leading cities, each one linked to others in their catchment area, being themselves linked to a set of tourist attractions; with such a rapresentation the tourist activity data can be obtained by aggregations on multiple levels.

This representation needs on one hand of *geographical* data, that allows to create the hierarchical part of such structure, creating links on a geographical distance based criteria; on the other, it needs *tourist activity* information to populate the structure just defined.

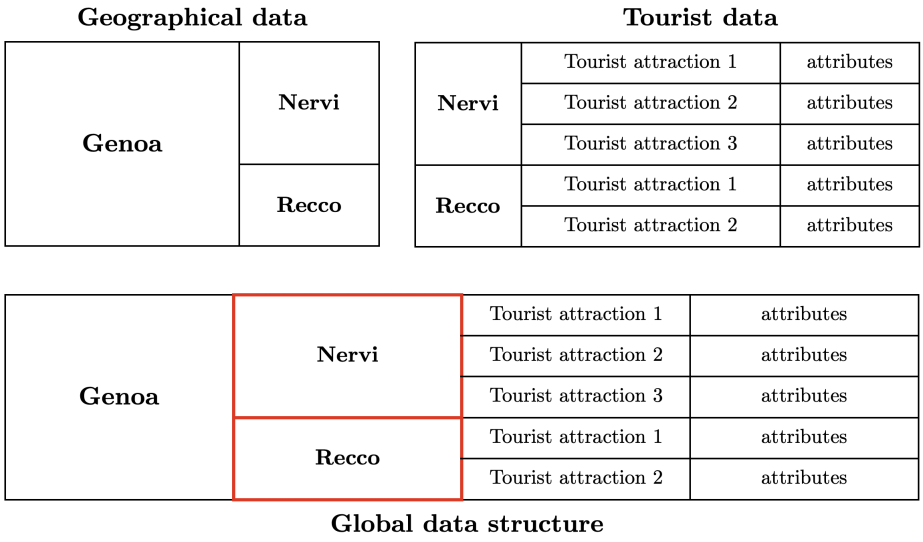


Figure 3.5: An example of the theoretical data structure.

After some time spent on online search and evaluation, 3 different variables have been identified for summarizing tourist information of the attractions, once chosen the online platform of reference; more, they are addressable to two different categories in respect of their role:

- ***Tourism Magnitude***: Variables in these clusters are related as the overall importance of tourist attraction. They are:

  - *Average rating*: it represents the average score, usually in a 0 to 5 range, obtained by people reviewing it.

  - *Number of reviews*: it is the number of reviews the attraction has received since the creation of the online evaluation platform.

- ***Tourism Topology***: Variable in this category is responsible of discriminating between attractions in relation to the type of people they are most likely for.

  - *Type of attraction*: it is a list of tags, giving a summarized description of the attraction (example are *food*, *beach*, *castle* etc.).

These variables have been chosen to be the attributes for the different tourist attraction.

The datasources

In the project, different online data sources have been analyzed and evaluated and the choice has been made on the basis of three different parameters.

The first is *reliability*: the data source, in the nature of the provided records, has to be as much reliable as possible in order to reflect the real scenario.

The second parameter is *numerosity*: in order to have a statistical importance, the number of samples has to be significant.

Finally, the last parameter is *retrieval methodology*: even if not related to any theoretical reason, it needs to be analyzed as well as the previously cited parameters when chosing the most suitable data sources.

Given these parameters, the touristic data source has been identified in Google Places while the geographical one has chosen to be GeoNames[4].
The choice of using two different sources came from the following observation: while Places is surely reliable, complete and with an easy retrieval procedure (by the mean of dedicated APIs), it unfortunately lacks of the tool to get a list of different cities; this information is instead present in the GeoNames platform, which is again easy to scrape being reliable and complete at the same time, but missing instead any kind of tourist information.
For this reason, merging data deriving from these sources was evaluated to be a valid choice.

[4]https://www.geonames.org/about.html

Scraping methodology

Being the data deriving from different sources, a custom and sequential scraping procedure has been implemented.
In particular, the first step has been the selection of the city to explore and populate with all the list of tourist information from the GeoNames database, complete with a set of coohordinates identifying the cities.
Secondly, given this list of cities and coohordinates, these last attributes have been used as keys to explore the tourist activities by the mean of the Google Places APIs, building the initial data structure.
Last note to share is that Google Places allowed the retrieving of only a maximum of 60 attractions for every location searched; this restriction clearly limits the generality of the results, but the number of collected samples still permits to perform some interesting analysis.

Collected data

Using the above explained automated methodology, the data has been collected from the chosen sources.
Specifically, the initial data structure was constitued by 108.000 samples, collecting more than 33.500 tourist attractions, relating to approx. 1500 locations linked to 300 cities.

City	Location	Attraction	Average score	Tags	Number of reviews
Palermo	Palermo	Quattro Canti	4.6	['history']	7874
Genoa	Genoa	Acquario di Genova	4.5	['art','museum','family', ...]	41700
Genoa	Camogli	Basilica di Santa Maria Assunta	4.6	['art','museum','history','place of worship']	135
Napoli	**Caserta**	Castello di Limatola	4.3	['castle','museum','history']	4908

Figure 3.6: Samples from collected data structure.

3.3.3 The process

The choice of a hierarchical structure allows to rank data on different levels of aggregation. In fact, different aggregations can be performed when grouping by *Location* or by *City*, creating the features as counts and averages.
In this direction, the basic unit for the two possible aggregations is represented by the *attraction*, with its *tags*, *number of reviews* and *average score*. Geographical information do not take part in the comparison because only involved in the aggregation phase; for this reason no further exploratory analysis is required on this type of data.

Ranking attractions

Before approaching the problem from a numerical point of view, it needs to be understood what, ranking different locations in terms of their tourist importance, means.
One of the possible definitions that can be assigned to ***tourist importance*** of a location, and the one in this work choosen, is *the mass, in a one year time window, of people visiting the location and the overall sentiment they express after the visit.*
In this view, the key to sort the locations by is the tourism importance as just defined, and the variables that define this importance are the number of reviews received, representing the mass of visiting people, and the average score, representing the overall sentiment produced.

Given this definition, what follows is the analysis of the variable R and S in order to use them aiming at the definition of a ranking metric.

Being the goal of this pilot to be able to compare cities, the first step has been analyzing the distributions of R and S.

By looking at the scatter plot in Fig.3.7 between these distributions, without any preprocessing performed, some information clearly emerge.

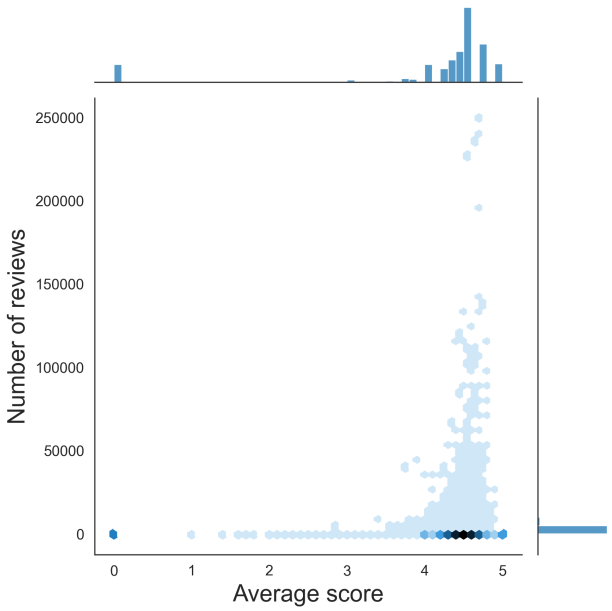


Figure 3.7: Scatter plot with densities between the *Number of Reviews* and the *Average Scores.*

First, a strong imbalance in the given scores with the predominance of

high scores in respect to lower ones emerges, in particular with scores in the range going to 4 to 5; this clearly reflects the general behaviour of people who give score to things especially when they like them.
Secondly, R is almost always lower than 10000 with some *outlier* locations which have between the 10000 and 250000 reviews.
It is clear then that neither R nor S are normally distributed, fact that needs to be dealt with being the goal of the project to use these measures for comparison reasons; for this purpose, the need is to achieve as much normally distributed measure sets for both the variables R and S as possible.

Processing R

By looking at the plots in Fig.3.8, it is evident that the data is skewed, with a huge number of small values and very few high ones, making the distribution similar to a log normal density.

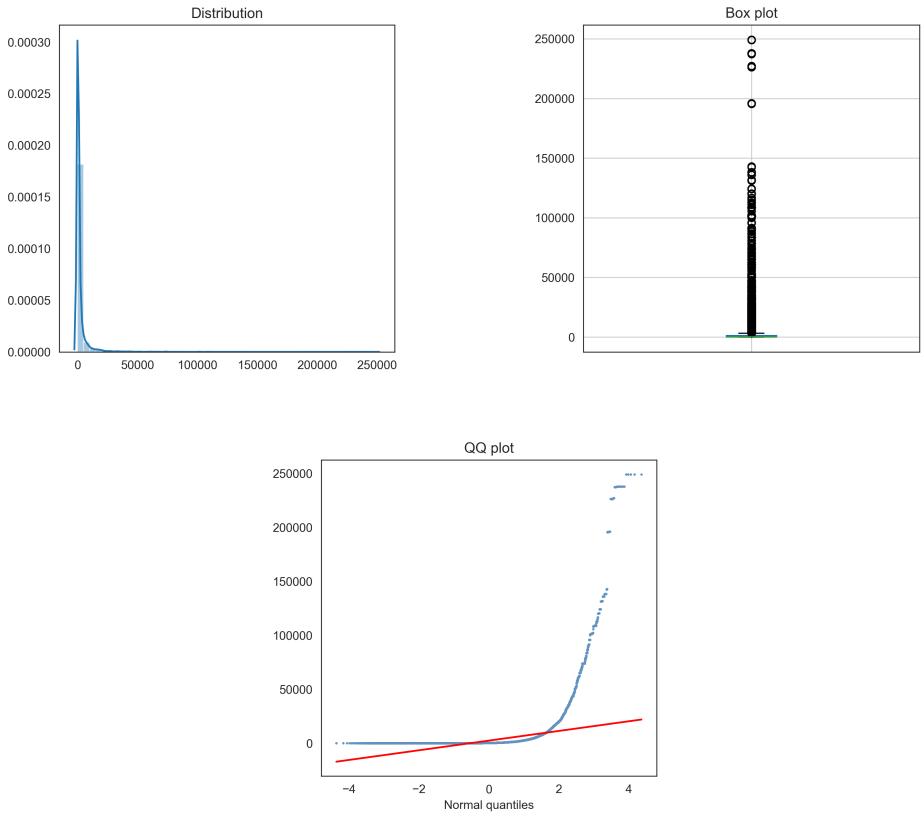


Figure 3.8: Up left the data distribution; up right the box plot and down the QQ plot of the density.

The first operative task has been the search and removal of attractions

with no reviews or just one, because of their lack of importance; if present, duplicated attractions are removed.

Secondly, following the hints provided by the plots above, a logarithmic transformation has been applied on the data.
This procedure lead to the data plotted in Fig.3.9.

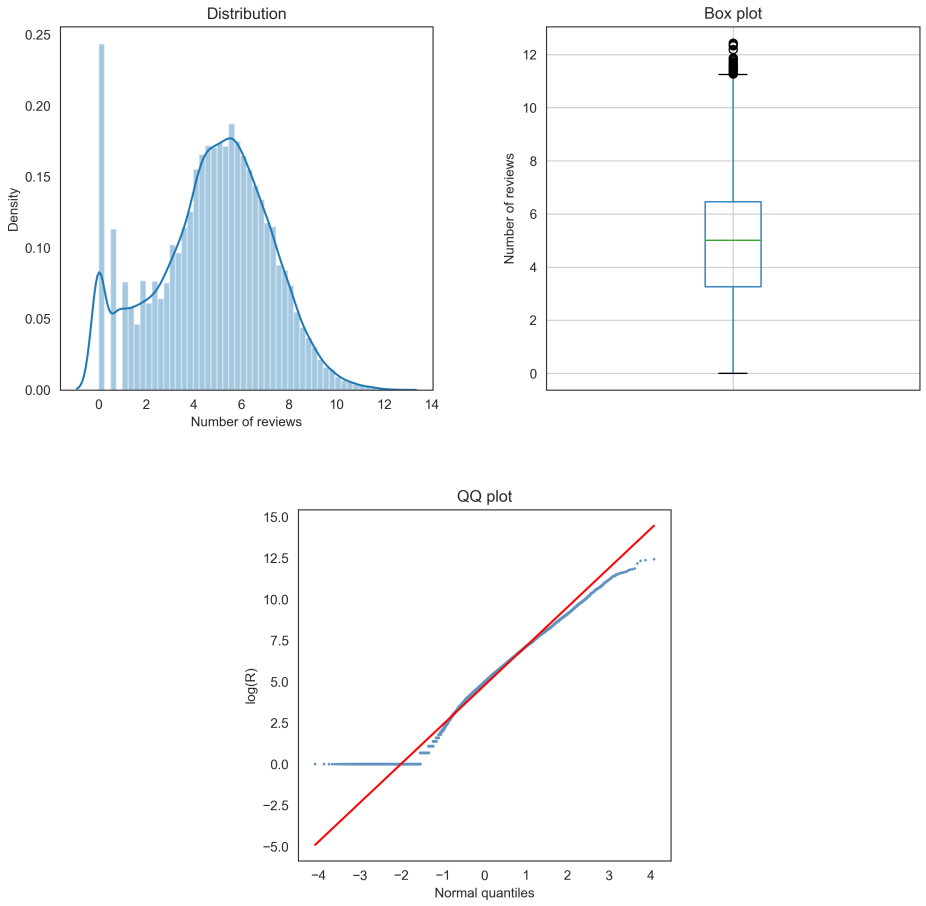


Figure 3.9: Up left the data distribution; up right the box plot and down the QQ plot of the density.

As it can be observed from the previous plots, after the transformation the data is much more similar to a normal distribution and much better distributed.

Processing S

When it comes to average scores, as already mentioned, it emerges the tendency of people in giving only high scores, like shown in the plots of Fig.3.10.

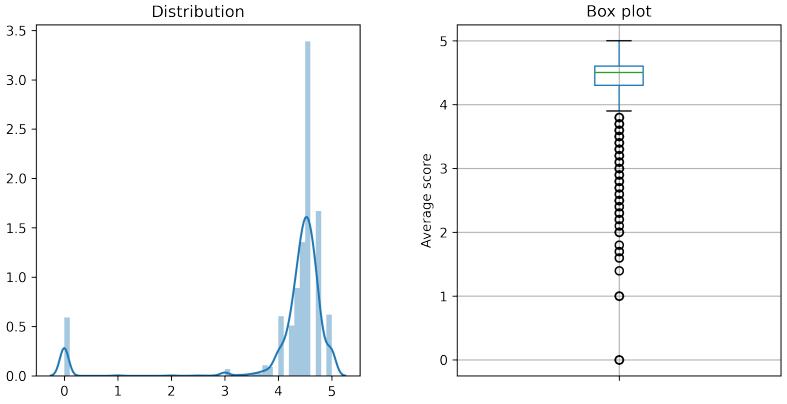


Figure 3.10: Distribution and box plot of average score data.

What emerges from these plots is that, if looking to scores above 3.9 approximately, the situation is the one expected.
The removal of attraction having 0 as average score and duplicated, again, anticipated the main transformation performed, that has been the mapping of values lower than 3.9 into 3.9.
These processes produced the new much more balanced data plotted in Fig.3.11

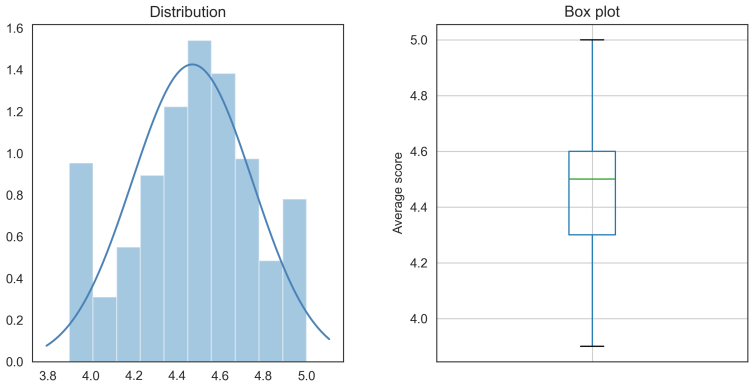


Figure 3.11: Distribution and box plot of average score data after transformation.

By comparing now the cross plot of the original data, as in Fig.3.7, and the one relative to the transformed data, it is evident how the situation improved and the data is much better distributed, with the two densities that are more similar to normal densities.

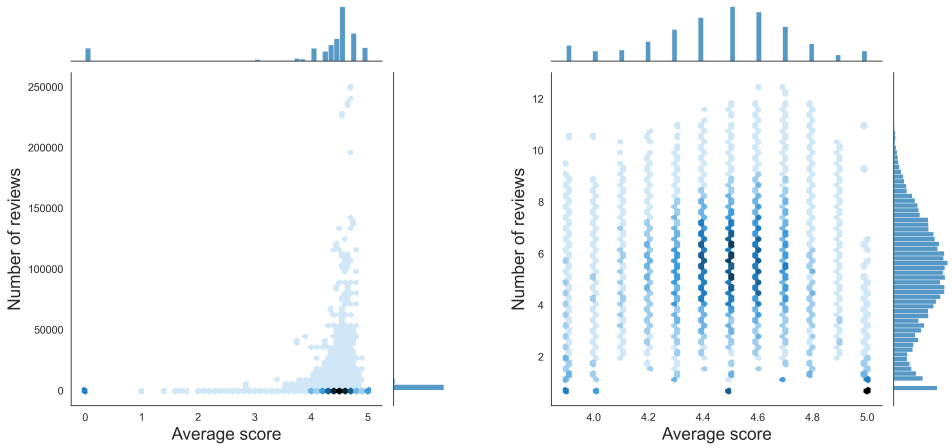


Figure 3.12: Comparison between plots of original data (left) and transformed one (right).

Ranking methodology

After the processing phase, another variable has been created for every attraction, representing its final *tourism score* (T_s):

$$T_s(\text{attraction}) = \hat{S} \cdot \hat{R}$$

where $\hat{S}$ and $\hat{R}$ represent the new score and number of reviews variable, as previously obtained.
Finally, the so obtained score is normalized into $[0, 1]$ range, defining the final normalized score $\tilde{T}_s$ and producing the ranked list of attractions.

Attraction name	**Tourism score normalized**
Trevi Fountain	**1**
Colosseum	**0.996**
Louvre Museum	**0.979**
...	...
Lavian Kotiseutumuseo	**0.000**

Figure 3.13: Examples from ranked attractions table.

The tourist score $\tilde{T_s}$ just built is attraction related and deals only with the impact of tourism, not with the type of attraction.

3.3.4 Similarity analysis

The next step of the project aimed at defining a metric to take into account also the type of tourism in the comparison.

Processing city related data

In terms of data structure, at this point it was composed by different attractions with a tourism score $\tilde{T_s}$ attribute, a list of tourists' tags associated and linked to possible locations and cities on the basis of the geographical distance.
In order to perform a fair analysis of the typology, every attraction has been duplicated as many time as the number of tags representing it, creating samples which differed only for the type feature.

Attraction name	Tags
Trevi Fountain	History, Family

→

Attraction name	Tags
Trevi Fountain	History
Trevi Fountain	Family

Secondly, the number of tags has been drastically reduced; in fact, the detail level is far away too deep with respect to the dimension of the data. The original set of tags, containing elements like *museum*, *park* and *church*, has been reduced from 23 elements to 6. This reduction has been accomplished by a filtering phase, removing non-tourist related tags, and a mapping phase, in which micro tags have been mapped to *macro* tags.
In particular, the selected macro tags are:

- Nature (N)
- Culinary (K)
- Entertainment (E)
- Adult Entertainment (K)
- Family (F)
- Art (A)

Micro Tag	Macro Tag
Museum	Art
Casino	Adult Entertainment
Acquarium	Family
Zoo	Family
Stadium	Entertainment
...	...

Figure 3.14: Example of some instances of the map from micro to macro tags.

Then, a city aggregated data has been built. In doing so, the aggregation function has been chosen to be the sum of the tourist score $\tilde{T}_s$. Specifically every city c has been represented by 6 variables, one for every macro tag, as follows:

$$c = \left(\sum_{\substack{a_i \in c \\ a_i \in A}} \tilde{T}_s^{a_i}, \sum_{\substack{a_i \in c \\ a_i \in C}} \tilde{T}_s^{a_i}, \sum_{\substack{a_i \in c \\ a_i \in F}} \tilde{T}_s^{a_i}, \sum_{\substack{a_i \in c \\ a_i \in E}} \tilde{T}_s^{a_i}, \sum_{\substack{a_i \in c \\ a_i \in K}} \tilde{T}_s^{a_i}, \sum_{\substack{a_i \in c \\ a_i \in N}} \tilde{T}_s^{a_i} \right)$$

This representation aimed from one hand at representing a city in terms of the different typologies of attractions (expressed by the mean of the different macro tags) that constituted it, and from the other at taking into account the tourist impact of the attraction previously defined.

After defining such variables, two of them, specifically the ones linked to E and K macro tags, have been removed, due the little information adding to the data; finally, missing values in the data have been filled with 0 value, representing a missing tourist category.

Simultaneously, a *density*-like representation for the same data has been chosen; these two representations for the same data are at the core of the comparison metrics chosen, and the reasons will be clear later.

City	Art	Culinary	Family	Nature
Genoa	44.19	2.67	2.83	6.96

City	Art	Culinary	Family	Nature
Genoa	0.77	0.047	0.05	0.133

Figure 3.15: The city of Genoa in the two different representations: up the *point*-like, down the *density*-like.

Comparison metrics

Let us now analyze the motivations that have lead to the choice of the two representations previously introduced.

As already noticed, a specific location has a tourist impact due to the type of attractions that collects, and the magnitude with which attracts people; in other words, balnear locations can attract as many people as a mountain one, still being very different from it in terms of people it attracts; on the other hand, two balnear locations can attract and satisfy people in a very different way.
The goal is to consider these two factors in the comparison metrics and, for this reason, the two representations have been chosen.
In fact, the *point*-like representation allows to take into account the magnitude of the tourist importance of the locations, being instead approximate in discriminating between cities with different topology; on the other hand, the *density*-like representation allows to purely represent the topology of the location, removing information related to magnitude.

Having chosen these representations, what is still missing are the suitable metrics to compare such represented elements.
To compare Euclidean *point*-like elements the natural choice is to use the *Euclidean distance* as comparison metric.
When it comes to compare *density*-like elements, the choice has been more difficult, but eventually the choice fell on the usage of the *Jensen-Shannon divergence*[5]; in particular, the square root of it has been used, being it a metric [64].

For each one of the two introduced metrics, all the pairwise distances have been computed and a 2D array has been defined for every pair of cities, collecting the just obtained distances.
The so obtained symmetric $N_c \times N_c$ matrix, where N_c is the number of cities in the set, collects all the measurements computed pairwisely and representing the two different lens used to compare the pair of cities.

Formally, being D^{M} the just defined matrix, then the (i, j) entry is defined as

$$D^{\mathrm{M}}_{i,j} = \left(E_{\mathrm{d}}(c^{\mathrm{P}}_i, c^{\mathrm{P}}_j), JS_{\mathrm{d}}(c^{\pi}_i, c^{\pi}_j)\right)$$

where E_{d} and JS_{d} represent the Euclidean distance and the Jensen-Shannon distance respectively, and c^{P} stands for the *point*-like represented city while c^{π} stands for the *density*-like represented one.

[5] See Appendix A.

	A Coruña	**Aabenraa**	**Aalborg**
A Coruña	(0,0)	(0.28, 0.079)	(0.46, 0.13)
Aabenraa	(0.28, 0.079)	(0,0)	(0.26, 0.08)
Aalborg	(0.46, 0.13)	(0.26, 0.08)	(0,0)

Figure 3.16: Sample from the matrix obtained in applying the two metrics to pairs of cities.

Finally, the L2 norm has been applied to each array of the matrix representing the pair of computed distances; the matrix obtained represents the final similarity metric between the cities.

	A Coruña	**Aabenraa**	**Aalborg**
A Coruña	0.00	0.292730	0.483672
Aabenraa	0.292730	0.00	0.275485
Aalborg	0.483672	0.275485	0.00

Figure 3.17: Sample from the matrix obtained in applying the L2 norm to the 2D arrays of distancies for evey pair of cities.

Having such representation, to get the list of most similar city to a selected one c_k, the k-th row (or column equivalently, being the matrix symmetric) has to be extracted and sorted; then, the generic i-th entry of such array represents the *distance*, in terms of tourism importance, from the city c_k to the city c_i.

3.3.5 Results and conclusion

At the end of the whole analysis, the result is a $N_c \times N_c$ symmetric matrix, where N_c is the number of cities involved in the comparison, collecting on every row all the similarity scores between the set of cities and the one represented by the row itself.
Depending on the purposes, some normalization could be performed in different steps of the metrics building phase and different aggregations could be performed on the data. In this pilot the data has been aggregated on the cities and the normalization has been performed on the two distances E_{d} and JS_{d} before the application of the L2 norm, making every row not comparable with others.
An interesting way to read the results, could be obtained by normalizing the

rows and expressing the similarity by the mean of percentages, where the 100% is the similarity of a city to itself, and the 0% represents the similarity from a city to its most different in the row.

	City similarity
Genoa	100%
Istanbul	95%
Corinth	94%
Amsterdam	94%
. . .	. . .
Rapallo	93%
Chiavari	92%
. . .	. . .
Antivari	0%

Figure 3.18: Most similar locations, due to the selected metric, to the city of Genoa.

Generally, the results obtained have been satisfactory, confirming similarities known between particular cities, adding value in this way to non trivial ones.

3.4 To sum up

In this chapter we presented two different tasks accomplished in a real company scenario. The two projects are related to the revenue management in a tourist framework, forecasting specific revenues deriving from items belonging to a cross-selling business models in the first one, and analyzing overall tourist attractiveness of locations in the second.

While the first could be totally completed resulting in a forecasting and simulation interactived dashboard, the second project has been interrupted, allowing us to accomplish it only in half; however, it still generated very interesting highlights.

Conclusions

The work presented in the Thesis, performed within a company framework, dealt with two different topics belonging to the same business branch, the *Destination Management*, that is the analysis and the implementation of actions aimed at better managing the company offer related to touristic experiences.

In particular, two projects have been followed along the path: the first one has been related to *Revenue Forecasting* with the goal of building a model to forecast the revenues produced by a specific corporate asset; the second project, linked to the *Destination Discovery*, aimed at the high-level analysis of tourism opportunities in different geographical areas of possible interest of the company itself.

While these problems were very different in the details and the implementation, both from a modelistic point of view and the data typology involved in the process, their high perspective goal can be seen as an attempt of making *forecasts* at different levels with the common purpose of increasing business performances.

For this reason we started the work by introducing the concept of *forecast* and a set of its possible meanings depending on the framework.
We then focused on statistical approaches in forecast problems and we presented a set of mathematical techniques that are usually implemented in order to deal with such problems, in particular related to the Machine Learning field; we saw how this set of techniques can be very large, spatiating from *Linear Regression* and *Tree Based* methods to the more modern, but much computationally heavier, *Artificial Neural Networks*, depending on the task and the available resources.
Then, an overview of the process of using some historical data to build and *train* a ML model has been provided, starting from the beginning, with the data extraction and preparation, to the choice of the best set of hyparparameters given a specific model, to the implementation of a validation phase for the whole process, namely the *cross validation*.

After that we approached the practical problems of forecasting variables in a business frameworks, defining the meaning of this process and introducing the different actors constituting the business framework.
Finally, we presented the problems approached in the details providing the implemented methodologies, as well as data and results, in accordance with the existing NDA between the company and the University.

To sum up, even though a lot of details and real data had to be omitted and hidden in the analysis, proposed and implemented process still remains valid. Evaluating the impact of the work, the results obtained facing the different tasks generated value and produced interesting hints for the company itself, validating the impact of a *data driven* methodology in a business framework.

For a future perspective, from a business point of view, the first model built as a revenue forecaster could surely continue to be used in the near time horizon without any major change needed; only some minor updates and little changes will be needed in order to continuously shape the model around the dynamic framework it was built for.
The second project can be seen instead as a very interesting pilot, which showed the potential of the implemented methodology in attempting to make *quantifiable*, and then usable within different comparations, a complex variable like the *Tourism attractiveness* of a location, justifying then any future work in that direction which tries to exploit such a characterization.

Appendix A

In this Appendix we provide formal definitions of few mathematical objects that we used throughout the Thesis.

Definition A.0.1. Given a set X, a **metric** or **distance** is a function

$$d : X \times X \to [0, \infty)$$

for which $\forall x, y, z \in X$ the following three conditions are satisfied:

I) $d(x, y) = 0 \Leftrightarrow x = y$

II) $d(x, y) = d(y, x)$ (*symmetry*)

III) $d(x, y) \leq d(x, z) + d(z, y)$ (*triangle inequality*)

From these it also follows the *non-negativity* condition:

IV) $d(x, y) \geq 0 \quad \forall x, y \in X$

Definition A.0.2. Being $X = \mathbb{R}^n$ and $P = (x_1^P, x_2^P, \ldots, x_n^P)$ and $Q = (x_1^Q, x_2^Q, \ldots, x_n^Q)$ two n dimensional point in it, it is called as **Euclidean metric** (or distance) the function defined as

$$D_{\mathrm{E}}(P, Q) = \sqrt{(x_1^P - x_1^Q)^2 + (x_2^P - x_2^Q)^2 + \cdots + (x_n^P - x_n^Q)^2}.$$

Definition A.0.3. Being $p(x)$ and $q(x)$ probability densities of continuous random variables, the **Kullback–Leibler divergence** D_{KL} (or **relative entropy**) is defined as

$$D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \left(\frac{p(x)}{q(x)} \right) dx, \quad \text{if } q(x) \neq 0 \ \forall x$$

where P and Q are the random variables relative to the distributions p and q respectively.

Equivalentely, given the probability space $\mathcal{X}$ and two discrete random variables P and Q on it defined, the **Kullback–Leibler divergence** is defined as

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right), \quad \text{if } Q(x) \neq 0 \ \forall x \in \mathcal{X}$$

The KL divergence is not a metric; in fact, even though it is always true that $D_{\mathrm{KL}}(P\|Q) \geq 0$ and

$$D_{\mathrm{KL}}(P\|Q) = 0 \Leftrightarrow P = Q \text{ a.e.} (Gibbs' inequality)$$

it's not symmetric and doesn't satisfy triangle inequality either.

Definition A.0.4. Given the KL-divergence definition in Def.A.0.3, the **Jensen-Shannon divergence** is defined as

$$D_{\mathrm{JS}}(P \parallel Q) = \frac{1}{2} D_{\mathrm{KL}}(P \parallel Q) + \frac{1}{2} D_{\mathrm{KL}}(Q \parallel P)$$

While it satisfies the properties I) and II) of Def.A.0.1, inheriting the I) from KL divergence and II) trivially, it is not a metric; on the other hand, It can be proven [47] that its square root is a metric satisfying also the triangle inequality.

Bibliography

[1] [Lorenzo Rosasco "Introductory Machine Learning Notes.", 2017.]

[2] [Trevor Hastie, Robert Tibshirani and Jerome Friedman "The Elements of Statistical Learning: Data Mining, Inference, and Prediction.", Springer, 2016.]

[3] [Ken Littlewood "Forecasting and control of passenger bookings.", Journal of Revenue and Pricing Management, Vol. 4, No. 2, pp. 111–123, 2005.]

[4] [Bellman R. E. "Adaptive control processes.", Princeton University Press, 1961.]

[5] [Leo Breiman, Jerome Friedman, Charles J. Stone, R.A. Olshen "Classification And Regression Trees.", Taylor & Francis, 1984.]

[6] [Gilles Louppe "Understanding Random Forests.", PhD dissertation, University of Liège, 2015.]

[7] [Tin Ka Ho "Random Decision Forests.", Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 pp. 278–282, 1995.]

[8] [Hossam Zaki"Forecasting for airline revenue management.", The Journal of Business Forecasting Methods & Systems; Flushing Vol. 19, Fasc. 1, 2000.]

[9] [Yoav Freund and Robert E. Schapire "A decision-theoretic generalization of on-line learning and an application to boosting.", Journal of Computer and System Sciences, 55(1):119–139, 1997.]

[10] [Md Zahangir Alom, Tarek M. Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C. Van Essen, Abdul A. S. Awwal and Vijayan K. Asari "A State-of-the-Art Survey on Deep Learning Theory and Architectures.", Electronics, 2019.]

[11] [Ian Goodfellow, Yoshua Bengio and Aaron Courville "Deep Learning.", MIT Press Ltd, 2016.]

[12] [Biljana Petrevska "Predicting tourism demand by A.R.I.M.A. models.", Economic Research-Ekonomska Istraživanja, 30:1, 939-950, 2017.]

[13] [Dilek Penpece and Orhan Emre Elma "Predicting Sales Revenue by Using Artificial Neural Network in Grocery Retailing Industry: A Case Study in Turkey.", International Journal of Trade, Economics and Finance, Vol. 5, No. 5, 2014.]

[14] [Jie Mei, Dawei He, Ronald Harley and Thomas Habetler "A Random Forest Method for Real-Time Price Forecasting in New York Electricity Market.", IEEE Power & Energy Society General Meeting, 2014.]

[15] [JWan-I Lee, Cheng-Wu Chen, Kung-Hsing Chen, Tsung-Hao Chen and Chia-Chi Liu "A comparative study on the forecast of fresh food sales using logistic regression, moving average and bpnn methods.", Journal of Marine Science and Technology, Vol. 20, No. 2, pp. 142-152, 2012.]

[16] [Atesh Koul, Cristina Becchio and Andrea Cavallo "Cross-Validation Approaches for Replicability in Psychology.",Frontiers in Psychology, 2018.]

[17] [Miriam Seoane Santos, Jastin Pompeu Soares, Pedro Henrigues Abreu, Helder Araujo and Joao Santos "Cross-Validation for Imbalanced Datasets: Avoiding Overoptimistic and Overfitting Approaches.", IEEE Computational Intelligence Magazine, Volume: 13 , Issue: 4, 2018.]

[18] [Guozhu Dong and Huan Liu "Feature Engineering for Machine Learning and Data Analytics.", Taylor & Francis Group, LLC, 2018.]

[19] [Damir Krešić and Darko Prebezac " Index of Destination Attractiveness as a Tool for Destination Attractiveness Assessment.", ORIGINAL SCIENTIFIC PAPER, Vol. 59 No 4/ 2011/ 497-517, 2011.]

[20] [João Paulo Teixeira and Paula Odete Fernandes"Tourism Time Series Forecast - Different ANN Architectures with Time Index Input.", Procedia Technology, 5 , 445 – 454, 2012.]

[21] [Sebastian Vengesayi, Felix T. Mavondo and Yvette Reisinger" Tourism destination attractiveness: attractions, facilities, and people as predictors.", Tourism Analysis, Vol. 14, pp. 621–636, 2009.]

[22] [Alice Zheng and Amanda Casari "Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists.", O'Reilly Media, 2018.]

[23] [John Trevor Coshall "A Selection Strategy for Modelling UK Tourism Flows by Air to European Destinations.", Tourism Economics 11(2):141-158, 2005.]

[24] [Stephen F. Witt, Haiyan Song and Stephen Wanhill "Forecasting Tourism-Generated Employment: The Case of Denmark.", Tourism Economics, 10, 167–176, 2004.]

[25] [Egon Smeral "Long-Term Forecasts for International Tourism.", Tourism Economics 10(2):145-166, 2004.]

[26] [Jeff Heaton "An Empirical Analysis of Feature Engineering for Predictive Modeling.", arXiv:1701.07852v2 [cs.LG], 2020.]

[27] [Sebastian Raschka "Python Machine Learning.", Packt Publishing Ltd., 2015.]

[28] [Robert Layton"Learning Data Mining with Python.", Packt Publishing Ltd., 2015.]

[29] [Ken Chatfield, Victor Lempitsky, Andrea Vedaldi and Andrew Zisserman "The devil is in the details: an evaluation of recent feature encoding methods.", Proceedings of the British Machine Vision Conference, pages 76.1-76.12. BMVA Press, 2011.]

[30] [Daniel Drubach "The Brain Explained.", New Jersey: Prentice-Hall, 2000.]

[31] [Kedar Potdar, Taher Pardawala and Chinmay Pai "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers.", International Journal of Computer Applications 175(4):7-9, 2017.]

[32] [Isabelle Guyon and Andrè Elisseeff "An Introduction to Variable and Feature Selection.", Journal of Machine Learning Research 3 (2003) 1157-1182, 2003.]

[33] [Haykin S. " Neural networks and learning machines.", Prentice Hall. New Jersey, 2009.]

[34] [Daphne Koller and Mehran Sahami "Toward Optimal Feature Selection.", Computer Science Department Stanford University, 2000.]

[35] [Sebastian Ruder "An overview of gradient descent optimization algorithms.", arXiv:1609.04747v2 [cs.LG], 2017.]

[36] [Arbib, M. A. "Brains, machines, and mathematics.", New York: Springer-Verlag, 1987.]

[37] [Mark A. Hall "Correlation-based Feature Selection for Machine Learning.", Department of Computer Science, University of Waikato Hamilton, New Zealand, 1999.]

[38] [Mark A. Hall "Correlation-based Feature Selection for Discrete and Numerical Class Machine Learning.", Department of Computer Science, University of Waikato Hamilton, New Zealand, 2000.]

[39] ["Statistical Language - What are Data?", Australian Bureau of Statistics. 2013-07-13. Archived from the original on 2019-04-19. Retrieved 2020-03-09.]

[40] [Arbib, M. A. "The handbook of brain theory and neural networks.", Cambridge, Massachusetts, London, England: MIT press, 2003.]

[41] ["Body, Physics of.", Macmillan Encyclopedia of Physics. New York: Macmillan, 1996.]

[42] [Florian Pargent "A Benchmark Experiment on How to Encode Categorical Features in Predictive Modeling.", Master Thesis in Statistics Ludwig-Maximilians-Universität München, 2019.]

[43] [Paul J. Werbos "Backpropagation Through Time: What It Does and How to Do It.", PROCEEDINGS OF THE IEEE, VOL. 78, NO. IO, 1990.]

[44] [Doris Gomezelj Omerzel and Tanja Mihalic "Destination Competitiveness—Applying Different Models, the Case of Slovenia.", Tourism Management, 294-307, 2008.]

[45] [Anna Leask"Progress in visitor attraction research: Towards more effective management.", . Tourism Management, 155-166, 2010.]

[46] [Rosenblatt, F. "The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain.", Psychological Review. 65 (6): 386–408, 1958.]

[47] [Dominik M. Endres and Johannes E. Schinde "A New Metric for Probability Distributions", IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 49, NO. 7, 2003.]

[48] [Rosenblatt, F. " Principles of Neurodynamics.", Washington, DC:Spartan Books., 1962.]

[49] [Matti Hämäläinen, Riitta Hari, Risto J. Ilmoniemi, Jukka Knuutila and Olli V. Lounasmaa. "Magnetoencephalography - theory, instrumentation, and applications to noninvasive studies of the working human brain.". Low Temperature Laboratory, Helsinki University of Technology, 01250 Espoo, Finland.]

[50] [Suzana Herculano-Houzel "The Human Brain in Numbers: A Linearly Scaled-up Primate Brain.", Hum Neurosci. 2009; 3: 31, 2009.]

[51] [Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio "Generative Adversarial Networks.", Proceedings of the International Conference on Neural Information Processing Systems. pp. 2672–2680, 2014.]

[52] [Jiawei Zhang "Basic Neural Units of the Brain: Neurons, Synapses and Action Potential.", IFM Lab Tutorial Series, 2019.]

[53] [Akarsh Agarwal and Akash Motwani "Deep Learning Algorithms.", Computer Engineering Department, Institute of Technology, Nirma University, 2016.]

[54] [Warren S. McCulloch and Walter Pitts "A logical calculus of the ideas immanent in nervous activity.", The bulletin of mathematical biophysics volume 5, pages115–133, 1943.]

[55] [Alex Sherstinsky "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network.", Elsevier journal “Physica D: Nonlinear Phenomena”, Volume 404, 2020.]

[56] [Sepp Hochreiter "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions.", International Journal of Uncertainty Fuzziness and Knowledge-Based Systems 6(2):107-116, 1998.]

[57] [Stelios Timotheou "A novel weight initialization method for the random neural network.", Neurocomputing, Volume 73, Issues 1–3, Pages 160-168, 2009.]

[58] [Jim Y.F.Yam and Tommy W.S.Chow "A weight initialization method for improving training speed in feedforward neural network.", Neurocomputing, Volume 30, Issues 1–4, Pages 219-232, 2000.]

[59] [Clive William John Granger"Forecasting in Business and Economics.", Academic Press Inc, 1980.]

[60] [Pavlo M. Radiuk "Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets.", SSN 2255-9094 (online), ISSN 2255-9086 (print), vol. 20, pp. 20–24, doi: 10.1515/itms-2017-0003, 2017.]

[61] [Dominic Masters and Carlo Luschi "Revisiting Small Batch Training for Deep Neural Networks.", arXiv:1804.07612v1 [cs.LG], 2018.]

[62] [David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams "Learning representation by back-propagation errrors.", Nature, Vol. 323, 1986.]

[63] [S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.", In S. C. Kremer and J. F. Kolen, editors, A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press, 2001.]

[64] [Bent Fuglede and Flemming Topsøe "Jensen-Shannon divergence and Hilbert space embedding.", Proceedings of the International Symposium on Information Theory, 2004.]

[65] [James Bergstra, Yoshua Bengio and Balázs Kégl "Algorithms for Hyper-Parameter Optimization.", hal-00642998, version 1, 11.]

[66] [Xu Chu, Ihab F. Ilyas, Sanjay Krishnan and Jiannan Wang "Data Cleaning: Overview and Emerging Challenges.", SIGMOD '16: Proceedings of the 2016 International Conference on Management of Data, Pages 2201–2206, 2016.]

[67] [Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei and Si-Hao Deng "Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization.", Journal of Electronic Science and Technology, Vol. 17, No. 1, 2019.]

[68] [Cristóbal Casanueva, Ángeles Gallego and María-Rosa García-Sánchez "Social network analysis in tourism.", Current Issues in Tourism, 19:12, 1190-1209, DOI:10.1080/13683500.2014.990422, 2016.]

[69] [Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun and Rob Fergus "Regularization of Neural Networks using DropConnect.", Proceedings of the 30th International Conference on Machine Learning, PMLR 28(3):1058-1066, 2013.]

[70] [Kalyan T. Talluri and Garrett J. van Ryzin "The Theory and Practice of Revenue Management.", Springer Science+Business Media Inc, 2004.]

[71] [Mairead L. Bermingham, Ricardo Pong-Wong, Athina Spiliopoulou, Caroline Hayward, Igor Rudan, Harry Campbell, Alan F. Wright, James F. Wilson, Felix Agakov, Pau Navarro and Chris S. Haley "Application of high-dimensional feature selection: evaluation for genomic prediction in man.", Sci. Rep. 5: 10312. Bibcode:2015NatSR...510312B. doi:10.1038/srep10312. PMC 4437376. PMID 25988841, 2015.]