ALMA MATER STUDIORUM
UNIVERSITY OF BOLOGNA

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS
in
Artificial Intelligence in Industry

A procedure for modular forecasting at scale with constraints for business time series

Supervisor
Prof. Michele Lombardi

Candidate
Daniele Domenichelli

Academic Year 2020-2021

to my father

Abstract

Nowadays artificial intelligence algorithms are capable to achieve impressive results with a reduced amount of physical and time resources. They cover many different topics with a discrete success, but one of the most challenging subject to model is the prediction of future trends in complex and mutable environments, such as market sales.

From a deterministic point of view, the knowledge of the exact state and the rules of a system in a certain period intrinsically brings the faculty to forecast any future state. This perspective yields an exact prediction, but it lays its foundation on the assumption that its possible to model every aspect of the system, a premise that is usually satisfied only in simple cases.

The difficulty of predicting time-series is amenable to many factors, one of the most important is the drastically instable and mutable domain subjected to the competitiveness of its constituents, vendors and buyers, as described by the principles of the game theory; in such system, the rules are constantly changing, hardening the predictions of future states.

Furthermore, financial studies produced a variety of economic models which help to understand the market behaviour. By applying specific constraints to the prediction, it is possible to exploit these models to reach better and more explainable results and relate their components to the relative sources.

The aim of this work is to propose a procedure capable of inject domain constraints in the prediction in a declarative fashion, addressing different economic models. This procedure helps the analysts to better express their domain expertise while keeping a completely explainable approach to describe their outcomes.

Acknowledgements

I'm extremely grateful to the professor Michele Lombardi who were exceptionally available to kindly drive and suggest the steps of my work.

My gratitude goes to Alessio Bonfietti, the responsible of my internship at MindIt, who gives me the opportunity to live this experience.

This project would not have been possible without the precious help and sustain of Matteo Di Pisello, the company tutor who introduced me to the MindIt company and who were always by my side. He never let slip my call for help and he had always actively encouraged, advised and guided me through the internship and the development of this work.

I am very grateful to the academic professors, teachers and tutors, they invested their time and resources to contribute in many aspects to my formation, as a student, as a person.

I would like to express my deepest thanks to my mother, my brothers, my uncles and every member of my family. They sustained my studies, both economically and emotionally, encouraging me to express my best self. Without their effort I would not be gone that far off.

A debt of gratitude is owed to all my colleagues who have collaborated with me. It was a pleasure for me to exchange our perspective and grow together.

A special thanks to my dear friends who have supported me in any moment of my life, the bright as well as the dark ones.

Contents Index

Abstract **ii**

Acknowledgements **iii**

Contents Index **iv**

Image Index **vi**

Table Index **vii**

Introduction **1**
Business time series 1
Market as Complex System 2
Problem definition 3
Approaches 4

1 Related Works **5**
1.1 Recurrent Neural Networks in time series 5
1.2 Forecasting at scale: Facebook Prophet 5
1.3 Painting the future 6
1.4 PSO in time series forecasting 6

2 Data and Feature Extraction **7**
2.1 Harvesting 7
2.2 Validation 8
2.3 Pre-processing 9
2.4 Seasonality extraction 10
2.5 Dataset modularity 13
2.6 Intra-product validation 13

3 Methods and Models **15**
3.1 Methods 15
3.1.1 Polynomial regression model 15
3.1.2 Mean model 16
3.2 Neural Models 18
3.2.1 Multi-layer perceptron 18
3.2.2 Recurrent Neural Network 19
3.3 Prophet 21
3.3.1 Structure 22
3.3.2 Stochastic regression 24
3.3.3 Declarative model 25
3.3.4 Non-positive constraint 25
3.3.5 Piece-wise linear regression 26
3.3.6 Modularity 30

4 Experiments and Results **31**
4.1 Synthetic test 31
4.2 Price regression test 32
4.3 Real-case scenario 32

5 Results **34**
5.1 Synthetic test 34
5.2 Price regression test 35
5.3 Real-case scenario 38

6 Discussion and Remarks **39**
6.1 Limitations 40

7 Conclusions **41**
7.1 Future works 41

Bibliography **A**

Image References **D**

Image Index

Fig 2.1	Example of an auto-correlation (normalized) plot	11
Fig 2.2	Example of a fast Fourier transform (FFT) plot	12
Fig 3.3.1	Prophet forecasting loop	21
Fig 3.3.2	Prophet components summary	22
Fig 3.3.3	Saturating market law, “supply and demand”, Jenkin 1870	27
Fig 3.3.4	Regression over a cloud, comparison	28
Fig 3.3.5	Gaussian distribution	29
Fig 3.3.6	Laplace distribution	29
Fig 3.3.7	Cauchy distribution	30
Fig 5.2.1	Prophet, example of a price regression, base model	35
Fig 5.2.2	Prophet, example of a constrained price regression, proposed model	35
Fig 5.2.3	Prophet, example of a price PWL regression, proposed model	35
Fig 5.2.4	Comparison of base (Stan v1), non-positive (Stan v2), constrained and unconstrained PWL (Stan v3) Prophet models	36

Table Index

Table 2.1	Dataset Description	10
Table 5.1.1	Prophet synthetic test results	34
Table 5.2.1	Prophet models performance comparison	37
Table 5.3.1	Real case model MAE comparisons	38
Table 5.3.2	Real case model R^2 comparisons	38

Introduction

Forecasting business time series is nowadays still a hard task and requires analysts specialized in this field. Many researchers recently invested their efforts to study the best procedures which produce high quality forecasts, but the practice of prediction is usually intrinsically hard to be explained and, even in the case of a simple domain, it particularly needs specialized analysts to make intelligible the results. Furthermore, modern data-oriented approaches are often treated as black box models which are hard to tune as well as difficult to properly be explained and usually inflexible to human observation and domain expertise.

The proposed procedure brings the two realities of interpretability and domain knowledge injection within the reach of specialists as well as people who may possess domain knowledge but unexperienced in time series methods or without any forecast modelling proficiency.

Business time series

Market sales are in constant movement because they are affected by an immeasurable number of agents, becoming sometimes very hard to understand even for an expert. Indeed, trading is one of the activities which most touches the human social behaviour, and its complexity keeps growing as more studies are experienced in this field, leading to an expansion of the rules that regulates the market.

In control theory, to precisely predict a future state (and its relative output) of a system[1] it is needed that this one satisfies the controllability principle [1], which states that it is possible to modify the system internal state by interacting with one or more inputs. In open world and very complex environments, it is practically impossible to deterministically model inputs and state dimension beforehand knowing the market developments: introducing more agents (vendors or buyers) lead to a change in terms of internal rules and states of the modelled system. So, changing rules means

[1] Controllability is usually described in linear systems, but it is proven that can be extended to non-linear systems [1]

changing models and so the predictability of a deterministic result become impossible.

When it could not be possible to deterministically model in detail a system, because of the many variables to take in account but also for being able to observe the inputs and the outputs of the system, it is always possible to stochastically model the same system by assuming that the model is responsible to introduce an uncertainness in the process. This duality in the way of seeing the same system is well visible in the famous "coin toss" problem: we can describe the physical laws underneath the toss of the coin, needing the initial state, and predict exactly the output [2], or we can describe the randomness introduced by the unknown initial state in a stochastic fashion.

Considering the variability of the environment, business time series are usually analysed, treated and forecasted by leveraging of uncertainty of unknown or non-modellable factors with the use of stochastic models. Mahmudov proved that is possible to describe a stochastic systems in terms of controllability [3], which means that in principle it is possible to predict, with a certain degree of confidence, the output of a liquid system such as the market.

Market as Complex System

Even though we could treat the uncertainty with stochasticity, we can't exclude that the market is a reactive system composed by agents. Business analyses are always performed in order to make an action in the sales environment. Each action is responsible to a slight change of the system behaviour, given the fact that the agents can adapt and modify their conduct with the respect to that action or to the system change.

The market can be briefly described as a collection of two kinds of entities: vendors and sellers. From the 19th century many studies were conducted in the attempt to model the relations between the agents. One of the most famous result of these studies, regarding in particular the relation in between vendor and seller, is the economic model of "supply and demand" described by Cournot in 1838 [4]. Cournot described a linear proportion between the price of the product and the sold quantity, which was then expanded by Jenkin in 1870 [5] with a diminishing return economic model. Other studies, instead, better describes the relation between different

vendors. These are centred on the game theory, well represented by the famous "Two prisoner's dilemma" exposed by Poundstone [6]. Two agents indeed maximize their expectations by assuming a competitive attitude, especially if they cannot rely on each other. Inspired by the mean-field physical theory, Caines has widen and explained what can happen in the case the agents are a minimal part of the system, constituting a sub-environment by their own [7].

Other researchers tried to view the market as an undirected graph, trying to map the relations and transactions between vendor and buyer nodes to achieve a predictable model for the field.

One last important source of discrepancy between model and reality to keep in consideration is the human bias, reflected in its models, to learn from past experiences, which can result into ineffective predictions of the future in systems which constantly and repeatedly change, even more if the experience is part of the system change itself.

Many attempts to understand and regulate the market were made, but the attempt itself is representable as an action of an agent which produces effects in the system, perturbating the behaviour of the other agents. Still the prediction of business time series through economical and sociological models struggle to completely describe the complexity of such a sophisticated and dynamic environment. These reasons are sufficient to restrict the analytics to a narrower perspective and draw strict assumptions (such as the Cournot's supply and demand relation) in order to simplify the modelling process.

Problem definition

Many modern methods offer the possibility to deal with stochasticity during the modelling of business time series, but almost all of them lack a procedure capable to inject domain constraints (discussed in the previous sections) with ease. Usually, the insertion of such constraints is demanded to experts who often have no expertise in the market sales domain, creating a divergence between the implementation and the usage of the model.

This observation inspired this work to focus on the creation of a procedure capable to respond to different needs: the prediction of a business time series; the large scale, as amount of data, of the prediction; the easy usability of the model even for non-experts who may have domain knowledge about the process; the modularity of the method to be employed multiple

times in the same problem but also the capacity of generalization to reach the largest variety of forecasting problems (not only related to market sales domain); the possibility to inject constraints with ease just by exploiting the owned domain knowledge.

Approaches

The development of the procedure was bound to a real case scenario, thanks to the work proposal by the MindIt company. The constraint injection property of the model was initially aimed to prevent unexplainable and unoptimizable behaviours in the company pipeline, but then it was expanded to include a more generic approach, imitating the same generalization approach used by previous models.

The work revolved around the analysis of the data available for a supermarket chain. For a retailer it is useful to make a distinction between two different period of the sale: sell-in, when the retailer buys the product from the manufacturer, and sell-out, when it sells to the single customer. This distinction is also present in the way the data is collected. Indeed, the sell-in data is almost always available because it is the retailer itself that manage the transaction with the manufacturer, furthermore it is also possible to check the correctness of the data because the source is very reliable. On the other hand, the sell-out data is harvested by crowd-mining methods, which sometimes could be unreliable in terms of presence and trustiness of each single sample, or by assuming companies responsible to collect and process the data coming from each single store cash desk. This second method is way more reliable, but it comes with an economic cost that could not be worth enough to spend.

The taken approaches firstly aimed to ensure the domain constraints in the relationship between price and the number of units sold, subsequently expanding the procedure to involve the prediction of a projected comprehensive units sold baseline for the sell-out. The baseline would be affected by many components (e. g. leafleting or discounts), so we can simulate their removal such that we obtain a cleaned baseline, which can be used in subsequent analysis to understand the impact of certain decisions in the sales market, such as the appliance of a discount in a certain period of the year, like Christmas.

1 Related Works

1.1 Recurrent Neural Networks in time series

Hybrid approaches can be employed in the time series forecasting. We can inherit the robustness and explainability of the old fashion models, such as the autoregressive moving average model (ARMA), but exploiting the flexibility of neural models. In particular, the use of a recurrent neural network (RNN) in an extended neural ARMA model (NARMA), was employed to filter the outliers and improve the training of the hybrid model [8]. Other approaches benefit from the use of forward and feedback paths while using a bidirectional RNN directly on the data to make order and sense in the chaos of a time series [9], achieving better results than old fashion models.

This work employs an unsophisticated recurrent neural network model to have a basic comparation between deterministic neural approaches and stochastic methods.

1.2 Forecasting at scale: Facebook Prophet

Meta (Facebook at the time of the paper) employees invested much effort in the time series forecasting related to the prediction of periods with overcrowding events [10]. The big company had the necessity to develop a web environment capable to anticipate event-congested periods to improve the service for the final user and keep the results as explainable as possible. The limitation of resources and the outcome clarity bring to the development of a robust infrastructure capable of predicting time series also in fields unrelated to the application events. Indeed, different articles mention Prophet, the model invented by Taylor and Letham at Meta [10], as capable of predicting business time series. Furthermore, some researchers recently conducted some studies in this sector, showing that it is possible to build frameworks for real-world data belonging to sales domain employing Prophet [11].

1.3 Painting the future

Given the success of the deep convolutional networks and the fast spread and growth in the last years, the newer trend is to find a way to depict data as a coherent image and try to make a completely new sense from the rearranged features. For time series, the pioneers in this technique found successful to convert the data into images that intrinsically contain the time axis embedded in each pixel position. This encoding is produced by two different transformations employing transformations such as Gramian Angular Field (GAF) [12, 13, 14] and Markov Transition Field (MTF) [15].

The concept underneath these encodings is to convert the time proximity of a sample into a spatial proximity in an image. Then the usual procedures used in the image processing pipeline are employed to predict the expected result (e.g. classification, regression, etc.).

By leveraging the experience already accumulated over years of research in convolutional networks, this approach seems to mine good results, but this promising approach is still in its infancy.

1.4 PSO in time series forecasting

All the methods described so far do not take in account or exploit the non-stationary property of the market sales. The financial system is so complex that even if we perform the same action in two different time periods, the outcomes could be drastically diverse.

Recent studies discovered that neural models, which employ Particle Swarm Optimization (PSO) as learning techniques [16, 17], are able to extract patterns from non-stationary data, improving the performances of the model. Even if these researches are still experimental, because they just partially probed the reasons behind their success, their promising outcomes are attracting interest in the PSO employment for time series forecasting, and their growing number through the years is proving of this.

2 Data and Feature Extraction

The dataset for the real case scenario was provided by the MindIt company and it was collected over a period of four years. The data contains sensible information, so every reference to it will be concealed or transformed for privacy reasons, while keeping the understandability of the discussion.

As already mentioned, the dataset is collected in two different times and pre-processed to guarantee a unique relation between the two periods. The curves that were collected for the sell-out were also projected to be compatible with the curves at sell-in and merged by matching the dates.

Below, we will summarize the process that led the data from the collection to the usage in the procedure.

2.1 Harvesting

The collection of the data was divided in two different periods: sell-in and sell-out. The first is easy to harvest, since each retailer has an internal balance sheet account that enumerates the invoices relative to each purchase from a specific manufacturer.

In the case of the sell-out instead, the retailer has no information about the single transaction of each owned store, so it needs to directly collect the data from them. This usually can be done with crowd-mining techniques or by assuming companies which collect and process data from the stores.

A generic approach to crowd-mining is to drive customers to record their receipts through incentives, like discounts and coupons, or by using loyalty cards. One of the advantages of crowd-mining techniques is the thin capillarity of the collection, since each person can describe its tiny situation, but on the other hand the reliability of each one of them is brought up.

For our final dataset the retailers decided to have a reliable source by investing in other companies that harvested the data. This meant that crowd-mining is not a viable option, so they have chosen to entrust the data harvesting to a third company, which grants for the dataset quality. By involving other companies, the retailer has to spend more money in the process, and sometimes could not be interested or it could have other reasons not to buy the data harvesting for a certain period, product or seller. Filling

the gaps introduced by the absence of sell-out data is part of the objective of this work.

2.2 Validation

The data can be a source of error, especially in the case it is harvested through unreliable processes. In this work two different datasets were used in different experiments.

For what concern the price regression experiment (section **4.2**), the used dataset was in part collected through a crowd-mining technique, requiring a more in-depth analysis and validation before proceeding with the usual process.

In general, due to a possible partial unreliability of the harvesting method, or more broadly to a possible data inconsistency during any previous step, it is needed to perform a validation to ensure the coherence of the process. The crowd-mined collected material indeed was intrigued by insertion errors and corrections (e.g. invalid dates, negative or invalid number of units sold or paid amount, etc.) but also by behaviour anomalies in the data recording (e.g. users that accumulate receipts for weeks or months and record them in a single day).

Another important aspect that was taken in account was the effect of the lockdown due to the pandemic 2019 COVID. In this period indeed the data, where present and collected, drastically changed because of the forced modification in the shopping behaviour of the people. For this reason, the data during this period was briefly analysed to understand the causes and the effects derived by this period and subsequently was discarded by the process pipeline to assess more consistently the whole procedure.

In the real-case scenario (section **4.3**) was employed a dataset collected by a third company, which processed and validated the data in advance, relieving the needing of such operation.

2.3 Pre-processing

After the validation, the data needs to be pre-processed to obtain coherent results. The first step is to aggregate the sell-in transaction dataset into coarser grained sets, which can contain daily, weekly or monthly information. The daily dataset is responsible for the forecasting, while the others are useful for a broader inspective analysis (i.e. finding anomalies in the insertion behaviour for the harvesting phase). Subsequently the sell-out dataset, which has already a daily granularity, is merged into the aggregated sell-in dataset, obtaining a unique set.

The outcoming set is primarily composed by the following columns:

Column	Description
Date	Time of transaction (with daily granularity)
Product	Code representing a product
Retailer	Code representing a retailer
Store	Code representing a store owned by a retailer
Units sold SO	Units sold in sell-out (retailer to store)
Units sold SO PROMO	Units sold in sell-out (retailer to store) during a retailer promotional event
Amount SO	Total income in sell-out (retailer to store)
Units sold SI	Quantity of units sold in sell-in (store to shopper)
Units sold SI PROMO	Units sold in sell-in (store to shopper) during a retailer promotional event
Amount SI	Total income in sell-in (store to shopper)
Brand	Informative code of the product brand

Pair	Tuple describing Product and Store
Price SO	Amount SO / Units sold SO
Units SO NP (No Promo)	Units sold SO – Units sold SO PROMO
Price SI	Amount SI / Units sold SI
Units SI NP (No Promo)	Units sold SI – Units sold SI PROMO

Table 2.1 *– Dataset Description*

We can compute the price columns (*SI* and *SO*) by dividing the amount by the relative units sold column. The promotional event columns will be later useful to understand the impact of a retailer investment during different periods of the year.

Every prediction regards a single product and a single store, so we can namely index the dataset by the product code and the store code, which we will call pair.

2.4 Seasonality extraction

A peculiar part of the analysis and pre-processing of a business time series is the possibility to observe a periodicity in the behaviour of the vendors selling attitude. This periodicity can have a different time scale. For example, there are seasonal products which are sold just in a part of the year, indeed we expect ice-creams to sell more in the summertime while hot chocolate to

be more purchased during the wintertime; recreative or party related products, such as beverages and finger food, have a selling increasing just before or at the start of the weekend.

The seasonality of a product is the description of the recurrence of a selling curve (as a one-dimensional signal) and is composed by two factors: cadence (or phase), which is the time interval between the peaks of the curve, and strength (or amplitude), which describe the general intensity of the recurrence event. The position of each peak represents the time of the selling concentration, while its height represents how likely is to sell the product in that time. For example, seasonal (ice-creams) and event-bound (Christmas decorations) products both share an annual cadence, given the fact that they will be always sold in certain months of the year, but the latter have a huger strength then the first. This is because the period of the events is usually fixed and has a shorter duration than a season, concentrating the selling in well-defined periods.

The usual way to discover the most qualitatively intense seasonality cadences is to use the autocorrelation method. It computes, for increasing lags, the auto-covariance of the signal, or the covariance between a signal and its time-translated version. The single auto-covariance for the lag L describes how much the curve match with the same curve but translated for a L period.

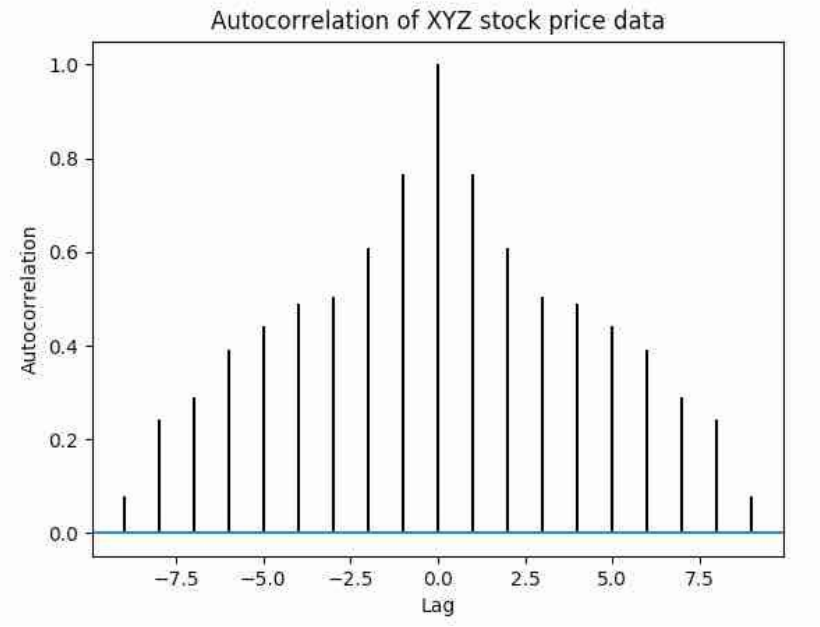

Fig. 2.1 *– Example of an auto-correlation (normalized) plot*

Once the scale of the most promising periods is extracted, we can proceed with a selective fast Fourier transform (FFT) to decompose the original signal in a sinusoidal series, including the ranges extrapolated by the auto-correlation.

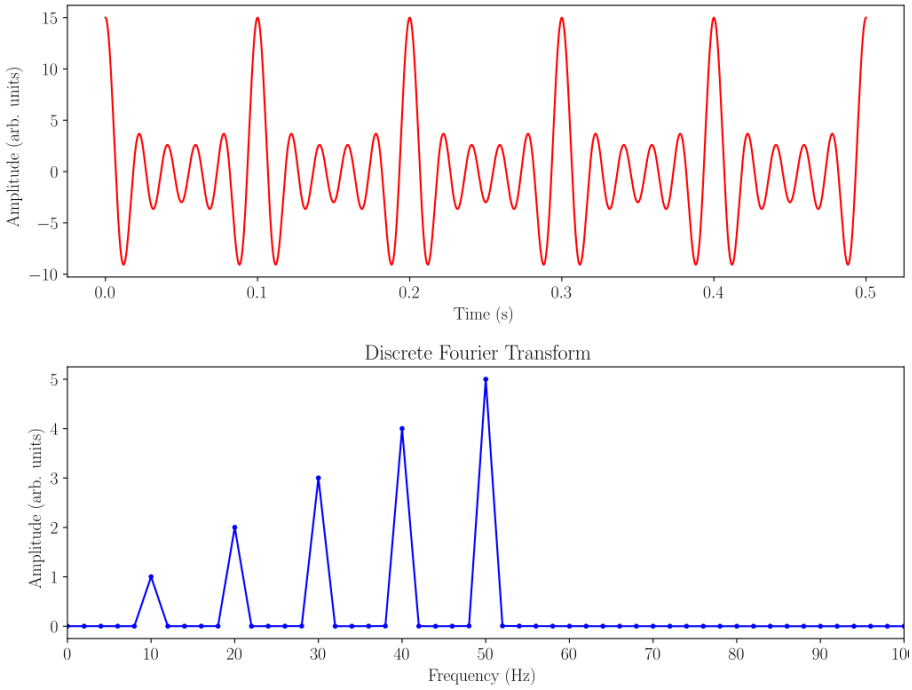

***Fig 2.2** – Example of a fast Fourier transform (FFT) plot*

The seasonality can be useful when employed in the procedure because it can well explain the measure of the impact in the units sold quantity due to the period of the year, leading to a better understanding of the best time to apply a discount. For a trivial example, we can understand that an ice-cream promotion is ineffective during the wintertime just because the population is not available to buy that product in that period. Obviously, other cases are not so trivial to be observed just by looking at data, even for expert analysts, so this procedure helps to bring out this feature but also allows to quantitatively compare the intensity between two distinct products.

2.5 Dataset modularity

The whole data came in different chunks, from different kind of sources and at different times. It was decided to proceed the analysis with a modular approach, starting by selecting a manageable portion of the dataset, which was restricted to a part of a chosen target product category, and then more and more quantities and categories were introduced to work with. This slowly insertion of increasing chunks of data allowed to better separate which of the effects were bounded to smaller parts and their categories and which instead were more broadly describing the whole dataset. The future selection of which features must be included in the process was also influenced by this procedure step.

2.6 Intra-product validation

The last aspect that the framework needs to take in account is the capacity to assess the predictions in a coherent way. The evaluation method must be equal for all the models but also for all their training conditions.

It is a good practice to analyse the data under different aspects, one that resulted important is the distinction of the behaviour observed between different products with respect to the conduct observed in the sellers which share the same product. We can describe the first perspective as inter-product, which compare the dissimilarities of different products, while the second is called intra-product, which instead focusses on the diversity found in the pairs that shares the same product, but their sellers differ one each other.

The curves shown a discordant variation in the units sold column, higher for inter-product and lower for intra-product. This was enough to create a model validation system which train and validate the model in a product-seller pair rotation.

This rotation enhances the distinction of different products, in this way we can create a model specialized in the prediction of a single product, improving the performances and highlighting product specific features, which are very valuable for an output backtracking and in subsequent analysis.

Coming to practice, the inter-product validation algorithm for each train step:

1. Selects a single product from the training set
2. Selects a single target seller from the pairs containing the selected product
3. Trains the model for all the pairs containing the product and not containing the target seller
4. Validates the model by computing the metrics on the target pair predictions

At the end of the training, the model parameters are stored for the subsequent test phase. This kind of validation better highlights the possibility of a model overfitting with a specific product, and it also allows to understand which inter-product features are better represented from different models.

3 Methods and Models

In this chapter we will discuss about the methods and the models used to assess the overall performances of the forecasting procedure. The final evaluation of each model and method was assessed with a regression metric between the target column, which is the reconstruction of the sell-in units sold curve, and the predicted column.

3.1 Methods

Before using modern approaches, the work started by considering the history of business time series forecasting. The first approaches studied came from the research of mathematician in the regression field, with a special focus over the market sales domain.

3.1.1 Polynomial regression model

This model employs a polynomial regression in order to predict the units sold at sell-in. This model was the first to be employed since it was already used as a control step in the reference company analysis pipeline. There are different grades and approaches to fit a polynomial over a dataset. The most widespread used approach is the least square method, which was attributed by both Legendre and Gauss at the beginning of the 19th century. Gergonne in 1815 then published a first application of this method [18], making the studies of the two mathematicians practicable. With the advent of the information technologies, more and more machine learning techniques involved this method of regression in order to train an algorithm to infer polynomial curves from a point cloud. The most recent and effective employment is the LS-SVM (Least Squares Support Vector Machine). It reformulates the usual SVM classifier in order to include the least squares method by reinterpreting the classification as a binary regression [19].

Parallelly with the machine learning development, in the second half of the 20th century, computer graphic also was a hot topic for the research. Another point of view in the polynomial description was found by Pierre Bézier, which pointed out a way to draw a polynomial curve from a set of control points. This method was borrowed from Paul de Casteljau, which invented an algorithm capable of describe a polynomial curve of grade n through a summatory of basis polynomial (a linear combination of different

polynomial of different grades). Years later, De Boor improved and generalized the Casteljau's algorithm revisiting the form of the basis polynomial, exploiting the Bézier finding, reforming it in the B-spline form. De Boor stated that a curve is decomposable into a piece-wise concatenation of curves that can be expressed by a single control point. Theoretically, this algorithm should be able to describe any polynomial (or any curve locally expressible through polynomials) of any grade, but practically its complexity narrows down the possibilities for higher grades. Modern programming libraries includes implementations of the mentioned algorithm in many different forms [20], not only for computer graphics fields.

From these two historical branches the choice fell to the Bézier regression. The reasons we opted for this alternative were the huge expressiveness and application speed for lower grades, the capacity of the algorithm to take in account the derivatives during the regression of the control points and also the intrinsic smoothness of the outcomes.

The polynomial regression model was bound to the piece-wise prediction of a Bézier curve chain. The advantages presented by this choice are the non-compliance to the intra-product validation needing and the possibility to make a prediction without any history or training. Indeed, this model was useful to have an advanced windowed smoothing of the sell-out data. This behaviour exploits the fact that sell-in sales are a diluted and noisy version of their sell-out counterparts, giving the possibility to lay down an understandable curve that maps the two domains to each other. This approach led to a better view over the possible problems in the subsequent models.

At the end, a simple seasonality extraction (FFT) was introduced to partially inject domain knowledge in the curve trend and give more context to the actual outcome.

3.1.2 Mean model

The mean model is the most basic and simple model taken in account. It is based on the assumption that each product has a very strong yearly seasonality, which means that every year the same store will sell almost the same number of units in the same periods. This strong assumption is very effective in non-occasional and seasonal products; indeed it was so efficient in the real-world dataset and also so simple and explainable that was taken as a baseline approach for this work. It simply computes the average of the past

years for each needed curve, rescaling it by the average of units sold. The rescaling is a very important step because it reduces the effect due to the seller size. Logically and just by giving a simple glad to the data, it is very visible that a huge metropolitan shopping centre will have a number of units sold way bigger than a small shop of a little town. The observation also shows that the selling behaviour, and the relative seasonality, of a certain product will be almost the same for both, just with a different magnitude. The rescaling allows to make comparable the sellers in the context of a product.

Since we are using the intra-product validation (as explained in section **2.6**) this will traduce in computing a rescaled average of the target curve over all the train pairs for a product and then multiplying the result by the test factor, which is the mean of the units sold for the target pair.

The model is described by the following formulas:

$$a_p(t) = \frac{x_p(t)}{w_p}, \;\; w_p = mean_t\big(U_p(t)\big) \qquad \textit{eq. 3.1.1}$$

$$A(t) = mean_p\big(a_p(t)\big) \qquad \textit{eq. 3.1.2}$$

$$y(t) = w_{val} \cdot A(t), \;\; w_{val} = mean_t(U_{val}(t)) \qquad \textit{eq. 3.1.3}$$

Where $U_p(t)$ is the curve of the units sold for the product p; $w_p(t)$ is the product weight factor; $x_p(t)$ is the input curve selected for the product p; $A(t)$ is the average curve, representing the parameters underneath this model; then w_{val} and y are respectively the weight factor and the prediction for the validation pair.

The generalization of the input curve allows to better explore the landscape of the interactions between different columns. The input curve should be the target column or a (usually linear) combination of columns (including the target one). In our experiments we measured the performances of a tiny set of combinations, the units sold in sell-out was the target column which was enriched by other components, like its seasonality or trend.

For training the model it is possible to use any combination of the columns brought from the dataset. It was firstly used the target column, but also other combinations were explored. Indeed, the MindIt proprietary algorithm extracts a baseline reconstructed curve composed primarily by a trend and a seasonality. The mean model was capable to explore both each single component of the reconstructed curve and a linear composition of them, finding out that this last option was the most promising.

For the simplicity of this approach and its understandability, it was taken as a baseline model to compare the efficiency of the other models.

3.2 Neural Models

Recently many studies focused the attention over the possibility to forecast time series with neural models, which are nowadays the most successful approaches to mimicking the human expertise for complex environments. Neural methods introduce the possibility to employ machine learning algorithms to tune a set of function parameters such that is possible to reproduce the outcomes of an unknown function. In particular the training phase of the machine is usually driven by gradient-based learning, making use of techniques based on gradient descent and backpropagation.

We will explore some of these modern approaches to compare the outcomes and discuss about the pros and cons with respect to the other methods.

3.2.1 Multi-layer Perceptron

The most widespread class of artificial neural networks (ANN) employed in many fields is the so called Multi-layer perceptron (MLP), more precisely a generic feed forward ANN.

The term perceptron is historically confusing in this case because it refers to a single artificial neuron with a threshold function (used in binary classification) [21] that was inspired by the natural neurons of the human brain. While the strict term of perceptron explicitly refers to this activation, it became popular when many algorithms employed a collection of perceptrons stacked in layers (single or multiple layer perceptron) [22]. The studies deepened through the years the concept of activation function of a neuron, but the term of layered perceptron became more popular for binary classificators that employ a group of neurons that shares the same activation. So, the term of perceptron passed from representing a single neuron to representing a complete collection of classificators. Nowadays a multi-layer perceptron can be intended in the literature as a collection of many layers of threshold-activated neurons but also as any feed forward artificial neural network, usually represented by a set of fully connected layers.

The research over the time series analysis and prediction through deep learning models is still active and debated. Gamboa summarized the latest

most explored techniques in its work [23] by showing that some works could be promising and mined some good results in the appliance of deep learning technique in this challenging field.

The oldest and simplest (but not simplistic) model employed in the time series analysis is the multi-layer perceptron (intended as a feed forward ANN). Since the latest 20th century, many researchers started to be interested in the application of this model in the business time series forecasting. Hoptroff shown in its work the practical feasibility of this task with the employment of traditional neural networks approaches [24], such as the MLP. Many papers of this subject were published until now, meaning that the more performant approaches are still discussed in the scientific community.

In our case, we decided to implement a fully connected model capable to analyse an entire year of data and simulate the relative sell-out curve. The yearly seasonality choice was selected because the human social behaviour is intrinsically imbued of such periodicity, and also the seasonality extraction (see section **2.4**) from the dataset confirmed this attitude.

The model is composed by a stack of fully connected layers with varying dimensions. As a good measure, the number of layers and their relative amount of neurons, are treated as model hyperparameters and were tuned by an extensive grid search. The search was executed over the full dataset for a reduced amount of iterations, and then the metrics were compared in order to declare the baseline model for this class. The most promising models were then trained for a longer period and their results were compared once again. After this competitive selection, the best model was stored for the comparison between the other approaches.

3.2.2 Recurrent Neural Network

With the success of deep learning in the natural language processing (NLP) field, the recurrent neural network (RNN) became more popular for its capacity to encapsulate temporal dynamic features in the network weights. This property comes in handy also when forecasting time series. We can see a RNN as a peculiar linear[2] time-invariant system which has an impulse response that does not ground to zero after a certain period (infinite impulse response property) [25]. This means that a portion of the input effects on the

[2] It can also be expanded as a non-linear system by using non-linear activations

system are retained in a sort of internal memory, condition the outputs at different time steps.

The innovative discovery about the system temporal dynamic behaviour allowed to create dedicated network that is responsible to works as independent memory cell (exploiting feedback loops), surrounding it with another part which is dedicated to a sort of management of the memory. This principle is used in the long short-term memory (LSTM) and gated recurrent units (GRU) networks. These are common and performant expansions for the RNN network, broadly used in NLP tasks.

Borrowing the experience from the language models, RNN were employed to assimilate, analyse and predict samples in the time domain, including the business time series forecasting field. Connor et al. work shown the promising efficiency of RNN in the cleaning of the data [8]. Later others were inspired and interested in the deepening of this approach, trying also to apply it to the market sales domain [9, 25].

The mode employed is a tiny stack of recurrent layers with a decreasing encoding, which makes use of lookback technique [26]. In practice the network takes in input more than just one sample but predicts the output relative to the last item of the input sequence. This gives more context to the network to work with and it better prevents overfitting, but on the other hand requires more data to train. Conceptually, it is easier to sees a period (such as a month or a year) and decide from that context what value should come next instead of looking at just one sample. The RNN memory is indeed influenced from a wider period, being able to generalize better the data behaviour over different time scales.

As for the MLP model, the depth and size of the network layers, as well as the lookback period, were considered as hyperparameters to tune, by using a grid search for a reduced amount of iterations. Given the unexpected poor results, we decided to just keep the best model instead of running another performance test over the most performant ones.

3.3 Prophet

The procedure, final outcome of this work, was inspired and based on the model developed by Meta (Facebook at time of the paper) employees, called Prophet [10].

Prophet is a stochastic forecasting framework employed for the prediction of event-congested periods in the Facebook application. The aim of this model is to define a framework capable to forecast, through regression methods, a time series just by declaring the domain knowledge. This is done by automatizing some procedures that the developers were used to follow during their data analysis tasks at Facebook. The principle is to involve the human analyst expertise in the forecasting system by building a solid framework which takes care of the knowledge needed in the time series domain.

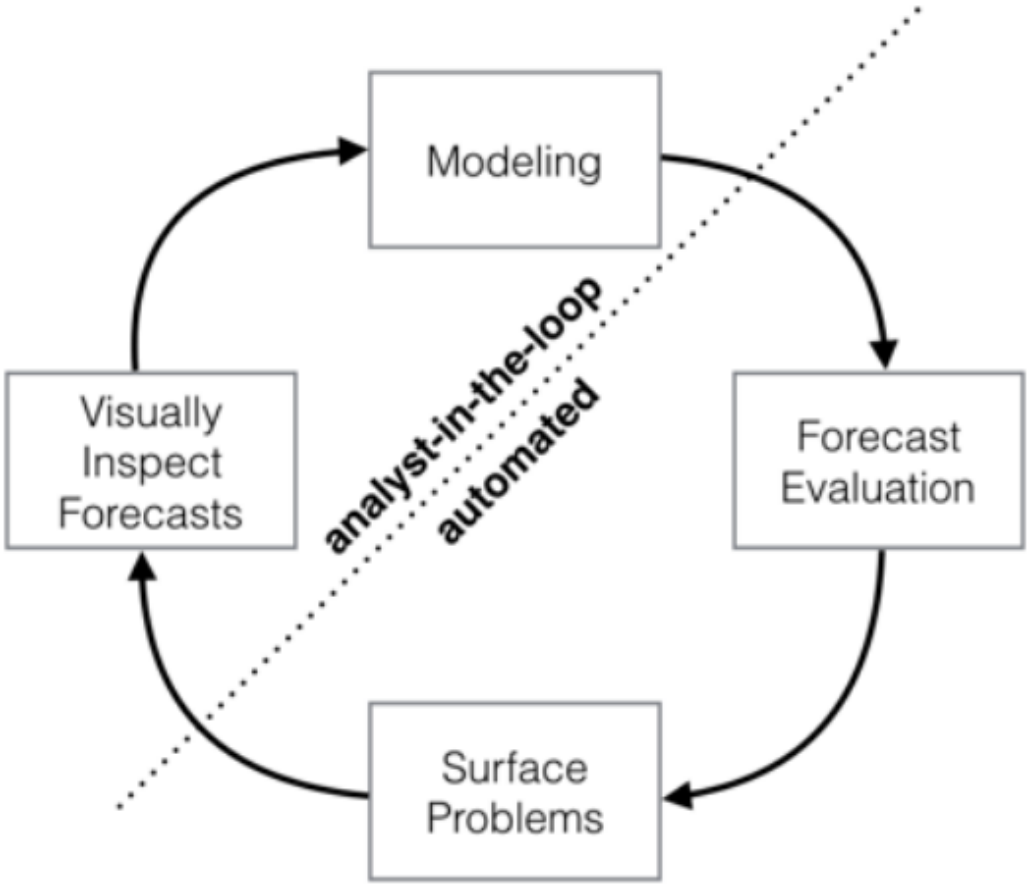

Fig 3.3.1 *– Prophet forecasting loop*

3.3.1 Structure

The Prophet model is mainly composed by two elements:

- **Pre-processor**, responsible to pack and prepare data for subsequent steps
- **Analyst**, a stochastic inference model

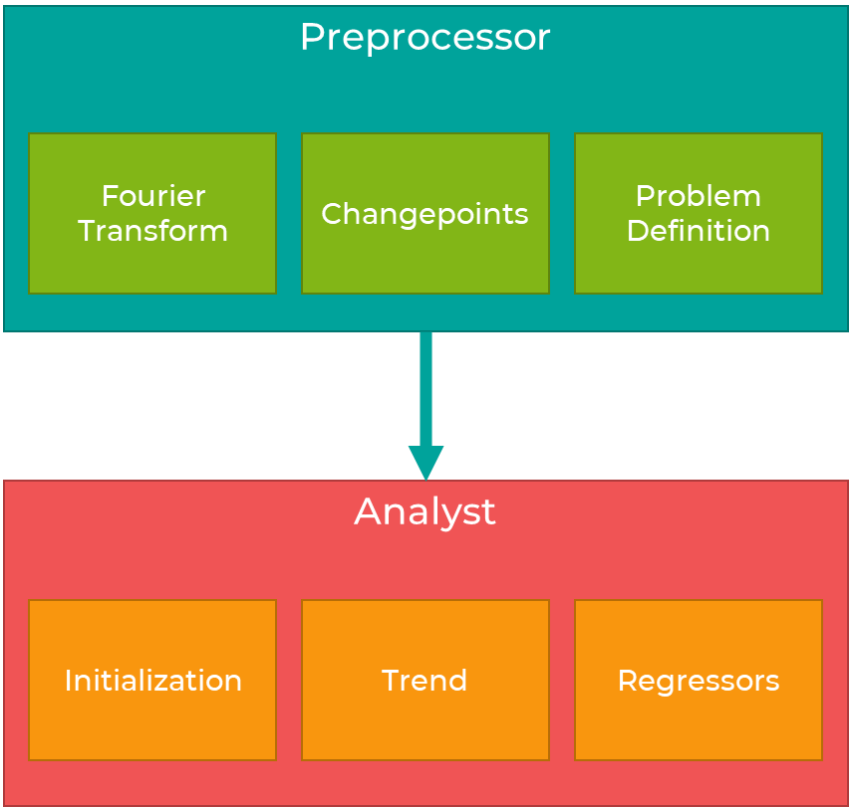

Fig 3.3.2 *– Prophet components summary*

The pre-processor works interleaved with the analyst, forming a communication interface with the analyst, making the process more usable.

The analyst is the core of the Prophet forecasting model and uses a decomposable time series model that combines a baseline trend with the effects of each regressors:

$$y(t) = g(t) \cdot r_m(t) + r_a(t) + \epsilon_t \qquad \textit{eq. 3.3.1}$$

Where $g(t)$ is the trend component, responsible of non-periodic changes; $r_m(t)$ and $r_a(t)$ are the overall effect of, respectively, multiplicative and additive regressors; ϵ_t is the (usually) normal distributed error of the

regression. Seasonalities in this model are converted and treated as regressors of the respective type.

It is possible to describe this kind of regression model in two principal factors: trend and regressors.

The trend is predicted through two different models:

1. Nonlinear, saturating growth - $g_{NL}(t) = \frac{C}{1+e^{-k(t-m)}}$
2. Linear - $g_L(t) = kt + m$

With C is the carrying capacity, k the growing rate and m the offset. Such models are not so flexible to changes in a very wide time period, so it was introduced a series of changepoints that breaks the regression in a piece-wise fashion. Each changepoint is responsible to perturbate the trend behaviour at a specific time.

This modification is implemented by adding to k and m the vector of changes $a(t)$, which holds just zeros in the position before the changepoint date and just ones otherwise, rescaled by the intensity vector (δ and γ):

$$a_i(t) = \begin{cases} 1, & if\ t \geq s_i \\ 0, & otherwise \end{cases} \qquad \textit{eq. 3.3.2}$$

$$\hat{k} = k + a(t)^T\delta, \qquad \hat{m} = m + a(t)^T\gamma \qquad \textit{eq. 3.3.3}$$

$$\hat{g}_{NL}(t) = \frac{C}{1 + e^{-\hat{k}(t-\hat{m})}}, \qquad \hat{g}_L(t) = \hat{k}t + \hat{m} \qquad \textit{eq. 3.3.4}$$

Where s represents the vector of changepoint times. We can describe then δ as the vector of *delta changes*, which are strictly related (exactly in the linear case) to the slope change at the time described by s. The same applies for γ which is just an offset adjustment in both cases. This approach allows to break the problem of the prediction in smaller, consecutive and dependent regressions, giving much more flexibility to the process.

The second factor in the regression is the contribution of extra-regressors described by the user in the model instantiation. A regressor can be intended as multiplicative (scaling the trend), or additive. Each regressor is computed punctually, so in the model output we can retrieve the single component effect in the total prediction.

For what concerning the pre-processor, it formulates the problem starting by the description of the user and encapsulate the data forwarded to the analyst. The first step of the pre-processing is the seasonality extraction

made possible with a fast Fourier transform (FFT). Since in the event forecasting domain there are common repeating seasonalities, the model, by default, understands and extracts daily, weekly, monthly and annual seasonalities automatically. Since the analyst can work just with regressors, the extrapolated seasonalities are converted into a regressor form. Next, if the user had not explicitly selected a set of changepoints, we have the computation of the changepoints which are extracted by selecting at regular intervals from a portion of the dataset (usually 80%).

At the end of the cycle, when the internal analyst produces the results of the stochastic inference, the pre-processor comes in act again by rearranging the outputs in a way understandable by the chosen language interface and, consequently, comprehensible by the final user.

3.3.2 Stochastic regression

As we discussed in the introductory chapter, deterministically forecasting time series for market sales domain is not viable. In this case a stochastic regression would come in hand for many reasons. First, we can't describe a deterministic model for the system; we have domain knowledge about the flexibility of the model parameters and last, we have practically small or no control at all over the extra regressors, we can just make observations from them.

Prophet employs the Stan platform [27] to build the analyst internal engine, which is accessed and interfaced with python or R programming languages.

Now that the principal model steps are described, we can rewrite the general Prophet trend linear regression into the general stochastic regression formula:

$$y(t) = g(t) \cdot r_m(t) + r_a(t) + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma) \qquad \textit{eq. 3.3.5}$$

$$g = g_L(t) = kt + m \qquad \textit{eq. 3.3.6}$$

$$P(g|t, m, k, \sigma) = N(g|m + kt, \sigma) \qquad \textit{eq. 3.3.7}$$

$$Posterior \propto Likelihood \cdot Prior \qquad \textit{eq. 3.3.8}$$

$$P(m, k, \sigma|g, t) \propto P(g|t, m, k, \sigma) \cdot P(m, k, \sigma, t) \qquad \textit{eq. 3.3.9}$$

$$Likelihood = P(g|t, m, k, \sigma), \quad y \sim N(m + kt, \sigma) \qquad \textit{eq. 3.3.10}$$

$$Prior = P(m, k, \sigma, t) = P(m) \cdot P(k) \cdot P(\sigma) \cdot P(t) \qquad \textit{eq. 3.3.11}$$

The stochastic model can be then plugged into the general model formula (*eq. 3.3.5*) assuming a normally distributed error. The result is a model comparable to the description of a generalized additive model (GAM) proposed by Hastie and Tibshirani [28]. This formulation fit quickly with LBFGS [29], backfitting or Newton methods. The Stan platform is then responsible to estimate the maximum likelihood given the set of priors imposed by the user.

3.3.3 Declarative model

One of the targets of the Prophet model is to drive the responsibility of the final user toward the modelling instead of designing a time series regressor. What really makes this possible, is the Prophet declarative interface. Indeed, the human analyst will describe the problem, defining a set of variables for the model (such as the priors, the chosen seasonalities, etc.) as well as a set of extra-regressors and their variables.

3.3.4 Non-positive constraint

Prophet is an event domain related model, indeed the default regressors available include one that is responsible to weight the effect of the holydays on the event number trend. While such a constraint would be effective in this domain, would not be the case for the market sales domain.

Here we are more interested to follow the knowledge that we borrow from the “supply and demand” law, for example.

The first expansion of this work over the Prophet model born from the necessity to constraint the relation of a regressor, which could be represented by the price, with respect to the target column, the sell-out units sold in our case. Following the economic model, the units sold should be inversely proportional to the price. Controversially, this relation does not hold many times during the fit of the Prophet model; often the inferred coefficient of the regressor is negative, meaning that increasing the price induces an augment of units sold. This is a deleterious effect, for example it could demolish any subsequent optimization. To prevent that an optimizer, situated after our model in the pipeline, would diverge toward an infinite price, breaking any expected outcome, we can inject a non-positive constraint in the model to enforce the inference to limit the search toward positive values.

By working on the Stan model, exploiting some of the language constructs, we can constrain a variable to be sampled over a half normal distribution and cut the search for negative values.

Unluckily, Stan does not allow the direct sampling of a model parameter from a distribution, but it is possible to use an intermediate backing parameter which is sampled from the half normal, and then constrain the first to be equal to the latter.

We can dissect a regressor, from the Stan point of view, in order to understand which impact has this modification to the global model:

$$r(t) = \beta_r t + \epsilon_{r,t}, \qquad \epsilon_{r,t} \sim N(0, \sigma_r) \qquad \textit{eq. 3.3.12}$$

Where $r(t)$ can be an additive or multiplicative regressor; β_r is the regressor coefficient and $\epsilon_{r,t}$ its normally distributed error; σ_r is the regressor prior scale. From the equation we can see that imposing the constraint $\beta_r \geq 0$ actually affects only the constrained regressor, leaving the other regressors (including seasonalities) and the trend intact. As a side effect, limiting the inference to positive coefficients can introduce a slight variation in the trend due to the inference iterative methods.

A possible drawback of this constraint is the possibility to infer constrained regressor coefficients close to but not zero. While from an analysis point of view, non-zero coefficients can help to understand the effect and the grade of impact of different regressors, for some post-operations (e.g. multiplications) could be necessary to apply a threshold for the process to be numerically computable.

3.3.5 Piece-wise linear regression

Another important aspect that Prophet does not provide, is the flexibility of a regressor to change its effect over the time.

The β_r coefficient (*eq. 3.3.12*) is inferred over the complete period of observation, resulting in an average of a punctual distribution. The linear model indeed was though for seasonalities and holydays in the event domain, which are not expected to change their effect over time. In market sales domain instead, we have not only the possibility of a regressor to change over time, but also the economic model of "supply and demand", refined by Jenkin in 1870 [5], introduce the effect of diminishing returns in the relation between price and units sold.

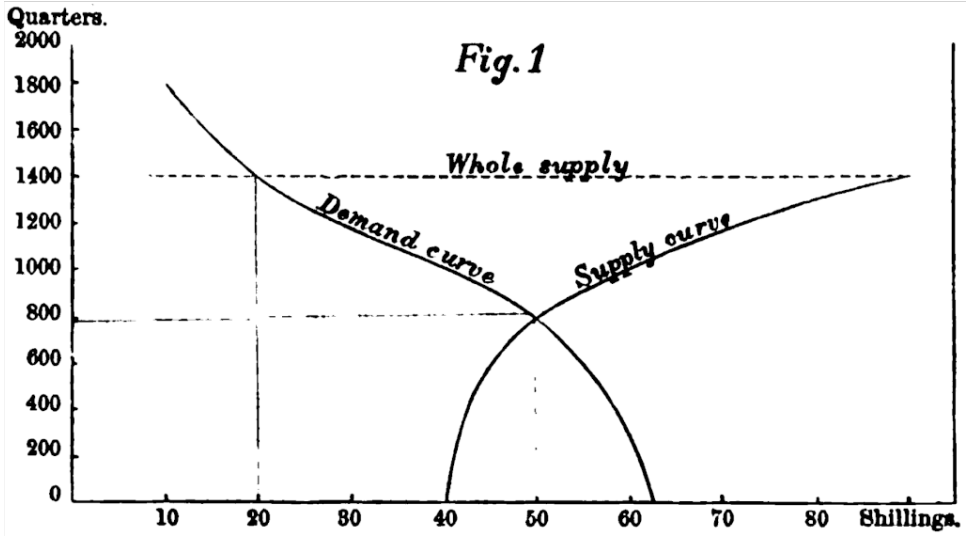

Fig 3.3.3 – Saturating market law, "supply and demand", Jenkin 1870

The most natural way to proceed would be the separation of the problem into smaller piece, leading to a piece-wise forecasting. But, partitioning the dataset for the prediction, such as each part has a different regressor coefficient, would be a mistake because first mess up the seasonalities, which in this way would not be inferred in the whole period anymore, but also it would produce discontinuities in the outcome prediction.

Prophet already overcame to this problem by evaluating the linear trend with a piece-wise linear (PWL) regression. It partitions the dataset through changepoints, which marks certain times for the trend regression where to change (*eq. 3.3.4*). We can exploit the same changepoints in order to change the regression not only for the trend, but also for a regressor.

A PWL regression can fit a regressor over a part of the dataset, includes a benefit effect in the case of diminishing returns [30] and also allows to inject domain constraints with ease (e.g. monotonicity, convexity and non-positive constraints).

Another benefit of this approach, in a stochastic model, is the possibility to use different uncertainties, or prior scales, for different periods. Indeed, we could have parts of the dataset with different variations that we can use to improve the prediction. High variation parts, which could be relative to smaller sellers for example, can be taken in account with a higher prior scale, while regions with a lower variation, such as metropolitan supermarkets, can benefit from a smaller prior scale.

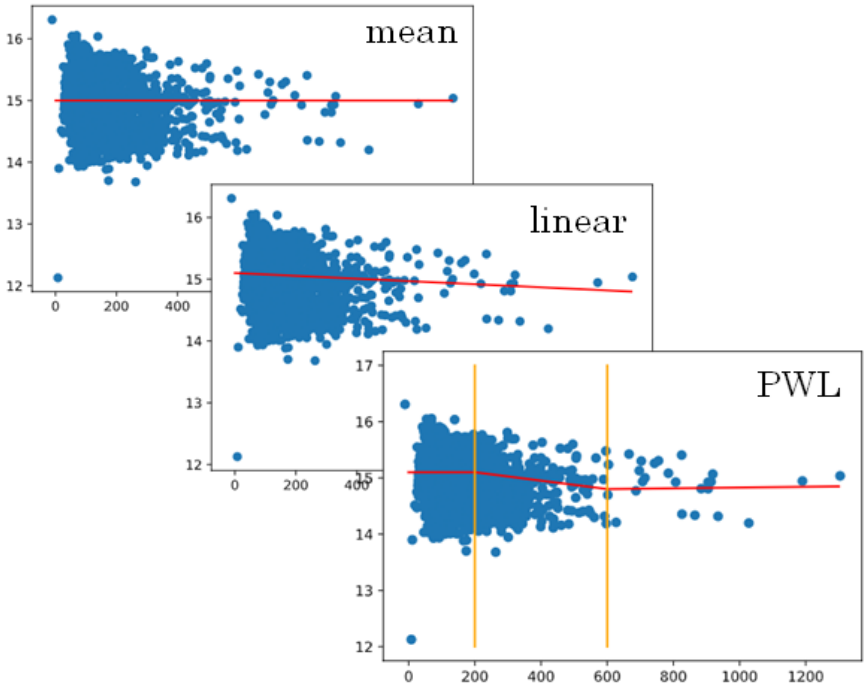

Fig 3.3.4 – Regression over a cloud, comparison

The last problem to take in consideration is the discontinuities in the prediction, which is solvable by better modelling the regression around the changepoints. Instead of "starting from scratch" a new regression at each changepoint, that could introduce the discontinuity, we can model the variation of the β_r coefficient (as δ_{β_r}, *eq. 3.3.13*) such that at each changepoint we have a variation of the slope of the linear regressor, ensuring the continuity in the change interface.

The regression, at the end of all these considerations, takes the shape of this formula:

$$r(t) = (\beta_r + a(t)^T \delta_{\beta_r})t + a(t)^T \gamma_{\beta_r} + \epsilon_{r,t} \qquad \textit{eq. 3.3.13}$$

$$\epsilon_{r,t} \sim Q(0, \sigma_{\beta_r}) \qquad \textit{eq. 3.3.14}$$

The error component $\epsilon_{r,t}$ was chosen to sample from different distributions to give more flexibility during the regression for the final user.

The distribution Q can be one of the followings, some experimental considerations are given for each distribution:

1. Gaussian distribution (normal) – *large and high bell, produces smooth changes at the interface*

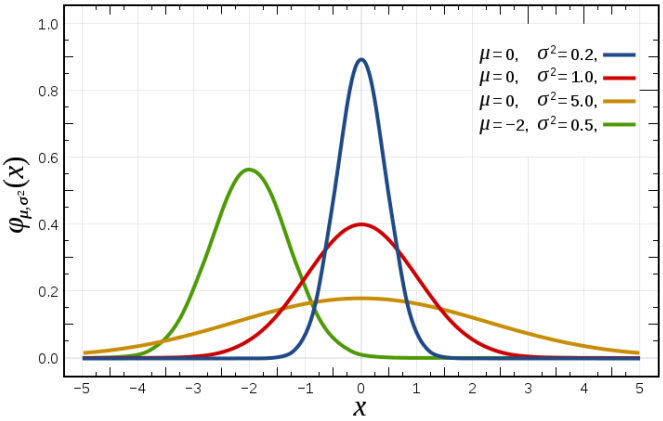

Fig 3.3.5 *– Gaussian distribution –* $\mu = 0, \sigma_{\beta r} = \sigma$

2. Laplace distribution (double exponential) – *low and narrow peak, drastic changes at the interface*

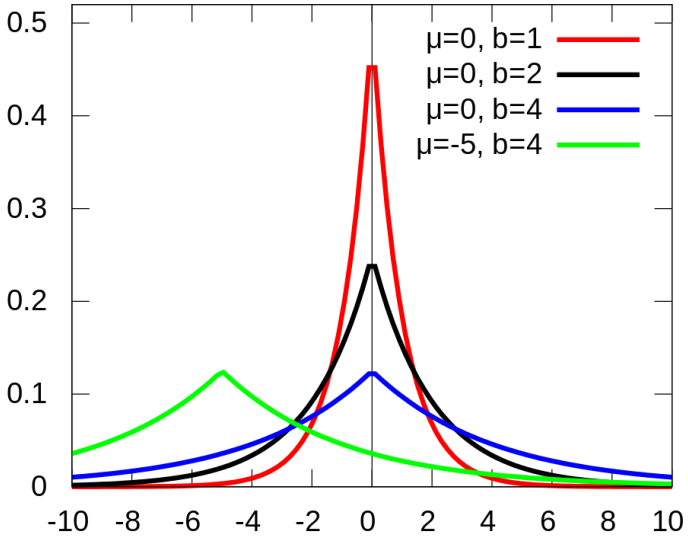

Fig 3.3.6 *– Laplace distribution –* $\mu = 0, \sigma_{\beta r} = b$

3. Cauchy distribution (Lorentz, Breit–Wigner) – *high narrow peak, better for distributions with marked variation, observed performance improvement for datasets with strong seasonalities.*

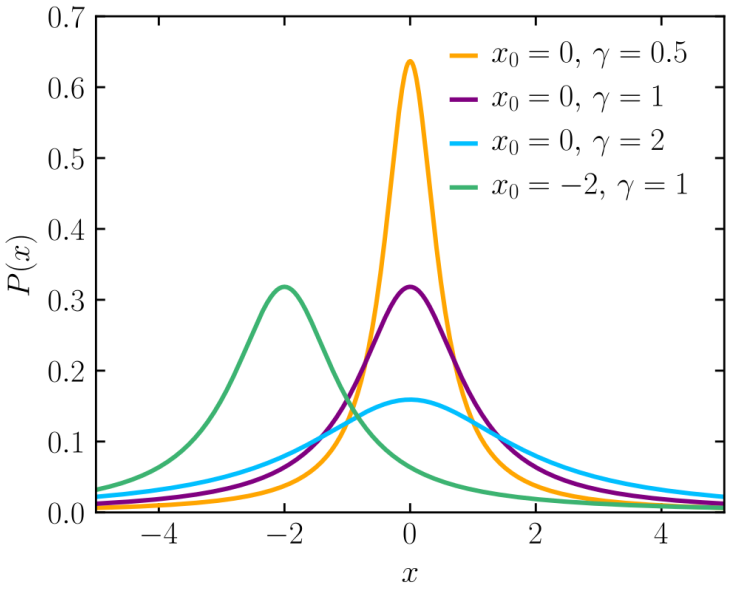

Fig 3.3.7 *– Cauchy distribution –* $x_0 = 0, \sigma_{\beta r} = \gamma$

3.3.6 Modularity

When editing the Prophet Stan backend model, it was followed the modularity principle by stacking the regression constraints over the already established model. This permitted to keep the declarative behaviour of the procedure. Indeed, it is possible to describe the constraints by declaring and adding the appropriate regressor, removing the responsibility of the constraint implementation from the final user.

In this way, following the original Prophet conduct, an analyst with domain expertise can inject non-positive constraints and describe a PWL regressor without worry about the implementation of the framework. Since the additions are modular, during the procedure the framework automatically detects the use of constraints or PWL regressions and act consequently by employing the right regression model for the declared problem.

The modularity is implemented through a selective (usually binary) multiplication which considers just the selected components. The way the selection is performed allows to express an amount n in the range $n \in [0, 1]$ that indicates the component strength of incidence in the global model.

4 Experiments

The evaluation of the procedure stability and robustness was conducted through a set of experiments.

Usually, the regression metric taken in consideration is the Mean Squared Error (MSE), but in the market sales domain we are not interested in punishing too much farther errors. An error in this field means an effective economical loss, so a more precise indicator of the model performance, also considerable in terms of money, is the Mean Absolute Error (MAE):

$$MAE(y, x) = \frac{1}{n}\sum_{i}^{n}|y_i - x_i| \qquad \textit{eq. 4.1}$$

This metric indeed represents the average loss in volume that the model introduces; this was used in all the experiments to strictly evaluate the best procedure in term of performances and also during the search of the baseline model for each discussed class.

Another employed metric is the coefficient of determination (R^2 or "R squared") as an indicative statistic of the quality of the fitting. This coefficient is an index of how well the model can approximate the real data, but in stochastic or non-linear multivariable problems could be inefficient. For this reason, it will be reported just as a qualitative result of our experiments.

$$R^2(y, x) = 1 - \frac{\sum_i (y_i - x_i)^2}{\sum_i (y_i - \bar{y})^2} = 1 - \frac{\sum_i (y_i - x_i)^2}{\sigma^2} \qquad \textit{eq. 4.2}$$

4.1 Synthetic test

A first experiment regards the Prophet PWL model testing. A tiny dataset was generated by sampling the units sold from a composition of a linear trend, weekly seasonality and a noise (sampled from normal distribution); the price regressor was sampled by a relation with the units sold, with respectively a positive linear coefficient (*a*), a negative linear coefficient (*b*), a periodical alternating coefficient (*c*). The choice was made to test that the new procedure can forecast exactly the same value of the old one in the same conditions (*a*); introducing the non-positive constraint induces the price beta coefficient to be negative and close to zero (*b*) and the trend outcome differs from the old procedure, but not the seasonality; lastly that

the addition of the PWL regression for the price can improve the performances of the procedure at least within a simple scenario (*c*).

4.2 Price regression test

After proving that the proposed procedure is capable of improving the performances in a toy environment, we proceeded to test it over a specific dataset. The old procedure, executed over this dataset, produced many incongruences with the economic model of "supply and demand" [5], starting from the incoherent relation between price and units sold.

The price regression test was performed with the aim of assessing the quality of the new procedure over those cases which do not follow the economic model, so it would be possible to employ an optimizer in the process pipeline. The dataset was provided by the MindIt company, which already faced the optimization problem in the process. The availability of a procedure capable to exclude the risk of diverging results, thanks to the possibility of taking in account the non-positive domain constraint, resulted ideal.

The target of this experiment was to test and regulate the non-positive constraint over the price regressor, forgetting about the prediction of the sell-out curve. The investigation then was expanded to include the effects of the appliance of the PWL for the price regressor, to better understand the response of the theorized model with real data.

4.3 Real-case scenario

The real benchmark for the Prophet model was the assessment of the procedure over a dataset coming from a production environment and then we compared the results with the old procedure.

A framework was built in order to evaluate each discussed model and compare their outputs. The framework includes the possibility for a process to use the intra-product validation, and all the models (except for the polynomial regressor, section **3.1.1**) make use of this feature. The framework provides a division for test and train set, ensuring that the number of sellers is sufficient for the intra-product validation and dropping those pairs which do not respect this property.

The experiment was conducted by selecting the best model for each class, excluding the Prophet model. The procedure used to select which would be the most performant model, including seasonalities and regressors, in the case of the Prophet class, was found by running an extensive grid search for each pair in the dataset. The grid search includes 2'160 combinations of which approximately 1200 uses a combination of non-positive constraint and PWL regression. During the search the frequency of each combination were stored, and those that were selected less than 1% of the times were discarded to improve the speed of the process. The dataset contains almost 1800 valid pairs, and the combinations survived to the pruning process of the grid search are 113 and 97 (85.8%) of them make use of non-positive constraint or of PWL regression.

Furthermore, with the proposed approach, we were able to predict better the sell-out curve by assuming that each regressor (price excluded), after the training, would have a value of zero, completely removing its effect. In practice, this separation between training and test prediction is asking to the model to predict the curve, from the past experience, as the only present effect to impact the number of units sold is the price; the rest would produce a variation of the baseline, which we are not interested in.

Once the best model of each class was selected it was trained over the whole test set, composed by a selection of the 80% of all the pairs of each product, ensuring that the number of remaining pairs is greater than 3 (minimal amount for intra-product validation). After the training, the remaining pairs (20% for each product) were used to evaluate the metrics of each model.

Thanks to the intra-product validation mechanism, we can now restart the training process by selecting a different permutation of the train and test set, reinitializing the model parameters. At the end of all the test runs, an informative aggregation of the stored metrics is elaborated to compare the methods and models. The descriptive aggregation includes average, standard deviation, minimum and maximum value of each metric.

5 Results

This chapter will show and describe the results, in terms of metrics (MAE and R^2), obtained in each experiment.

Where possible, it will be reported a comparison between the described models and a briefly description which presents the considerations taken in account during the assessment and after its outcome.

The values written in bold will point out the best performant model or result.

5.1 Synthetic test

The synthetic test produced the expected results: by using a regressor which can be positive the base and PWL models' outcome is the same because we are not involving any type of constraints; in the case where we have the necessity of a negative regressor the performances are slightly improved; in the last case, where the regressor keep changing the relation coefficient, the improvement is less marked but still present.

Here are reported the obtained results:

Benchmark	MAE		R^2	
	Base	PWL	Base	PWL
Positive Linear Coefficient	**312.5**	**312.5**	**0.977**	**0.977**
Negative Linear Coefficient	373.1	**358.2**	0.963	**0.971**
Alternating Coefficient	512.4	**501.8**	0.961	**0.965**

Fig 5.1.1 *– Prophet synthetic test results*

5.2 Price regression test

For the price regression test, the first target was to assess the response of the model to a non-positive constraint. As the results show, the constraint is working as expected, the β_r becomes negative as soon as the constraint is applied.

Here there are two plots showing one of the inversion cases:

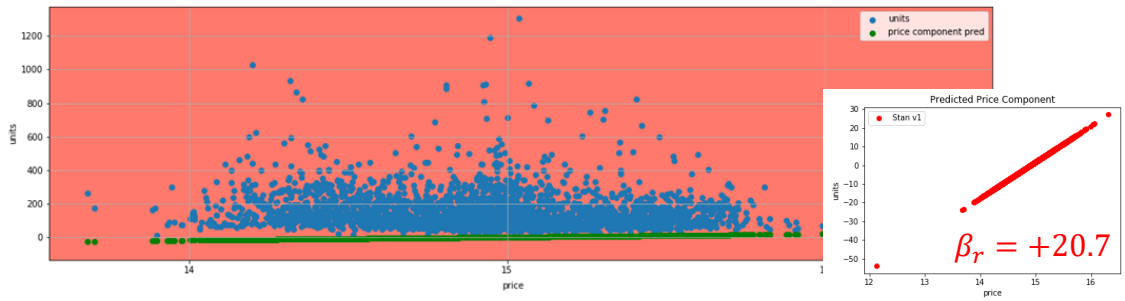

Fig 5.2.1 *– Prophet, example of a price regression, base model*

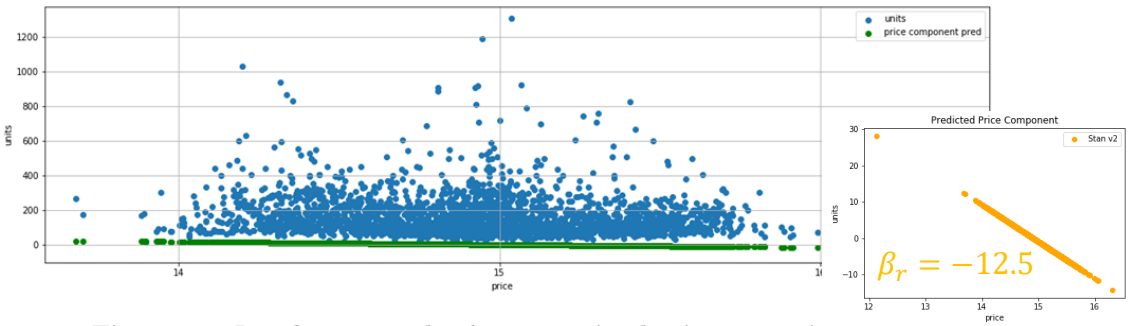

Fig 5.2.2 *– Prophet, example of a constrained price regression, proposed model*

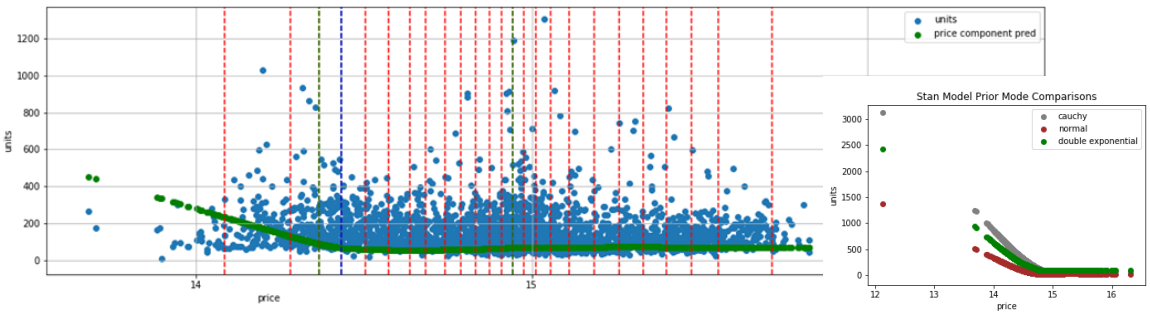

Fig 5.2.3 *– Prophet, example of a price PWL regression, proposed model*
On the right are reported the results of different distributions:
Cauchy (grey), Gaussian (red), Laplace (green)

Here is shown an in-depth analysis of the precedent case, showing how the different models behave on the same data:

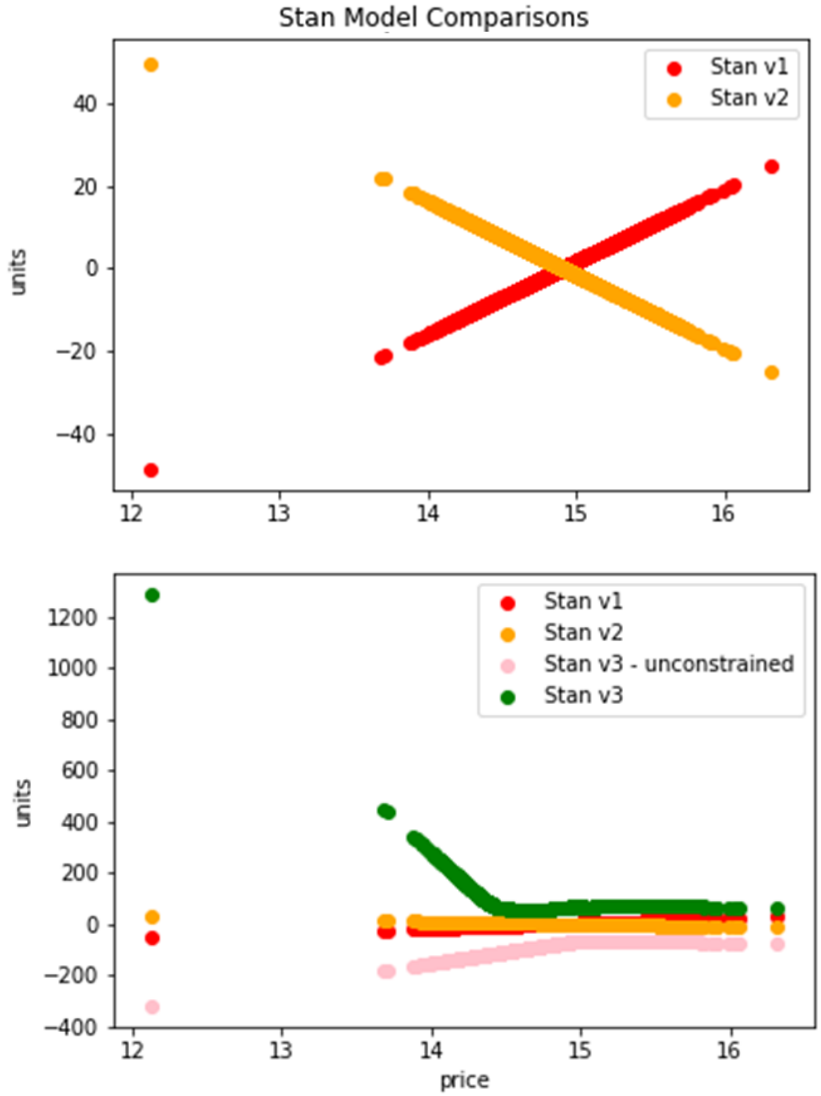

Fig 5.2.4 *– Comparison of base (Stan v1), non-positive (Stan v2), constrained and unconstrained PWL (Stan v3) Prophet models*

Below, the table of the metrics shows how the proposed procedure performs twice as better as the base Prophet model. The introduction of the PWL slightly reduces the performances, which are still improved with respect of the base model. The reduction is probably due to the peculiarity of the price regression task or a non-optimal choice of the changepoints used to partition the dataset, which subsequently introduces discrepancies between the seasonalities and the price regressor, slightly degrading the performances.

Prophet Model	MAE	R2
Base	6366	0.4675
Non-positive constraint	**3867**	**0.5642**
PWL – Gaussian	4284	0.4717
PWL – Laplace	4193	0.4952
PWL – Cauchy	4195	0.4935

Table 5.2.1 *– Prophet models performance comparison*

5.3 Real-case scenario

The last and most important experiment is conducted over the real dataset provided by the MindIt company.

Even in this experiment, we found an improvement of the proposed procedure with respect of the base Prophet model. The improvement is not so considerable as the precedent experiments but still noticeable. The proposed approach performs similar to the most effective model, the Mean model, which is one of the most both reliable and explainable techniques employed in the market sales domain. Furthermore, it is the only approach which decomposes the output in a composition of effects, making easier to understand and analyse the causes of the output.

The following tables show the results obtained in this experiment:

Model	MAE			
	Mean	STD	Min	Max
Prophet PWL	**1483.73**	5412.47	**4.85**	77424.48
Prophet (base)	2788.22	5859.11	17.21	101774.12
Mean	1486.48	**4362.16**	5.48	**65981.98**
Mean windowed	1488.29	4411.57	5.46	67571.30
Polynomial (Bézier)	2171.97	6727.76	10.08	99410.46
MLP	1980.00	7754.54	9.25	147643.40
RNN	4267.92	12550.98	28.11	200329.50

Table 5.3.1 *– Real case model MAE comparisons*

Model	R^2			
	Mean	STD	Min	Max
Prophet PWL	0.553	0.790	0.980	4.58e-08
Prophet (base)	0.511	0.512	0.981	1.12e-08
Mean	0.597	0.303	**0.996**	**6.34e-07**
Mean windowed	**0.598**	0.303	**0.996**	1.42e-09
Polynomial (Bézier)	0.438	0.303	0.988	1.75e-08
MLP	0.411	0.292	0.985	6.98e-09
RNN	0.127	**0.224**	0.935	2.43e-07

Table 5.3.2 *– Real case model R^2 comparisons*

6 Discussion and Remarks

While it may seem not enough to reach, with the proposed approach, a result comparable to the Mean model, with this work we succeeded to reach different objectives.

The first and most important is the explainability of the result. The customers of data analysis companies are usually interested to understand which process the machine followed because they have a strong need of reliability and they want a certain degree of control in their operations. There are also other minor or more specific necessities, sometimes related also to domain experts which need details for further analysis. The decomposition of the effects is a very important feature for the market sales domain. Thanks to the modularity of the Prophet model and the declarative way to describe regressors, it is possible to go back to the regressor which is the source of a certain result.

Another target was the success in the improvement of the Prophet base model. The implementation of a non-positive constraint was a debated and discussed topic in the Prophet GitHub community. The proposed solution successfully addressed the problem and expanded to the implementation of a PWL regression.

One last impressive advantage of this procedure, with respect of the Mean model, is the capacity to work with an arbitrary number of pairs. The procedure, indeed, could predict a sell-in baseline by looking at just one pair of product-seller, while the other models need a huger amount of data.

So, summarizing the evaluation, the success of this procedure does not lay only in the performances, but in the enhancement that provides to the already consolidated procedures.

The Mean model, even if worse in absolute terms, still yield more compact results, gifting to us a suggestion: a point to be taken in consideration in a future deepening could be an extensive analysis of the worst cases predicted by our procedure. The proposed model indeed got a discretely higher MAE standard deviation, meaning that there is a more evident difference between good and bad predictions.

6.1 Limitations

The proposed procedure comes with some limitations. Below, we will discuss about some known limitations and their overcomes, where possible.

The first and most known is due to a lack of interest in the developing process toward undesired complexities. The work revolved around practical problems and use cases, so it was decided not to expand the logistic regression already present in the precedent model. The piece-wise regression is actually available only for linear regressors, while it could be possible to extend the concept also to the logistic regression already implemented for the trend. Such a feature could be interesting only in very specific domains and could also be achieved by simulating it with a dense number of changepoints, distributed with a symmetrical logistic decay (increasing density toward the start and the end of a time period).

Another limitation comes from the modular declarative setting of the procedure. To maintain the simplicity of the work, the model is able to accept non-positive constraints for simple as well as piece-wise linear regressors, but it can't apply directly any constraint to both monotonicity and convexity/concavity of the regression. We can address the monotonicity with the non-positive constraint, which can be intended as a restriction to the first derivative sign, resulting into a non-strict monotonicity constraint by propagating it to each chunk of the piece-wise regression; on the other hand, we have no control over the regressor second derivative, so a chunk could change the convexity (or concavity) of the global function. In theory, we could predict a logistic shape with a constrained piece-wise linear regression, even for those cases we are interested to maintain a certain concavity. A restriction over the second derivative would be more effective in diminishing returns scenarios, exactly what we could expect from the "supply and demand" economic model.

7 Conclusions

At the end of this work, the proposed approach led to promising results in the business time series analysis and forecasting. The advantages offered by the proposed model are the possibility to describe the problem in a declarative fashion; more than decent performances; huge flexibility with regressors and periodicity analysis with seasonalities; last, and very important, the decomposition of the effects involved in the computation which means an impressive degree of explainability. The model also comes with some limitations, such as the unimplemented piece-wise logistic regression for peculiar function shapes, that are also found in the market sales domain.

The work proceeded with a modularity of increasing complexity, starting from the resolution of simpler or synthetic problems, expanding then to more complex and real ones. Each experiment of the work was also driven by concrete necessity in actual and common problems found in company process pipelines. We started by solving the problems related to the optimization divergence due to "supply and demand" economic model inconsistencies, then we added more complexity by expanding the concept of PWL regression over the already present extra regressors.

The final results show also that there are further margins of improving in this field, this work could open different fronts toward the distinction of stochastic and deterministic methods, inferential and neural models.

7.1 Future Work

A natural expansion of this work could be the employment of hybrid and/or ensembled regressor. We could for example use a collection of Prophet models, with a special initialization and a specific construction, in order to lay a variety of results that take in account different type of errors, hoping that they can compensate one each other. Another approach would be the establishment of a chain of models that refine the output in different ways; we could design a MLP or RNN network which takes the proposed procedure output and corrects those parts which are more likely to be incongruent within a certain context.

Another point of possible improvements is the development of a piece-wise polynomial (or spline, PWS) regression which would allow the possibility

to inject a concavity/convexity constraint, given that a PWS (of grade greater than 1) belongs to the second order differentiable functions (PDIFF).

This essay also touched other approaches that are worthy to a further deepening.

A particular attention was recently put toward the time series analysis and forecasting as an image processing task. Future works may include an in-depth comparison of the advantages given by a fully convolutional network employed with the Gramian Angular Field method. This technique is being more and more used in market trading and exchange fields, which are very unpredictable environments, even for human experts, yielding promising results.

Another interesting approach to explore, firstly appeared in the computer vision area and then expanded to other fields, is the study of time envelops through non-local operators, which can simulate and implement a behaviour similar to the self-attention used in natural language processing.

Bibliography

[1] R. Hermann and A. J. Krener, *Nonlinear controllability and observability,* 1977.

[2] P. Diaconis, S. Holmes and R. Montgomery, *Dynamical Bias in the Coin Toss,* Stanford University, Stanford, 2007.

[3] N. Mahmudov, *Controllability of linear stochastic system,* IEEE Xplore, 2001.

[4] P. Groenewegen, "Supply and Demand," *The new Palgrave Dictionary of Economics,* 2008.

[5] A. Grant, The Graphical Representation of the Laws of Supply and Demand, Edinburgh: Edmonston and Douglas, 1870.

[6] W. Poundstone, *Prisoner's Dilemma,* New York - Anchor: 1st Anchor Books ed., 1993.

[7] P. E. Caines, *Mean Field Games,* Encyclopedia of Systems and Control, 2014.

[8] J. T. Connor and L. E. Atlas, *Recurrent neural networks and robust time series prediction,* IEEE Transactions on Neural Networks, 1994.

[9] J. S. Zhang and X. C. Xiao, *Predicting Chaotic Time Series Using Recurrent Neural Network,* IOPScience, 2000.

[10] S. J. Taylor and B. Letham, *Forecasting at Scale,* PeerJ Preprints, 2017.

[11] E. Zunic, K. Korjenic, K. Hodzic and D. Donko, *Application of Facebook's Prophet Algorithm for Successful Sales Forecasting Based on Real-world Data,* International Journal of Computer Science & Information Technology, 2020.

[12] N. Maaroufi, M. Najib and M. Bakhouya, *Predicting the Future is Like Completing a Painting: Towards a Novel Method for Time-Series Forecasting,* IEEE Access, vol. 9, 2021.

[13] S. Barra, S. Carta, A. Corriga, A. S. Podda and D. R. Recupero, *Deep Learning and Time Series-to-Image Encoding,* University of Naples, Journal of Automatica Sinica, 2019.

[14] J. F. Chen, W. L. Chen, C. P. Huang, S. H. Huang and A. P. Chen, *Financial Time-Series Data Analysis Using Deep Convolutional Neural Networks,* IEEE, 7th International Conference on Cloud Computing and Big Data (CCBD), 2016.

[15] Z. Wang and T. Oates, *Imaging Time-Series to Improve Classification and Imputation,* Neural and Evolutionary Computing (cs.NE), 2015.

[16] R. Adhikari and R. K. Agrawal, *Effectiveness of PSO Based Neural Network for Seasonal Time Series Forecasting,* Indian International Conference on Artificial Intelligence (IICAI), 2011.

[17] G. K. Jha, P. Thulasiraman and R. K. Thulasiraman, *PSO based neural network for time series forecasting,* International Joint Conference on Neural Networks, 2009.

[18] J. D. Gergonne, *The application of the method of least squares to the interpolation of sequences,* 1815.

[19] J. A. K. Suykens and J. Vandewalle, *Least squares support vector machine classifiers,* Neural Processing Letters, 1999.

[20] E. T. Y. Lee, *A Simplified B-Spline Computation Routine,* Computing. Springer-Verlag, 1982.

[21] F. Rosenblatt, *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain,* Cornell Aeronautical Laboratory, 1958.

[22] S. I. Gallant, *Perceptron-based learning algorithms,* IEEE Transactions on Neural Networks, 1990.

[23] J. Gamboa, *Deep Learning for Time-Series Analysis,* University of Kaiserslautern, Germany, 2017.

[24] R. G. Hoptroff, *The principles and practice of time series forecasting and business modelling using neural nets,* Neural Computing & Applications , 1993.

[25] M. Miljanovic, *Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction,* Indian Journal of Computer and Engineering, 2012.

[26] E. Waite, D. Eck, A. Roberts and D. Abolafia, *Project Magenta: Generating longterm structure in songs and stories,* Tensorflow, https://magenta.tensorflow.org, 2016.

[27] B. Carpenter, A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. A. Brubaker, J. Guo, P. Li and A. Ridell, *A Probabilistic Programming Language,* 2017.

[28] T. Hastie and R. Tibshirani, *Generalized additive models: some applications,* Journal of the American Statistical Association, 1987.

[29] R. H. Byrd, P. Lu and J. Nocedal, *A limited memory algorithm for bound,* SIAM Journal on Scientific and Statistical Computing, 1995.

[30] M. Gupta, D. Bahri, A. Cotter and K. Canini, *Diminishing Returns Shape Constraints for Interpretability and Regularization,* Advances in Neural Information Processing Systems 31, 2018.

Image References

Fig 2.1	*https://pythontic.com/AutoCorrelation.jpg*
Fig 2.2	*https://upload.wikimedia.org/wikipedia/commons/thumb/3/30/FFT_of_Cosine_Summation_Function.svg/1024px-FFT_of_Cosine_Summation_Function.svg.png*
Fig 3.3.1	*Forecasting at scale* [10], *page 3*
Fig 3.3.3	*https://upload.wikimedia.org/wikipedia/commons/thumb/5/58/Jenkincurves.gif/1024px-Jenkincurves.gif*
Fig 3.3.5	*https://upload.wikimedia.org/wikipedia/commons/thumb/7/74/Normal_Distribution_PDF.svg/720px-Normal_Distribution_PDF.svg.png*
Fig 3.3.6	*https://upload.wikimedia.org/wikipedia/commons/thumb/0/0a/Laplace_pdf_mod.svg/1024px-Laplace_pdf_mod.svg.png*
Fig 3.3.7	*https://upload.wikimedia.org/wikipedia/commons/thumb/8/8c/Cauchy_pdf.svg/1024px-Cauchy_pdf.svg.png*