Universidade de Brasília
Faculdade de Economia, Administração, Contabilidade e Gestão de Políticas Públicas
Departamento de Economia

Matheus José Silva de Souza

RECOGNIZING ECONOMIC BEHAVIOR

Brasília

2023

Matheus José Silva de Souza

RECOGNIZING ECONOMIC BEHAVIOR

Tese de Doutorado apresentada como parte dos requisitos necessários à obtenção do Título de Doutor em Economia pelo Programa de Pós-Graduação em Economia da Universidade de Brasília.

Orientador: Prof. Daniel Oliveira Cajueiro, PhD

Brasília

2023

Matheus José Silva de Souza

RECOGNIZING ECONOMIC BEHAVIOR

Trabalho aprovado. Brasília, 30 de Março de 2023:

Brasília
Março de 2023

FICHA CATALOGRÁFICA

Souza, Matheus José Silva de

Recognizing Economic Behavior / Matheus José Silva de Souza; orientador: Daniel Oliveira Cajueiro – Brasília, 2023. XXX p.129

Tese (Doutorado - Doutorado em Economia) -– Universidade de Brasília/ Faculdade de Administração, Contabilidade, Economia e Gestão de Políticas Públicas / Programa de Pós-Graduação em Economia, 2023.

1. Economics. 2. Machine Learning. 3. Model Evaluation. 4. Artificial Neural Networks. 5. Aspirations. 6. Preferences over Menus. 7. Random Choice. I.Orientador: Cajueiro, Daniel Oliveira.. II. Unioversidade de Brasília. III. Faculdade de Administração, Contabilidade, Economia e Gestão de Políticas Públicas - FACE. IV. Departamento de Economia. V. Recognizing Economic Behavior.

Acknowledgements

I’m thankful to God and the countless miracles He made on my own life and on others’ life as well across the time and space. It is wonderful seeing impossible things happening due to His endless kindness and our faith on Him. I am thankful to God to all the marvelous people I have met here and there. Those people together with His presence make the life immensely blessed, joyful and outstanding. Whoever reading this, be sure that God is all about limitless love and the best about this love is that anyone can reach and feel such a happiness. So, just remember:

Draw near to God, and He will draw near to you. (Jacob 4:8) *Blessed are those who have not seen and yet have believed*. (John 20:29)

Approchez-vous de Dieu, et il s’approchera de vous. (Jacques 4:8) *Heureux ceux qui n’ont pas vu, et qui ont cru*. (Jean 20:29)

Achegai-vos a Deus, e Ele acolherá a todos vós. (Tiago 4:8) *Felizes os que não viram e creram*. (João 20:29)

Nahet euch zu Gott, so naht er sich zu euch. (Jakobus 4:8) *Selig sind, die nicht sehen und doch glauben*. (Johannes 20:29)

RESUMO

O objetivo deste trabalho é contribuir para a compreensão do comportamento humano em termos da preferência revelada. Para tanto, elaborou-se uma revisão de literatura que cobre o progresso da pesquisa econômica em meio aos avanços na tecnologia, sobretudo com relação às novas técnicas de aprendizado de máquina. Argumenta-se que as ferramentas computacionais devem contribuir para o desenvolvimento da economia como ciência. Ademais, propusemos avaliar o desempenho dos modelos teóricos com base no desempenho de modelos de redes neurais a partir de métricas de restritividade e completeza propostas por Fudenberg, Gao & Liang (2020). Também é apresentado um exemplo de modelagem teórica voltada para atualização de crenças dos indivíduos em processos de escolhas de dois estágios. A partir dessa proposta, obtemos, de maneira geral, que: *i*) o *perceptron* multi-camadas mostrou-se uma alternativa promissora para aprimorar a estrutura de avaliação de modelos baseada nas medidas de restritividade e completeza; *ii*) o axioma de reflexividade mostrou-se fundamental na preparação de dados; *iii*) as métricas de restritividade e completeza podem ser utilizadas para construir uma ponte entre um modelo determinístico e outro estocástico, o que contribui para a análise do potencial conjunto dos modelos de entender o processo decisório por trás dos dados; *iv*) a Consistência de Flexibilidade, apresentada em Riella (2013), que é a condição necessária e suficiente para obter uma atualização bayesiana em processos de escolha de dois estágios, sendo que em ambos os estágios menus são escolhidos, é muito próxima da Consistência Aleatória, que é a condição análoga de processos de escolha em que no segundo estágio são escolhidas alternativas do menu selecionado no primeiro momento, muito embora uma não implique a outra.

Palavras-chave: Economia, Aprendizado de Máquina, Avaliação de Modelos, Redes Neurais, Aspirações, Preferências sobre Menus, Escolhas Aleatórias.

ABSTRACT

The goal of this work is to improve the comprehension of human behavior in terms of the revealed preference. To do so, we have made a literature review that covers the economic research progress as the technologies advance, mainly accounting for the machine learning new algorithms. We argue that computational tools should contribute to the economics development. Furthermore, we propose a way to evaluate the performance of theoretical models regarding the performance of neural networks using the restrictiveness and completeness measures proposed by Fudenberg, Gao & Liang (2020). It is also presented an example of theoretical model about the update of individuals' beliefs when facing choice problems with two stages. From this approach, we obtain that: *i*) the multi-layer perceptron seems to be a promising option to improve the assessment framework based on restrictiveness and completeness measures; *ii*) the reflexivity axiom plays an important role on data preparation; *iii*) the restrictiveness and completeness measures can be used to build a bridge between any deterministic model and another stochastic model, which contributes to the analysis of their joint potential to understand the underlying behavior of observable choices; *iv*) the Flexibility Consistency, presented in Riella (2013), which is the the tightening condition to get a bayesian update on two stages choice procedures, in which menus are chosen in both stages, is pretty similar to Random Consistency, which turns out to be the analogous tightening condition when in the second stage the individual choose among alternatives of a previously selected menu, although one condition does not imply the other.

Keywords: Economics, Machine Learning, Model Evaluation, Artificial Neural Networks, Aspirations, Preferences over Menus, Random Choice.

Contents

Chapter 1

Introduction

People behave in many different and unique ways. To understand behavior, researchers split it up into preferences, which are the roots of all human behavior. As a researcher attempts to understand how choices are made, from the simplest, like deciding what to eat at the breakfast, to more complex ones, like deciding between joining a university or following a pro-fisherman life or even deciding a nominee for voting during a presidential election, many views and interpretations are possible, so, it is fundamental to understand how behavior connects to preferences.

Literature seems to walk towards a direction in which an individual cannot be fully characterized by a single steady preference. This way, in most recent behavioral models, individuals have many preferences, like multi-utility representations (Evren & Ok (2011)), and they change according to the knowledge individual gets from the environment she is in or initial conditions progress, like the random utility (Manski (1977), Gul & Pesendorfer (2006), Agranov & Ortoleva (2017)) and update models (Riella (2013)).

On that matter, a discussion that kept its relevance within the progress of decision and behavioral theory is how rationality can be comprised under these many different frameworks. The major separation segregates rational preferences and bounded-rational preferences.

Rationality, according to Clippel & Rozen (2022) is often characterized by a single, stable and well-behaved preference. In this sense, a preference is stable if it does not change according to environmental conditions and can be regarded as well-behaved if it is complete and transitive. As stated by the authors, a bounded rationality model, whose roots come from Simon (1955), is a model that looses some of these assumptions, grounded on the idea that humans' optimizing abilities fall short of supercomputers. The decision maker may have difficulty distinguishing which is the better alternative in a given context because she may not pay attention to all feasible options (Luce (1956), Eliaz & Ok (2006), Manzini & Mariotti (2012b)), she may overlook some alternatives (Luce (1960), Manzini & Mariotti (2012a)), she may be satisfied by fulfilling her preference to some degree (Tyson (2008), Barberà et al. (2021)) and many more.

To expand the scope of investigation of decision theory even further to better understand human behavior, some researchers analyze preferences over subsets of alternatives, called menus. Furthermore, preferences may also be stochastic rather than deterministic. This way, while we gain some flexibility, which allows theoretical models to find better support on data, assigning probabilities of choice for each alternative in a given context, some difficulties arise, namely the blur in the very notion of rationality, since one cannot easily state what is a rational decision in the world of choice probabilities.

The impetus behind the development of stochastic models comes from the assessment of deterministic models on data (Ok & Tserenjigmid (2019) and Ok & Tserenjigmid (2022)). All the way decision theory and science's overall progress is built on comparisons with data, in the sense theorists do theory and empiricists seek correspondence of those theories in real-world behavior. This way, data and model assessment play a central role in the very way theoretical models arise. For example, a specific conjuncture in which current models do not apply in a satisfactory manner makes researchers hectic to build models able to explain it.

Econometrics is the usual field of research within economics which helps with this task, using and developing statistical tools for economic theory problems (Frisch (1933)). Recently, it has been improved by many attempts to incorporate Machine Learning (ML) algorithms into it. ML methods have been widespread across many fields of knowledge, and their prediction power is superb when properly designed since the main goal of machine learning is to learn statistical patterns in such a manner that the acquired relationships could be generalized. That being said, whenever one may want to take advantage of high-accuracy predictions in order to improve decision-making, ML is surely the methodology of choice.

However, some recent papers evaluate economic models on data and compare their results with ML models' predictions (García-García et al. (2022), Hillel et al. (2021)) and suggests ML models are more suitable for economic matters than the economic models themselves. This way, we aim precisely at a question discussed by Cranenburgh et al. (2022), who asks “*Is the machine learning paradigm strong enough to overthrow the four-decade dominance of choice models?*”.

We highlight that this kind of comparison should be made to improve economic research and not to replace it, since the fields could help each other progress. As pointed out by Brathwaite, Vij & Walker (2017),the microeconomic framework should be used for interpreting ML models, and as stated by Fudenberg & Liang (2020), ML should be used to discover regularities in data that are not captured by economic models.

On this matter, we also provide a novel approach to evaluate models or finding structure in data grounded on the concepts of two measures developed by Fudenberg, Gao & Liang (2020), Fudenberg & Liang (2020), namely restrictiveness and completeness. The fresh touch to their proposed method is the use of Machine Learning (ML) to reveal known and well-documented structures on unknown data. In particular, we focus on Artificial Neural Networks (ANN) to help with the task of understanding data under the light of the wide economic theory research. We aim to answer the question “*Is this data set compatible with which behavioral structure in which degree?*”, fostering the idea that ML can help the development of economics since this method allows a better clustering of data sets, for example. This way, from a range of distinct models of preference of interest we seek which of them has a higher degree of correspondence in data and then uncover some structure of the behavior data set.

To make our method explicit, we compare a set of bounded rationality models that deal with reference points in a choice procedure (Guney, Richter & Tsur (2018) and Silva & Riella (2020)) together with some ANN designed to predict choice behavior. This way, the novel approach of this work stands for the exploration of new evaluation metrics proposed by Fudenberg, Gao & Liang (2020) to comprise a way to better evaluate economic models using ML models as yardsticks. To the best of our knowledge, this is the first attempt to use ML on the evaluation structure and not simply comparing economic models to ML.

The contribution of this work is three-fold and they are made on the following structure: Chapter 2 provides a comprehensive literature review of economics many shapes of rationality, going through econometrics, as well as the application of ML techniques. In the sequence, in Chapter 3 we propose a method in which ML is used to hone the evaluation of correspondence of theoretical models in the reality, recognizing behavior in data. Finally, Chapter 4 shifts the focus to economic modeling itself, contributing on how agents may update their beliefs in a dynamic environment from inner perspectives about the world.

Chapter 2

Economics, Econometrics and Machine Learning: A Literature Review

2.1 Introduction

The goal of this paper is to address the relationship between theoretical economic modeling and its object, namely the real world, and its empirical tools, mainly econometrics and the recently borrowed tools from Machine Learning (ML). We point out some papers on the vast and extensive literature regarding economics development as a science, which comprises its etymology (Leshem (2016)), its definitions (Smith (1977), Robbins (1932)), its tools (Meyer & Conrad (1957)), its rationality (Sen (1987), Booth (1993)) and how it fits or explain reality phenomena. Furthermore, we intend to provide a literature review that covers the foundations of economic modeling but also serves as an up-to-date to researchers concerning the most recent tools applied to data-driven studies on economics.

We first go into the roots of theoretical modeling, discussing the notion of rationality and how it stands as modeling goes from deterministic to stochastic frameworks. Rationality can be seen as any reasoning that does not contradict itself, thus making sense on its own and leading to a conclusion about its object, but it also comprises the usual notion of rational behavior, which is closely related to the idea that when individuals are confronted with a set of alternatives to choose, they choose what makes them happier and not the opposite. Next, we focus on the reality test of models and the tools used for it, ranging from regression models of econometrics to more advanced tools of machine learning.

We aim to build a comprehensive and tightening literature review, which adds value to current literature, allowing one not to only discover main the contributions to the area but how they evolved and how to think the modeling and assessment matters.

This paper is placed in the academic literature as an attempt to face the clash between economics and their methods in a more harmonic manner, trying to justify that these last cannot overshadow the first. To the best of our knowledge, there is a gap in the literature of papers that discuss the relationship between theoretical and experimental modeling, since most papers are focused on modeling itself or empirical tests or literature surveys, but the works that bridge economics and its tools, focusing on the main role of theory in this relation, like did in Gilboa et al. (2018), Gilboa et al. (2022b), Gilboa et al. (2022a), Gilboa & Samuelson (2022) are scarce.

To guide the reader, there are three pillars below this literature review. They are all described by Gilboa et al. (2022b), who put economic theory in three big subsets: *i)* economics, which examines economic phenomena; *ii)* methods, which deals with the use and development of analytical tools; and *iii)* methodology, which examines the scientific endeavor of economists. In those terms, in this work, we refer to economics as theoretical economics and econometrics, and ML are comprised on the methods. As for methodology, we address this question by considering economics as a useful compass for meaning.

To motivate our attempt, we highlight that, regarding theoretical economics and science overall, often good models are simple, not making complex assumptions about the complex reality. It is clear that a model reaches its objective when it helps to understand the world and not when it emulates the world, which usually does not improve the comprehension of reality. As Box (1976) remind us, the models are all wrong, but some of them are useful. Thus, we may infer that in building a simple model we work towards building a useful model as well.

That being said, the most powerful way to decide if a given model is useful or not is to confront it with its object, which turn out to be the reality. To wit, suppose a choice model which claims that people have desires that influence their actual choice under some circumstances like did in Guney, Richter & Tsur (2018) or Silva & Riella (2020). When a researcher tests such a model with data of a certain conjecture, it may be able to answer if those individuals' desires play a role in decision-making according to the model or not. In the case the model does not find support on this conjuncture, that is not sufficient to conclude that the model is wrong or useless. However, if for many tests with different data sets, the model always fails to get any support, then this model is probably close to the useless threshold.

It is important to note that even useless models have their space in the development history of science. Recalling our previous choice model with desires, if it does not have support in the real world at all, it at least points out that modeling desires' effect on the choice procedure is not a good direction to follow or the way it models that effect is particularly not right, which should lead to questions to the model's assumptions. This way, new models can arise and explain the world we live in.

The main tool to do these real-world tests and aid economics' progress is econometrics. On this matter, no one can discuss the great relevance of multiple linear regression. However, we cannot say our choice model with desires does not hold for a particular data set simply by doing a linear regression and finding statistically non-significant coefficients. A meaningful test in this case must start from the same assumptions the model makes, which is called structural economics (Keane, Todd & Wolpin (2011)).

However, as econometrics progressed, developing better tools, it tried to be independent of economics. It definitely should not be seen as an independent science looking for meaning on itself far from economics. Recalling Goodhart's law, present in Strathern (1997), "*when a measure becomes a target, it ceases to be a good measure*". That idea comes from a usual phenomenon in which a field of research is born to be useful to another, but as it improves and develops, researchers attempt to view it as an independent field. That being said, the advance of ML techniques, which is a completely independent field of research from economics, made econometrics borrow many tools, developing more robust strategies and improving itself, fostering its attempt to become independent.

In this sense, many empirical works simply aim to compare ML algorithms' performance on data with that of economic models (García-García et al. (2022), Hillel et al. (2021), Lee, Derrible & Pereira (2018)), which little contributes to economics progress, because a theoretical economic model hardly can perform better than an ML tool, since it is not designed for predictions like ML is. In the rare case in which that happens, such an experiment simply shows that there must be something wrong with the algorithm itself. That particular relationship between economics and ML is one of the main issues we address in this work.

We organize the paper as follows: Section 2.2 cover economic modeling and how the rationality idea can be seen in deterministic and stochastic setups, as well as the advantages and disadvantages of both of them, which is closely related to the ability to find support on real-world data, which is discussed in Section 2.3, which introduces econometrics approaches of reduced form and structural estimations of parameters; Section 2.4 turns the focus to recently developed ML tools and how they confront with economic modeling, covering many applications. Finally, Section 2.5 presents the conclusion with final remarks on the purpose of this literature review.

2.2 Deterministic and Stochastic Modeling in Economics

The main goal of this section is to discuss the root of modeling, which turns out to be the rationality notion behind it, and how it can be translated into deterministic and stochastic models in economics. Although a simple model has a higher probability of being useful, to do so, it is crucial to build a rationality idea that finds some correspondence in the world, being able to understand and explain it.

The main concept of rationality may vary across the several fields it appears in. In Choice Theory, a choice function or a choice correspondence is often said to be rational if it arises from the maximization process of a well-behaved preference relation (Ok & Tserenjigmid (2019)), that is complete and transitive. That interpretation is closely related to that presented by Ok & Tserenjigmid (2021), in which a rational individual makes choices in large menus in a consistent fashion when compared to the pairwise choices between alternatives, which does not to happen any kind of inattention. The literature on expanding this notion of rationality is vast, so it may still hold if the preference relation is incomplete (Ribeiro & Riella (2016), Ribeiro (2023)), which means the individuals may face some indecisiveness when choosing, which leads to overthinking and inattention situations.

Although rationality is a pretty fluid concept, no matter how it may change, it is always more intuitive and easier to catch under deterministic modeling (Ok & Tserenjigmid (2021)), in which choice models, for example, point out exactly what is chosen as the circumstances change. This way, under the rationality of a model which states that a certain individual always chooses x when facing x and y, choosing x is considered rational and choosing y is not a rational behavior. The main disadvantage related to deterministic models is their strictness in their choice predictions. Recalling the previous example, according to Apesteguia & Ballester (2020), if in any experiment the individual chooses y over x, the model does not comprise any explanation for that and it should be regarded as an error. For the authors, a perfect fit of that model in the real world requires that we must always observe the models prediction to hold, no matter how many times we repeat the experiment.

In the same sense, it is intuitive to believe that individuals are supposed to prefer more money than less. However if for a given data point we find, for example, an individual who chose a lower-income job instead of a higher-income job when both were available, it could be taken at first as irrational, but we are not taking into account that the effect of the time is crucial on decisions like that. To address this matter, according to Gul, Natenzon & Pesendorfer (2014) real data regarding consumption choices across multiple periods violates the weak axiom of revealed preference due to several reasons. They highlight the existence of income effects, intertemporal complementarities, preference for variety and dynamic expectations. This way, a deterministic model cannot capture if the lower-income job was chosen because the individual valuation of leisure changes dynamically over time and other circumstances or because the experience individual gets from that job helps her on finding an even higher income job in the next period and so on.

Decision difficulty is a matter closely related to intertemporal environments. It is addressed by Agranov & Ortoleva (2017). Performing two kinds of experiments regarding multiple choices, namely one with distant choices from each other, and another with explicit repetitions in a row, they find that stochastic choice is behaviorally consistent and present almost always on hard questions, those in which one cannot point out a better option than the other. Those hard questions are more related to which career to follow, rather than which career is better, which gaming monitor to buy, rather than which monitor is better or where to eat today, rather than which food you prefer. This way, when it decides with a real impact on someone's life, there is a lot more uncertainty.

Considering that some decision-making setups do not show an obvious correct alternative, like in the hard question situations, Gilboa & Samuelson (2022) propose to use decision theory as a coherence test. To those hard questions, we sum up the intertemporal issues, like deciding which career to follow or which main subject to study when joining a university. Even on those questions, one may simply argue that the right choice is the one that maximizes the utility function, but it should be accounted that no one has perfect knowledge about future events or even about their inner selves.

However, may be a difficult task to picture an utility function for some scenarios and this matter is still addressed by Gilboa & Samuelson (2022), who show concerns about it. That being said, when they propose using decision theory as a coherence test, they argue that from revealed preference assumption, it is possible to justify observed choices through a previous known model, that is, the researcher should ask herself "*there are beliefs that justify this choice?*". Basically, it would be a interaction between decision maker and decision theory in which the decision maker shows all the information he knows and ask what should she do and decision theory reply grounded on a set of models. Such an incredible view is applied to problems of all natures, we just require to exist scientific models about the matter of interest. Furthermore, since the authors suggest a convenient way to implement this interaction in real world situations, through a smartphone app, showing that there is a profitable commercial value on it.

Many deviations from standard rational model like described above lead some researchers to make the assumptions of rationality more loose, mainly flexibilizing the completeness and transitiveness. This allowed to uncover new behavioral structure, which is now well documented across the literature, being called bounded rationality models. Bounded rationality is still deterministic but comprises a set of behaviors that standard rationality cannot accomodate but we can explain them in a rational manner. It then led to the development of models based on multi-utility representations, lack of attention (Avoyan et al. (2022)), overthinking (Kahneman (2003), Rubinstein (2007)), stochastic choice and many others.

The issues brought by the lack of flexibility on deterministic models lead economic modeling to flirt with stochastic models, which deals with real world data better, fitting its uncertainties better at some extent. Ok & Tserenjigmid (2022) states, *ipsis litteris*, that "*Among the reasons behind the choice behavior of an individual taking a stochastic form are her potential indifference or indecisiveness between certain alternatives, and/or her willingness to experiment in the sense of occasionally deviating from choosing a best alternative to give a try to other options*". As for deterministic modeling, when individual is indifferent between x and y, we cannot determine the frequency in which any of them is chosen. Furthermore, being able to capture a somewhat willingness to experiment an alternative becomes important because people overall don't know their own preferences with perfection, that is why they change constantly over time. This way, once a researcher captures this effect within a model, she can better understand the whole process behind dynamic preferences. This way, the authors statement makes clear that the transition between deterministic and stochastic models is a matter that really could expand the comprehension of real world behavior.

Regarding the indecisiveness matter, a matter closely related to classical axiom of complete preferences, experimental studies show that it exists and lead to randomization (Eliaz & Ok (2006), Ok & Tserenjigmid (2022)). In this sense, Costa-Gomes et al. (2022) found on an experiment with real goods that agents are leaned to sacrifice present utility through a costly choice deferral and that makes the postponed choices more consistent with rationality idea. This finding foster that deterministic models of choice can be more accurate when dealing with one-period decisions without a great impact on future events. Whenever an agent may face decision problems with a big change to her future, pondering about probabilities arises naturally to better deal with risk.

However, when we go to stochastic modeling, the rationality idea becomes quite dim when a model assumes the world is ruled by probability distributions, since we do not observe probabilities in data, but they are convenient to understand how events, may it be natural or psychological, interact and produce outcomes that we can observe. For example, Lillie does not wake up everyday with probabilities on mind regarding taking the long road or the short road to go to the workplace, although she is likely to take the short road, to save gas money. However, she knows that on a given day, if it rains, there is a traffic accident, a friend ask for a ride, she is troubled with something or she just want to try a new way, the chances of taking any of the ways may change, but what we actually observe us the way she chooses.

The same way, Abraham does not choose probabilities when he goes for a dinner, let alone roll a dice to decide what to choose. Of course we may impose him to choose among lotteries when doing some research, but in the end of the day, he chooses a meal. However, it probabilities capture better how his preferences show up on given circumstances, like eat alongside his wife, his mother, his friends or alone. This way, modeling revealed preferences in stochastic framework may comprise such situations better.

Across the literature of probabilistic decision models, the Random Utility Model (RUM) is considered the best rational benchmark for a stochastic model. In this sense, Ok & Tserenjigmid (2019) say that the RUM could be taken as the definition of rational stochastic choice under the interpretation of the frequency of individuals on a society that happen to choose an option among a set, since it simply accounts the choices of a group of deterministically rational agents. Nonetheless, when the RUM is interpreted on the basis of a single individual choice, who is supposed to have multiple preferences which cover her whole tastes and personality, the rationality idea becomes dim.

The very first RUM model goes back to Marschak (1959). The author propose to combine data from controlled experiments on pairwise comparison between options, surveys of choice on large data sets and the temporal effect over total consumption. To do so, the study assume choice obeys a probability distribution, which lead to different kinds of choices even from the same problem. This way, RUM was born due to the inconsistencies found on individual behavior, Manski (1977) points out. The author consider the behavior to be deterministic over the set of alternatives, but the model maker only knows part of the information, instead of considering the agent rolls a dice (whatsoever it may look like) to choose. Theoretical researchers attempted to save the rationality in the terms they knew. When a rational model does not explain some individual data, rather than claiming individual is irrational, model makers claim those agents do not know all the information regarding the world and the alternatives' features. This way, preferences and thus utilities becomes random variables. That viewpoint is present on relevant papers like McFadden (1973) and McFadden (1976).

2.3 Economics and Econometrics

When a theoretical model seeks support on data, attempting to fulfill its purpose of explaining some real world phenomena under its own rationality notion, it resorts to a wide range of techniques, tools and strategies. In particular, economists often apply statistical tools to their matter's scope, what is the so-called econometrics. Econometrics, differently from Statistics, was born as a tool to serve economics.

In this sense, Hendry (2009) states that there is no way to dissociate applied econometrics from an economic theoretical framework, in such a way that a pure statistic model fails to interpret its findings. Doing so, it is like being lost in the high seas without a compass to guide the way. Furthermore, the author claims that since economic theory is not complete, correct and immutable and never will be, one cannot justify an insistence on deriving empirical models without theory. On the same line, Frisch (1933) states that, *ipsis litteris*, no amount of statistical information, however complete and exact, can by itself explain economic phenomena. If we are not to get lost in this overwhelming, bewildering mass of statistical data, we need the guidance of a powerful theoretical framework. Without this, no significant interpretation of our observations will be possible.

On the top of “atheoretical” econometrics or reduced form estimation, there is the ARIMA models, statistical tools used often to understand the statistical features of data, allowing to prepare it properly for future research. It is also called Box-Jenkins method, since it was introduced by Box & Jenkins (1970), and often figures out as a initial step to decompose time series data, so one cannot infer false causalities or derive misinterpretations, since they deal with data based solely on its statistical features, like auto regressive, integrated and moving averages process. Some applications are presented in Meyler et al. (1998), Dongdong (2010), Kiriakidis & Kargas (2013), Kharimah F. (2015), Yang et al. (2016), Barnichon & Garda (2016), Dritsakis & Klazoglou (2018). While it may be useful for understanding statistical features and may be able to do predictions, it has little to contribute to economics development and is far from replacing its contributions, as it is usually suggested.

Probably their ease to implement and understand lead to several meaningless applications in economics (Keane (2010)). Within such tests, theoretical models often get discarded because doing a regression with a data set does not support model theory, but note that the parameters estimation cannot translate model’s rationality if it does not assume the same assumptions as the model. To wit, the regression model often assumes the hypothesis of classical linear regression model in order to estimate its parameters and a given behavioral model supposes another completely different range of assumptions. This way, ignoring the model’s hypothesis and simply estimating a regression could not verify model’s relevance.

That being said, doing econometrics together with economics is now labeled structural econometrics, which brings to the parameter estimation task the hypothesis of the analyzed model. It shows many benefits: it enables to deal with many endogenous variables (Hausman (1983)), to deal better with consumer elasticities analysis (Reiss & Wolak (2007)), conduct welfare analysis under a dynamic environment with changing prices (Nevo & Whinston (2010)) and makes possible to do serious counterfactual analysis, crucial for policy design (Su &

Judd (2012)). The applications of structural modeling are many. For example, there are researches on investment on human capital Keane & Wolpin (1997), on the effect of social security and medicare on retirement behavior (Rust & Phelan (1997)), on adverse selection at auto insurance (Jeziorski, Krasnokutskaya & Ceccarini (2017)) and on industrial organization, labor economics and marketing (Arcidiacono & Miller (2013)).

2.4 Economics and Machine Learning

2.4.1 Origins and Classification of Machine Learning

More recently, the approach to capture statistical patterns of data was significantly improved with Machine Learning (ML) algorithms as they provide sumptuous advances on detecting non-linear interactions often not easily understood, uncovering hidden relationships in data (Goodfellow, Bengio & Courville (2016)). Naturally, econometrics borrowed the valuable contributions of ML to strengthen the reality tests of economics.

Regarding the origins of ML, it goes back to the use of computational tools to understand data. Some contributions on the very beginning are undoubted remarkable, like the concept of perceptron introduced by Rosenblatt (1957), which can be seen as a computational implementation of Hebb (1950) cell assembly, which comprises a chain of neurons and the connections between them. Back then, since the author algorithm's dealt with image classification based on probabilistic principles, being previously trained with labeled data, it was classified as a supervised learning algorithm.

Further in time, the algorithm developed by Samuel (1959), designed to play checkers, which was able to improve itself from past experience, is considered by many the very first in ML history, that, *in verbis*, is the field of study that gives a computer the ability to learn without being explicitly programmed. Nowadays that specific field of ML is called reinforcement learning, in which algorithm learns from the payoff it gets from past experiences.

Aside from supervised and reinforcement learning, there is also the unsupervised learning. ML is classified as unsupervised if the computer is able to learn with unlabeled data, that is, based on mathematical and statistical similarity between data points, the algorithm is able to learn statistical pattern previously unknown. This way, a computer can cluster observations based on a probability distribution (like Density-Based Spatial Clustering of Applications with Noise algorithm from Ester et al. (1996)) or on spatial criteria (like k-means algorithm, whose origins goes back to Steinhaus (1957), MacQueen (1967), and hierarchical clustering, whose roots are mainly find in Sibson (1973), Defays (1977)), for example.

One of the very first unsupervised algorithms were related to association rules, which got spread with Agrawal, Imieliński & Swami (1993), who developed an algorithm able to finding relationships between products on a supermarkets, detecting which bundle of goods are often bought together, so they can be arranged close to each other and increase sales, thus improving comprehension about purchase behavior.

2.4.2 On the Use of Machine Learning in Economics

Undoubtedly, ML is able to provide a whole range of new tools to econometrics, as pointed by Varian (2014), who covers classification and regression trees, random forests, LASSO (which stands for "*least absolute shrinkage and selection operator*") and Spike-and-Slab regressions, which helps on variable selection issues. The author still addresses important matters for practitioners like causality, prediction and uncertainty, providing comprehensible economic applications, like effect of race on mortgage using regression trees and variable selection on growth questions using LASSO.

However, it is extremely important to perceive that the use o ML on economics should be done according to the scientific method. According to Mullainathan & Spiess (2017), ML solves a different kind of problem when compared to econometrics as well. While ML focus on the problem of prediction, that is, $\hat{y}$, econometrics revolves around parameter estimation, that is, $\hat{\beta}$, whose meaning is associated to the impact or a variable on another. It is clear then that the frontier between them is quite thin and there is an intersection on their applications, since the estimation of the impact of a variable on another, allows one to make predictions. Nevertheless, it is important to perceive that, on their roots, their scopes are not identical. The key point the authors highlight is that applying ML to economics requires finding relevant $\hat{y}$ tasks, drawing attention to the fact that ML algorithms are now technically easy to use, what could be risky, when they are naively applied or their output is misinterpreted.

On the same way, when comparing ML to standard empirical techniques, Kleinberg et al.

(2015) clearly explains that the last are not optimized for prediction tasks, since they focus on unbiased estimation, which creates problems for out-of-sample results. Ordinary Least Squares (OLS), for example, does not allow the trade-off between variance and bias because the OLS predictions vary in sample. On the other hand, ML does not minimize only in-sample error, but also takes into account a regularizer that penalizes functions that create variance. This regularizer is calibrated by a parameter that, according to the authors, can be seen as the price between variance and bias, but can even be chosen using the data on more fancy and sophisticated approaches.

Turning specifically to the comparison between theoretical economics and ML, some other aspects also deserve attention. Theoretical economics, as previously discussed guides interpretations of the real world and gives meaning to related empirical works, since they are, before everything, grounded on a set behavioral hypothesis. Machine learning, on the other hand, gets its fundamentals from optimization methods designed to learn statistical patterns from data and generalize the acquired knowledge, that is, to capture hidden structural features of data. Although their applications are manifold and distinct, there is also an intersection, which turns out to be the predictions on behavioral environments.

In this sense, Parkes & Wellman (2015) makes a didactic distinction. While the *homo economicus* is a mythical perfectly rational agent constructed by economics, the *machina economicus* is a synthetic *homo economicus*. The same way economists know people slightly differ from *homo economicus*, not because they are crazy or irrational, *machina economicus* is a tool to better understand the interactions of *homo economicus* on complex environments, this way leading to better understanding of real people on some contexts.

Following this standpoint, theoretic models are designed to understand behavior that is previously characterized in terms of assumptions, not to fit data. That viewpoint is somehow a theoretical economists uprising, which is precisely shared by Gilboa et al. (2018). The authors propose a reflexion to think economics as a tool to point out weakness of arguments, that is, to offer critiques. A field of research does not need to fulfill human predictions desire to be scientific. They put the question in formal terms, formulating a model of economic modeling to foster that predictive side is separated from the criticism side of economics.

Given an economic model, it may happen that for some conjuncture the model's prediction may be or not well accommodated. A high evaluation for some data implies that analyzed individuals indeed act according to the the model's structure, since there is a tightening bidirectional result. A poor evaluation though reveals that some model's structure does not match those individuals. If it does not hold for any conjuncture, it can be improved instead of completely discarded. However, when a ML tool, trained and designed to generalize the learned patterns, fails for some data, the main goal of the algorithm fails, since its main objective failed, what is far from being the case of an economic model. To foster that, Gilboa et al. (2022a) reiterates that a theory does not die as soon as it is refuted. Since that happen, it can be refined or restricted. Even on the extreme case it is completely refuted, the conceptual framework could be used to come up with other related theories that fit data.

That being said, we argue that economics should play an important role on interpretability of ML models, since they are often seen as black-box algorithms. In this sense, a black-box algorithm is a computational tool that performs well on its prediction purpose but fails on the interpretability of its findings. According to Doshi-Velez & Kim (2017), interpretability is the ability to explain or to present in understandable terms to a human.

This way, often ML fails that because the basic process in which the learning happens is well known, like the calibration of weights and bias at an Artificial Neural Network (ANN) through a clear specification of crucial functions like loss and optimizers. However, usually, little is known about the hidden patterns found by the algorithm and how the output is generated. The authors still separates ML's interpretability into two branches: one that consider an algorithm interpretable if it is useful on a practical issue and the another only does it if the researcher claim that it is interpretable, providing a reasonable explanation for its findings. According to their first branch, almost all ML becomes interpretable, what express a false idea of understanding behind overall ML. This way, their second concept looks more accurate.

Some studies, then, attempt to obtain some behavioral interpretation from ML, seeking their interpretability to satiate avid researchers hungry for knowledge. In this sense, Montavon, Samek & Müller (2018) claims that being able to interpret machine learning algorithms finds its relevance mainly on fields like medicine and self-driving cars, in which wrong decisions may have a fatal cost. For the authors, one key aspect on the improvement of ML transparency is to identify the most important input variables, which they admit that figures out as a post-hoc interpretability, applicable when there is no control on the model structure at all.

Wang, Wang & Zhao (2020) aims to provide a way to compute all economic information from ANN and highlight some challenges for interpreting them on the light of economic structure. Their notion of interpretability is related to the recovery of crucial economic variables like elasticity, willingness-to-pay, marginal rate of substitution and consumer surplus. However, according to Gilpin et al. (2018), interpretability is not a sufficient feature for achieving trust on machines behavior. To do so, humans requires explainability, which is closely related to the ability of a algorithm to defend its actions, provide relevant responses to questions and be audited, the authors say. Thus, explainability is a stronger condition which assumes interpretability.

Looking from a side of the prism, explain a machine learning algorithm behavior is in fact easy, since they act to optimize an objective function which leads to the “discovery” of hidden patterns, which should be generalized in order to make predictions. What is hard and nearly impossible is to explain their behavior under legal, ethical an moral aspects, because they are often not taken into account when designing algorithms.

Such a task only becomes possible if the modeler makes those features clear on the root of the algorithm, as evident constraints that the process must face when optimizing. In this case, ML originally learns patterns assuming what cannot be done under the light of ethics, law and moral. So, the problem behing the lack of interpretability and explainability are, in the end of the day, the lack of structural constraints when designing these algorithms and that is where the real challenge lies, that is, translating ethical concerns to constraints to be incorporated to the algorithm structure. Of course if we go from a regular optimization problem to a constrained one, we do not reach the first-best solution, but that is a reasonable cost to design trustful algorithms.

Regarding the developed tools to provide some interpretability to black-box, Molnar (2018) and Samek et al. (2021) covers some of them, like LIME (Ribeiro, Singh & Guestrin (2016)), which stands for local interpretable model-agnostic explanations, global surrogate method and Shapley values of features, from Shapley (1953). The global surrogate method is often applied on engineering fields to replace a high cost resource for another substitute without a significant loss on the outcome. When applied to ML, it does seek to interpret a black-box using another surrogate black-box that approximates the first one. Likewise, as the very name implies, LIME seeks interpretation without any model. The Shapley value comes from coalitional game theory, which assigns payouts to players (or features in the ML case) depending on their contribution to the total payout.

We highlight that all these ways to interpret ML do not make the interpretability issue as transparent as it should be, since they mainly deals with post-hoc interpretability, simply trying to find a reasoning to justify the outputs of an algorithm. Sometimes, in order to do so, some works use ML itself to seek interpretability of ML, as pointed out by Molnar (2018). Such a task is dramatically different from the attempt to uncover the structure of black-box models in such a way humans can understand properly and trust. As pointed out before, interpretability and explainability should be a structural matter on ML algorithms, so they can be build ground on a interpretable foundation (Molnar, Casalicchio & Bischl (2020), Alwosheel, Cranenburgh & Chorus (2021), Martín-Baos, García-Ródenas & Rodriguez-Benitez (2021)).

On the other hand, supporting economists uprising to view their theory as a meaning compass, Andrews et al. (2022) find that structured economic models outperform ML models on out-of domain tests, but not on prediction errors regarding out-of-sample tests. A domain essentially differ from a sample because it comprises a more general class of problem. It could be a set of data samples from a research problem of same nature. For example, different time-periods could represent the general notion of a domain. They argue that black-box algorithms slightly fail on identifying structure that is commonly shared across domains, not being able to effectively extrapolate behavior between sets of features. This way, economic models allow to recover regularities that are general across a variety of domains.

2.4.3 Empirical Researches

Regarding the applications of ML on economics, they are essentially two-fold: applied researches comparing the prediction power of ML against economic behavioral models; and those which propose ML as a tool to improve economic modeling itself. We think that both branches are closely related and can be summarized only on the last one, since the competition between their prediction power makes no sense at all, as previously argued.

Even before the huge tide of ML models, one of the first works to compare their learning performance with economics models were Cohen & Axelrod (1984), in order to highlight some way to improvement. The authors study if a dynamic model of preferences fits data better than a standard utility maximization model with a non-varying preference on an environment where a factory manager maximizes output allocating fixed available labor hours between production and maintenance. The preference changing mechanism of their Artificial Intelligence (AI)[1] is driven by experience, as a reinforcement learning, but not necessarily conscious. They find the dynamic model based on a learning AI is better not only when stability is reached but throughout the entire process of adaptation. That is an evidence the agents are aware that their previous beliefs about the environment they are in were incorrect or incomplete, requiring to be updated in order to achieve better satisfaction levels.

As for applied researches of ML on stock market, for example, Souza et al. (2019) investigated how well ML algorithms perform on cryptocurrencies trading in terms of ability to make profit and compare them with standard methods of algorithmic trading, like technical analysis. They explore if support vector machines (SVM) and ANN could improve market efficiency or exploit its inefficiencies, since, according to Kim (2003) ANN minimizes empirical risk, that is, classification error, and SVM minimizes structural risk, accounting for the generalization of the model and penalizing its complexity. They find that while SVM are more compatible with risk-averse investors, since it ensures conservative returns even accounting for the risk and transaction costs, the ANN has a great potential to exploit short run inefficiency, allowing to obtain better profit during strong bull trends, even compared to buy-and-hold strategy.

Another application to finance is presented in Huang, Chen & Wang (2007), who highlight the performance of SVM on credit scoring analysis, which compared to ANN, genetic programming and decision trees needs less input data on training to reach same accuracy level, contributing to the minimization of risk of creditors. The authors also address the issue of over-training, which leads to over-fitting, that does not favor the generalization ability of the algorithms, and the black-box nature of overall ML, whose advantage is not requiring a previous knowledge of relationships of input and output variables. Thus, they suggest combining ML with "*other more interpretable models*". On this matter, Deng et al. (2017) focus on ANN on a deep and recurrent framework tested on stock and commodity future markets, dealing with real-time financial trading. They find that those tools can be in fact efficient alternatives to technical analysis indicators, which has a high-level of ad-hoc parameters.

[1]As a disclaimer, AI and ML are not interchangeable terms. According to Kersting (2018), for example, AI is a broader field which focus on making a machine behave as a human. On the other side, ML is part of AI which is related to the ability of learning of a machine, even if it is not explicitly programmed to do so.

Finally, Wellman & Rajan (2017) states that the use of AI on financial world reduces transaction costs and brings more efficiency but at the cost of legal and ethical concerns. It is due to the difficulty of distinguishing legitimate from illegitimate actions performed by autonomous trading agents because they are not built to act according to law or ethics. Instead of that, they are built to optimize, no matter the nature of the variables being used or the side effects of their optimal behavior. To get around this issue, as previously stated, the whole process of modeling a machine and building the algorithm should comprise clear constraints that capture what could be done and what is unacceptable.

Aside from financial applications, on policy analysis, Andini et al. (2018) study how decision trees optimize tax-rebate policies in order to accomplish the purpose of increased consumption. Their reason for using decision trees instead of other algorithms is a matter of transparency, since it provides a more transparent decision rule. They found that policy effectiveness can be increased, but they highlight that accuracy is not a sufficient metric to validate the model. Accuracy is a trustful indicator when the underlying data set of the classification task is balanced, with similar proportions of positives and negatives. Complete analysis also comprise another indicators like precision or negative predictive value, related to the prediction power respectively according to true predictions, and recall or specificity, mainly related to the prediction power with true data points.

On the context of a changing world in which agents perceive they must update their beliefs, Holland & Miller (1991) shows that Artificial Adaptive Agents (AAA) can acquire sophisticated behavioral patterns, showing that the observation of the course of learning can increase the very understanding of economic issues. Those are models based on pure linguistic descriptions that are infinitely flexible. While they are often logically consistent according to the authors, mathematical (and hence economic) models have consistent structure and allow general solution techniques, at the cost of losing flexibility. On this incredible flexibility, some words from the authors must be highlighted as a criticism: "*The possibilities (of AAA models) are so rich that it is often difficult to predict on a priori grounds what behaviors and structures will emerge*".

One of the most recent contributions to the use of ML in order to improve economic research is presented by Fudenberg & Liang (2020). According to them, one can use ML to identify regularities not captured by an incomplete theory, that is, with low correspondence in some data set. Aware of the aspects of real data that the model does not comprise, the researcher could develop better models that encompass them. Furthermore, they state that for complete theories ML can identify if that good fit is due to the capacity of the theory to comprise any possible data or if it reveal behavior on real world.

This way, it is clear that ML models not only compete with economic models, they also have been used to improve economics and decision making, thus improving policy design, for example. On this matter, according to Hrnjic & Tomczak (2019), ML allows to reconsider past decisions and to reevaluate the possible outcomes, reducing time to make a decision as well as the cognitive cost of judgment. Kleinberg et al. (2017) cites the problem of judges' decisions of releasing or not arrested criminals. They ask if a judge sentence could be improved by ML algorithms designed to fine-tune the probability of an arrested criminal on committing another crime or flee for example, allowing to reduce jail populations without increase the crime rate.

Another relevant example is present in Kang Polina Kuznetsova & Yejin (2013), who analyze social media relation with public health surveillance. The authors use the costumer review analysis of restaurants as a tool for accessing hygiene of restaurants, improving disclosure and decision making regarding when and where to conduct inspections. Their work was a prominent contribution to literature specially because they use Natural Language Processing (NLP) to deal with text data. Their algorithm has over 82% of accuracy in discriminating severe offenders from places with no violation.

When it comes to NLP, according to Hirschberg & Manning (2015), it speaks to the use of computational techniques for the purpose of learning, understanding and producing human language content. Nadkarni, Ohno-Machado & Chapman (2011) says that it is not a simple matter of text information retrieval, since it comprises the whole processing of natural language data. A great part of NLP is focused on Sentiment Analysis (SA), which is the identification of some feedback from reviews produced by humans.

That being said, regarding the use of NLP and SA on economics, we highlight Hansen, McMahon & Prat (2017). The authors analyze transparency matters on Central Banks policies. Their findings support the theory which claims that transparency improve accountability and behavior, leading to a positive discipline effect and a negative conformity effect, from agency theory. They mainly use Latent Dirichlet Allocation (LDA) method to reduce dimensionality of input text data from two different sections of meetings from Federal Open Market Committee and find that, during the meeting, the use of numbers and data as well as the number of topics discussed by rookie members (positive discipline effect) is higher. However, during policy discussion, those inexperienced members discuss less topics and engage in a herding behavior (negative conformity effect). It shows an important contribution for the interpretability of large texts, regarding macroeconomics directions. An extensive review of this topic can be found on Algaba et al. (2020).

Aside from NLP usage, regarding image processing in economics, Henderson, Storeygard & Weil (2012) proposes estimating Gross Domestic Product (GDP) from satellite images. They claim that satellite night-lights serves as proxy for economic activity at temporal and geographic dimensions. This finding allows to better estimate poor quality data or growth on sub and supranational regions where it is not accounted. They also find that this methodology has a bigger potential on improving measurements of economic activity on low-quality national accounts data.

Still on that matter, Naik, Raskar & Hidalgo (2016) use outer space images of the Earth to reveal details of health, education, mobility and criminal rates of regions. They pre-process data with Geometric Layout algorithm and then train a Support Vector Regression (SVR) algorithm, further analyzing the effect of government spending on a region and the physical urban change they observe in data. Their method provide an accurate estimate of the physical urban change after the government intervention. For a complete and detailed review of this literature, one may want to check Donaldson & Storeygard (2016).

Peysakhovich & Naecker (2017) study how economic models can incorporate ML. Using LASSO and Ridge regression. They find that their algorithm basically learns expected utility with probability weighting model from data under risk. However, as for ambiguity, the patterns that the algorithm learns slightly differ from theoretical works, showing a gain. It is rather important to clarify that risk supposes the individual knows the world framework and the probabilistic structure. Ambiguity, on the other hand, happens when she does not know for sure the probabilities assigned to events. That being said, ML points out that there are regularities on the ambiguity setup that expected utility models do not capture, pointing out a gap and a way theory could evolve.

Another relevant work that combines ML and choice models is Sifringer, Lurkin & Alahi

(2020). The authors propose new choice models called Learning Multinominal Logit (LML) and Learning Multinomial Nested Logit (LMNL) from combining ANN with those traditional discrete choice models. They attempt to develop models based on standard framework of logit models improving parameter specification as well as predictability of the model, without losing their high level of interpretability. To do that, they integrate ANN with the utility specification of a discrete choice model to discover the utility specification through data. This way, they weaken the assumption these models make that model specification is previously known to estimate its parameters. Thus, behavior is modeled first from a knowledge-driven part, which specifies the model, and then data-driven part to optimally calculate the utility form and parameters.

Other researches can address other features that ML lacks when compared to standard methods. For example, Tehrani, Cheng & Hullermeier (2011) point out that monotonicity is far from being guarantee on ML algorithms in general. They propose a way to improve the logistic regression, which shows three distinct features, namely the linearity, the compreensibility and the monotonicity. The authors change the linear restriction that logit model imposes between inputs and output for the Choquet integral, allowing to keep the interpretability of the model as well as the monotonicity but also capturing nonlinear dependencies. Furthermore, it is one of the first works to embed Choquet integral in the ML context.

Bräuning et al. (2017) propose a way to generalize lexicographic preferences in order to better understand them in a data-driven experiment. The relevance of such preferences comes from the need of decision makers to evaluate simple criteria when choosing options among overwhelming amount of psychological and environmental features. Lexicographic order ranks options according to the payoff of the first eligible attribute. Their generalized approach is based on decision trees and group some attributes in order to compare multiple criteria at the same time.

On that same matter, Hüyük, Zame & Schaar (2022) discuss preferences based on multiple and competing objectives. They provide an intuition for multi-utility models, which they call multi-objective such that the individual priorities over objectives are lexicographic. They attempt to infer multi-objective reward based representations of decision maker's observed preferences through a stochastic preference model, called LORI, from Lexicographically-Ordered Reward Inference. The main idea is to find a reward function that makes the observed behavior as optimal then acquired function reproduce optimal policies through reinforcement learning. Their contribution is to employ the method to find multiple multi-objective reward functions, this way improving and understanding better behavior.

Further in this task of learning individuals preferences over a set from the revealed preferences, we highlight Busa-Fekete, Hüllermeier & Szorenyi (2014). Like the so-called random utility models in Decision Theory, the authors assume there is a fixed and unknown probability distribution over the set of all total orders on a given set of alternatives. They aim to understand this distribution, like the most likely top-item and the highest frequency preference from pairwise preferences data, instead of individual evaluations. To do so, they assume the distribution is a Mallows model (from Mallows (1957)), since it is better tractable, as being an exponential distribution that is parametrized and based on distance. This way, they manage to guarantee confidence on their predictions as well as minimizing sample complexity.

Finally, since the researchers curiosity can lead them anywhere, there are even attempts to improve ML itself with economic theoretic contribution. Ahuja, Choudhury & Dandapat (2022) extend the very concept of production functions to performance functions to evaluate multilingual models based on NLP algorithms, such as some focused versions of Bidirectional Encoder Representations from Transformers (BERT). Accounting for the costs of collecting machine-translated and manually-created data, they point out the efficiency of their method showing that optimal performance is attained with minimum cost if there is some amount of manuscript data in the target language, since there is a positive cost for machine-translation.

2.5 Conclusion

This paper presents a comprehensive literature review on economics' way of progress as a science. We covered from the roots of modeling, discriminating deterministic and stochastic models, and the tools used to test models' potential to expand the understanding of the world, which turn out to be econometrics and, more recently, with contributions from machine learning. The very way models arise and fall pave the road of progress for economics, that is the reason why reviewing such a rich literature have its relevance.

We mainly aimed to provide a work that serves as an up-to-date overview so readers could get familiarized with the economics' reasoning and recent works that use Machine

Learning algorithms to understand economic phenomena. This way we intend to provide a helpful work for researchers planning to do research on this field.

In this work, we also address several issues concerning economics with econometrics as well with Machine Learning, such as the naive comparison between ML and economics models performance on data, disregarding that each one has a different objective or the excitement around the measure that makes it the target, which blurs the actual target, as stated by Goodhart's law. The reflexion about these issues and the implications of them draws attention to the advantages and disadvantages of some research directions.

Chapter 3

Identifying Structure on Data using Machine Learning

3.1 Introduction

On this chapter, we propose a way to evaluate how well economic models fit real world data based on the metrics of restrictiveness and completeness introduced by Fudenberg, Gao & Liang (2020). We say a model identifies data structure well if it is restrictive and complete at the same time. That being said, our main contribution is methodological and three-folded:

1. we use Machine Learning (ML) tools, namely Artificial Neural Networks (ANN) to hone models' evaluation results, may it be deterministic or stochastic. This way, one may be able to point out which model from a selected range of theoretical modeling strategies better identifies some data set structure. This allows one to compare theory-based models between them or even help to decide which ML model is better trained and better generalize some behavioral structure;

2. as for choice data preparation step, dealing with randomly generated data sets we also find that imposing reflexiveness axiom to the sample improves significantly the meaningfulness of the evaluation. On the other hand, imposing preference's completeness does not show any significant improvement for the meaning of evaluation;

3. we also use evaluation metrics to build a bridge between the notion of rationality and real behavior grounded on economic theory, that is, we explore the potential of two models, one deterministic and other stochastic, to explain rationality on behavioral data and understand its uncertainty.

Regarding the metrics, restrictiveness speaks to the potential of a model to be compatible with many random models and completeness is related to the compatibility of a model and a given data set. Both lie in the unit interval in such a way that if both are close to one, the model is restrictive and for the particular data set it is complete and thus identify its structure regularities well. Those measures are calculated regarding a benchmark, which normalizes the models' assessment and is called naive mapping by Fudenberg, Gao & Liang (2020), since they set it to be a simple model that any reasonable model could outperform. However, we call this benchmark yardstick mapping, since it can vary and change the evaluation results.

This way, instead of considering the benchmark map as the map based on the uniform distribution as did by the original work, we vary the yardstick mapping among four options and find out that setting it to be an ANN improves the ability to identify data structure according to the models, specially when it turns out to be a Multi Layer Perceptron (MLP).

Note that on one hand, restrictiveness is related to a specific model potential to comprise many different modeling outcomes when compared to a yardstick previously chosen. On another hand, completeness measures the degree of affinity of a model with collected data regarding the yardstick. That being said, when a model is highly flexible, showing a low restrictiveness, it is expected to have a high completeness for many data sets, which is due to model looseness to comprise many behavioral frameworks and not because it has the ability to unfold data structure. However a highly restrictive model with high completeness for a given data set indeed reveals much about its structure, since it can approximate only a few mappings good but for a given conjuncture it shows a great correspondence degree.

Our purpose is to assess decision models with data sets, revealing a degree of compatibility of a pair theory and data set. This way we can verify if, for example, the rational model reveals more structure on a specific data than a selected bounded rationality model. This helps on explaining some specific conjuncture or behavior, answering the question “This data set is justified by a theory to which extent?”, which leads to a more accurate discussion of further policy analysis, for example. It can lead to a cluster method to separate data sets within a selected models setup, which can lead to new research tasks. Lastly, we address a long time issue in which ML algorithms are often seen as a candidate to replace economic modeling itself, instead of a tool to aid its further development.

Our conducted tests evaluate two frameworks of models, one deterministic and another stochastic. Under the deterministic setup, we analyze the bounded rationality models developed by Guney, Richter & Tsur (2018) and Silva & Riella (2020) and compare their evaluations with deterministic models obtained from ANN previously trained, running their output probabilities of choice to get a binary output. As for the stochastic, we compare the assessment of ANN to a simple logit model, which is the usual benchmark for this kind of problem.

The chapter is organized as follows: Section 3.2 presents a literature review covering details on restrictiveness and completeness, artificial neural networks and the bounded rationality models from Guney, Richter & Tsur (2018) and Silva & Riella (2020); Section 3.3 details the assessment framework, presenting the choice framework, the models' parametrizations, the considered yardstick mappings and other computational details; Section 3.4 untangles the synthetic data we used for our tests; Section 3.5 presents results for deterministic and stochastic models and analyzes how these results change as the choice setup allows more conceivable alternatives; Section 3.6 presents an attempt to bridge deterministic and stochastic models using restrictiveness and completeness measures in order to identify their joint potential to explain data in terms or behavior and uncertainty; and finally Section 3.7 presents concluding remarks.

3.2 Literature Review

3.2.1 On the Measures

Here we cover the details of the two main measures proposed by Fudenberg, Gao & Liang (2020) used to identify structure on data sets considering some economic models. The purpose of the authors is to identify if a model fits data due to its potential to capture real world regularities, describing well the underlying behavioral structure, or simply because it is unrestrictive and flexible to accomodate all sorts of behavioral functions or correspondences. Their work is related to information criteria like Akayke Information Criterion (AIC) from

Akaike (1974) and Bayesian Information Criterion (BIC) from Schwarz (1978), used on model selection and parameter inclusion, since meaningful parameters, according to Fudenberg, Gao & Liang (2020), are the ones which brings a small decrease on restrictiveness and a high gain of completeness.

To address and detail the assessment method and the formalities, consider first we are interested in some aspect of the real world and we model it in such a way real situations are mapped to a model predictions according to the assumptions of the world it makes. For example, under the framework of decision theory, a model maps choice problems, that is, a set of alternatives that are feasible or observable or both, into choices, denoting what is actually chosen when individual faces a specific situation. That being said, following their notation, let $\mathcal{F}$ be a set of all possible mappings for a given problem.

The primitives of the the evaluation structure are a set of permissive mappings $\mathcal{F}_{\mathcal{M}} \subseteq \mathcal{F}$ and the distribution μ which rules the draw of random mappings from $\mathcal{F}_{\mathcal{M}}$. To provide a better understanding on $\mathcal{F}_{\mathcal{M}}$, consider the problem of modeling probabilities. In this case, $\mathcal{F}$ is the set of maps whose image lies in $\mathbb{R}$. However, since we are modeling probabilities, $\mathcal{F}_{\mathcal{M}}$ is restricted to the mapping whose image lies on $[0, 1]$. In another case, when modeling choices from a set S, $\mathcal{F}_{\mathcal{M}}$ is restricted to the mappings whose image lies in S.

The model we want to evaluate, which is in fact a parametrizable model, is represented by $\mathcal{F}_{\Theta}$, which comprises all parametrizations f_θ of that model, including an yardstick mapping $f_{\text{yardstick}}$ fixed by the researcher. Finally, consider a set of randomly generated mappings $\{f_m\}_{m=1}^{M}$ from $\mathcal{F}_{\mathcal{M}}$ according to μ and a metric d which evaluate the distance between two mappings. This metric d indicates how far the two mappings' predictions are. It could be, for example, the Mean Squared Error (MSE), Binary Crossentropy (BCE) or Categorical Crossentropy (CCE). This way, the discrepancy δ_m between a model $\mathcal{F}_{\Theta}$ and a map f_m, according to Fudenberg, Gao & Liang (2020), is given by:

$$\delta_m = \frac{d(\mathcal{F}_{\Theta}, f_m)}{d(f_{\text{yardstick}}, f_m)} \in [0, 1] \tag{3.1}$$

On the Eq. (3.1), $d(\mathcal{F}_{\Theta}, f) = \inf_{f_\theta \in \mathcal{F}_{\Theta}} d(f_\theta, f)$ and since $f_{\text{yardstick}} \in \mathcal{F}_{\Theta}$, it is clear that δ_m lies in the unit interval. That being said, they define the restrictiveness r of the model as the expected value of the discrepancy over the set of random mappings, that is, $r = E[\delta_m]$. Note that $f_{\text{yardstick}}$ is used to normalize the distance calculations. Thus, large values of δ_m

implies that the model being analyzed does not approximate f_m much better than the yardstick mapping does. A model is said to be restrictive if these values are often high, which means it does not have the ability to approximate a random set of M mappings better than the selected yardstick.

On the other hand, completeness is defined by the authors as:

$$\kappa^* = \frac{e_{P^*}(\mathcal{F}_{yardstick}) - e_{P^*}(\mathcal{F}_{\Theta})}{e_{P^*}(\mathcal{F}_{yardstick}) - e_{P^*}(\mathcal{F}_{best})} \tag{3.2}$$

On Eq. (3.2), $\mathcal{F}_{yardstick} = \{f_{yardstick}\}$, $\mathcal{F}_{best} = \{f_{best}\}$, f_{best} is the mapping that best approximates the true and unknown mapping f^* between X, which is the observable vector of data inputs, and Y, which represents the observable vector of data outputs, and e_{P^*} denotes the error between a map and a given data set according to the joint distribution P^* of (X, Y). Since we observe only a sample of the data, we do not know P^* and to evaluate completeness, the authors propose the following estimator:

$$\kappa^* = \frac{CV(\mathcal{F}_{yardstick}) - CV(\mathcal{F}_{\Theta})}{CV(\mathcal{F}_{yardstick}) - CV(\mathcal{F}_{best})} \tag{3.3}$$

Here, CV denotes the out of sample error of the model according to the k-fold cross-validation procedure detailed at Fudenberg, Gao & Liang (2020). Although completeness values lies within the unit interval, since the estimator in Eq. (3.3) uses the out of sample error CV, it may end with negatives values in the presence of some usual issues of data analysis, like overfitting or when the validation portion of the data is not a good replica of the training set. To wit, suppose a single parametrization of the model f_θ. This way, $\mathcal{F}_\Theta = \{f_\theta, f_{yardstick}\}$. During the k-fold cross validation process, if the minimal error mapping on the training set is f_θ and it shows a higher error on the test set than $f_{yardstick}$, it turns out that $CV(\mathcal{F}_\Theta) > CV(\mathcal{F}_{yardstick})$, which implies that completeness may get a negative value.

That being said, before going into the details of our experiment, we do a brief review of ANN structure and the bounded rationality models we chose then we present our results and conclusions.

3.2.2 On Artificial Neural Networks (ANN)

Artificial neural networks (ANN) are widespread learning tools often referred as supervised ML, although they can be used on unsupervised tasks as well. Nowadays, due to computational progress and the huge amount of bundled built-in methods, ANN are quite simple to understand and use. According to Charpentier, Flachaire & Ly (2019), ANN are semi-parametric models. One can view ANN as semiparametric when they act as a supervised algorithm, since the researcher specifies the shape of the output, but do not point out the form of relationship between the variables. That being said, when ANN are employed on unsupervised learning, they can be seen as non-parametric models. Thus, it is clear, that ANN is ideal to capture non-linearities on data.

In short, a set of inputs is transformed into a set of outputs according to a predefined setting made by the researcher in a similar fashion the logistic regression does. However, in order to better learn hidden features of data and generalize them, there is a number of hidden layers between the input and output layers in which data is transformed several times. The transformation process occurs on each of the smaller units of layers, called neurons, and the chain begins with the first layer transforming raw or input data. Each neuron of each layer takes the outputs of previous layer and transform then through the multiplication by weights and addition of a bias. The obtained value is used to trigger an activation function to send it to the next layer units or not.

There are two main groups of ANN, namely those in which the learning process occurs on a straight direction only, called feed-forward ANN, like the simple Multi-Layer Perceptron (MLP), and those that learns via a backpropagation process, like Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN). The backpropagation process introduced a new generation of efficient ANN, since after the feedforward movement, the calculated error goes back to layers in order to recalibrate their weights and biases. This method was popularized by Rumelhart, Hinton & Williams (1986).

Regarding RNN, we do not go into details, since they are better applicable to handwritting and speech recognition tasks. As for CNN, they introduced convolutional and pooling layers to the MLP structure that work as filters to identify relevant and distinct features on data. Viewing data as a matrix, these layers work as a sliding kernel over the matrix, doing sequential transformations to small areas of the matrix. The CNN is widely applicable on image and spacial grid analysis, showing high efficiency when identifying different features, like objects and living beings on images. To keep it simple, the convolutional layer produces a new matrix called feature map, which has a high dimension and then the pooling layer is used to reduce its dimensionality in the same style the previous transformations occur, that is, sliding a kernel over a matrix. The origins of CNN goes back to Lecun et al. (1998), who used the term for the first time.

As for the area with most advances regarding ANN, deep learning, its structures are often referred as a bunch of connected ANN that works like a brain. LeCun, Bengio & Hinton (2015) refer to deep learning as a branch of unsupervised learning. According to them, it discovers intricate structure in large data sets through backpropagation to change the even layers structure, that is, calibrating hyperparameters of the neural structure, which are often said to be fixed parameters chosen by the researcher. For Mosavi et al. (2020), deep-learning is associated with learning various weights at the same time for the same data, being better suitable for investigating market trends than traditional ML. Finally, Srivastava et al. (2014) say deep learning can even help on preventing a common issue, over-fitting, which decrease the good generalization of learning algorithms.

3.2.3 On The Models

In regard to the models we evaluate, we focus on two bounded rationality models, namely the aspiration-based choice from Guney, Richter & Tsur (2018) and the aspiration-based reference dependence model developed in Silva & Riella (2020). Both models assume that sometimes individual perceives an option that maximizes her utility but is not available for choice. Those alternatives, however have some effect on the choice procedure, inducing a reference point which affects the actual choice, for example.

According to Guney, Richter & Tsur (2018), the agent first identifies the aspiration over the whole set of observable alternatives and then it induces the agent to choose the closest feasible option to it, according to an endogenous and psychological metric. The main difference present in Silva & Riella (2020) is a more flexible result, since their conclusion states the existence of an attraction region, such that the closest option to the aspiration may not be chosen. Instead of that, it works as a reference point that draws individual attention to some subset of feasible options including the reference itself. Do not imposing the reference point to be the choice, the model amends a problem in which for any aspiration, one can find a choice problem in which the agent chooses a strictly dominated option.

We select these two models to test the methodological approach for some reasons: *i)* aspiration effect is a class of the so called and well-documented phantom effect in choice procedure (Highhouse (1996), Pettibone & Wedell (2000) and Pettibone & Wedell (2007)), which drops the axiom of independence of irrelevant alternatives or archimedian property; *ii)* desires are inherent to humans and an interesting behavior to model, since any person has some unavailable desired options that they dream about and define their present choices in order to reach them; *iii)* empirical tests regarding aspirations are quite scarce, maybe due to the difficulty to collect data that discriminates observable from feasible options; and *iv)* we provide a method to investigate if the flexibility brought by the attraction region of the identified reference point is actually relevant, since Fudenberg, Gao & Liang (2020) methodology allows one to identify the relevance of parameters on a model, such that important parameters tends to decrease restrictiveness of the model just a little, since adding parameters make models less restrictive and make models able to comprise more outcomes, but showing a significant gain of completeness.

To formally address the models, consider a set X of alternatives. The set of all non-empty subsets of X is denoted by $\mathcal{P}(X)$. A choice problem is defined as a pair (S, T), $S, T \in \mathcal{P}(X)$, $S \subseteq T$. The interpretation is that when the agent faces this choice problem, she observes all the alternatives in T, but only the alternatives in S are available for choice. The set of all choice problems is denoted by $\mathcal{C}(X)$. A choice correspondence is $c : \mathcal{C}(X) \to \mathcal{P}(X)$ such that for each $(S, T) \in \mathcal{C}(X)$, $c(S, T) \subseteq S$.

Under the structure defined by their axioms, one of the choice procedures in Silva & Riella (2020) is described by:

$$c(S,T) = \bigcup_{a \in a(T)} \arg\max u\left(S \cap Q(r(S,a),a)\right), \text{ for any } (S,T) \in \mathcal{C}(X) \tag{3.4}$$

where $u \in \mathbb{R}^X$ is a function, $Q : X \times X \rightrightarrows X$ is a correspondence, $d \in \mathbb{R}^{X^2}$ is a metric, $a(T) = \arg\max_{x \in T} u(x)$ is the set of aspirations and $r(S,a) = \arg\min_{x \in S} d(x,a)$ is the reference induced by each aspiration. It is important to highlight that their conclusions comprise more than one aspiration for a same choice problem. As for Guney, Richter & Tsur (2018), the representation is simpler:

$$c(S,T) = r(S,a) \text{ for any } (S,T) \in \mathcal{C}(X) \tag{3.5}$$

Note that they identify only one aspiration for each choice problem.

3.3 Assessment Framework

In this section we explain the minimum necessary to understand how we get our main results, giving further details on our tests and the data. We first cover the model parametrization we evaluate and then how the evaluation structure can be translated into the selected framework. In the end of the day, the goal is to calculate restrictiveness and completeness of some models and decide which yardstick option uncover most of the behavioral structure.

Regarding the aspiration models, once both of them provide the existence of a psychological metric d to evaluate similarity between options, it becomes convenient because we set from now on that all the distances calculations are made according to the usual euclidean distance, unless we explicitly highlight that a different metric is being used.

About the structure of the set of alternatives X, we consider it to have four alternatives arranged on the real line $\mathbb{R}$. To give a further intuition, one can think the individual must choose among four vehicles, namely an electric car, a hybrid car, a regular car or a motorcycle. They are ordered like that, from highest price $p \in \mathbb{R}^X$ to lowest. She has an endownment $w > p(x), \forall x$. When all options are feasible, ordering the alternatives according to their satisfaction levels results in the same order. Moreover, the distances between the alternatives are set to be 1, according to Fig. (3.1).

However, when the best perceived option is not available, the agent choose the most distant available option, in order to save her money and being likely to get the aspiration on a next opportunity. That is formally written on Eq. (3.6). This way, the psychological difference the agent perceives between the electric car and the hybrid car is the same she perceives between the regular car and the motorcycle and twice the difference between the electic car and the regular car for example. In short, if $S = T$ there is a crescent disutility on the change,

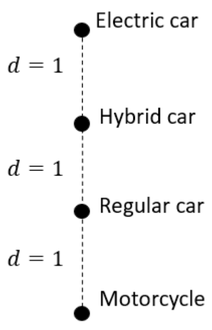


Figure 3.1: Arrangement of alternatives on the real line.

but a crescent utility on the change if $S \neq T$.

Another reason behind such a parametrization would be that when aspiration is not feasible, the individual becomes happier choosing something without a close resemblance to it in order to forget she cannot have her first-best option.

$$u(x) = \begin{cases} -[m - p(x)], \text{if } S = T; \\ -\left[m\left(1 - \min_{a \in a(T)} d(x,a)\right) - p(x)\right], S \neq T \end{cases} \tag{3.6}$$

We must also assume some form for the attraction region Q from representation of Silva & Riella (2020). We set that the reference point draws individual attention to the options that are different from it the same way it differs from the aspiration or less, like Eq. (3.7). For example, if the aspiration is the hybrid car but agent must choose between a regular car and a motorcycle, she first identifies the reference point, which turns out to be the regular car. Thus, the regular attracts itself and the motorcycle, because their distance from the reference is equal or less than the distance between the hybrid car and the regular car.

$$Q(r,a) := \{x \in X : d(x,r) \leq d(a,r)\} \tag{3.7}$$

Note that under this framework, the rational model, when $c(S,T) = c(S,S)$, showing that the individual ignores any unavailable options when choosing, coincides with Guney, Richter & Tsur (2018) model. Due to this reason, further we explore a more flexible framework in order to assess the bounded rationality models as well as the rational model.

Our set of permissive mappings $\mathcal{F}_{\mathcal{M}}$ are restricted to $c(S,T) \subseteq S \subseteq T$. It is worth mentioning that the randomly generated mappings f_m cannot predict an empty choice for any choice problems. The randomly generated mappings f_m are complete mappings, in the sense they impose both reflexivity and complete axioms on preferences. Furthermore, we set $M = 100$, that is, our simulations are made with 100 random mappings, following what was originally made in Fudenberg, Gao & Liang (2020). This way, to simplify the computational work, without any loss, on our conducted tests $\mathcal{F}_\Theta$ is restricted to a single parametrization of the model and a single naive mapping, in order to assess the model in terms of that parametrization, in comparison to the yardstick mapping of choice.

3.3.1 ANN Settings

In regards to our computational work, it was made using *Python* version 3.7.6. The ML distribution we chose is Tensor Flow version 1.14.0, from Google, which uses Graphics Processing Unit (GPU) to achieve better results and faster. The simulations were made on a device equipped with Windows 10, 64-bit, RAM memory of 16GB and GPU NVIDIA GeForce 930M with dedicated memory of 2GB.

Regarding our tested data sets, two aspects must be clear, namely that they are randomly generated and the following terminology we adopt: we refer to complete data as the sample of data generated according to a complete preorder, that is, we may observe one or more choice from the feasible set but never $c(S,T) = \emptyset$; incomplete data as the sample which allows indecisiveness between different feasible alternatives, thus allowing $c(S,T) = \emptyset$; and ordinary data as the data sampled from ordinary preorders, which could even not be reflexive, allowing $c(S,T) = \emptyset$ even if S is a singleton. Moreover, the randomly generated data does not allow one to choose an unavailable option. The codes containing the main classes used to produce our results are shown on Appendix.

On the core part of our investigation, lies the choice of the yardstick mapping. To better understand how to set the basis to normalize models evaluation, instead of simply choosing a fixed naive option, like the uniform-distribution mapping, like Fudenberg, Gao & Liang (2020), Fudenberg & Liang (2020), we tested four possibilities for it. The tested mappings for that purpose are:

- The nothing yardstick mapping, denoted by f_{nothing}, which maps a choice problem (S,T) to $\emptyset$ for any (S,T);

- The all yardstick mapping, denoted by f_{all}, which maps a choice problem (S, T) to S for any (S, T);
- The net yardstick mappings, denoted by f_{MLP} and f_{CNN}, which correspond to the mappings generated from a MLP and a CNN respectively. They learned from a balanced training set of complete, incomplete and ordinary data, with 50 observations each. Since ANN outputs are probabilities between the categories, to set it to a deterministic basis we run these probabilities once to get if options are chosen or not. Finally, they have 16 neurons per layer and 2 layers each.

All the MLP on our simulations, may it be the MLP that serves as yardstick or the one set to do predictions, have the same structure with 484 parameters and all of them are trainable. As for the CNN, they have 980 parameters, all of trainable as well.

About the way we run probabilities for stochastic models, we provide the following explanation. Only probabilities for feasible alternatives are run. For instance, for a choice problem $(\{x, y\}, \{x, y, z\})$, such that an ANN predicts choice probabilities of $(0.95, 0.70, 0.05)$, we roll a dice with 95% of chance of getting x chosen, then we roll a dice with 70% of chance of getting y chosen, but we do not run a dice with 5% of chance of getting z chosen, we simply set 0 for z. We do that in order to impose a model to not predict an unavailable choice, to keep the deterministic transformation of ANN in $\mathcal{F}_{\mathcal{M}}$. To better study how output probabilities for non-available option impacts the assessment of stochastic models, we change the euclidean loss function to binary crossentropy on a further next section.

It is important to clear out that ANN are trained with limited data sets, such that the data used to train them does not feature all the possible choice problems. This way, for some choice problems, they output choice probabilities from the patterns they learn from other choice problems.

When it comes to the calculation of the restrictiveness, we take into account 100 random mappings, following what is did by Fudenberg, Gao & Liang (2020). They are permissive maps, that is, cannot predict the choice of unavailable options, and their output is non-empty. To compute completeness we need to find the best mapping, f_{best}. This map is the closest one to the true mapping f^*, that is, for a given sample of data, it minimizes expected value of the loss function. Since we are dealing with ANN, in order to find f_{best}, we set it to be the

MLP with 32 neurons per layer and two hidden layers that is trained with the analyzed data, over fitting it. Furthermore, we set the length of evaluation data sets to 5000 observations.

We set our loss function to be the usual euclidean distance. This way,

$$d(f, f') = E_{P_x}[(d(f(X), f'(X))] \tag{3.8}$$

Finally, when calculating completeness, to get the out of sample error, when doing the k-fold crossvalidation we set $k = 10$. Regarding the ANN we set for predictions to compare their assessment with our selected bounded rationality models, MLP and CNN, they were trained from a balanced sample of complete, incomplete and ordinary data, each one with 1000 observations. Both have 16 neurons per hidden layer. We also train with the same data a MLP and a CNN without the ordinary data, which may be seen as a noise, since it does not even impose reflexibility. This way we also investigate how relevant this axiom is for empirical implications. For this set we consider 1000 observations of complete data and 1000 observations of incomplete data.

The data used to train those ANN are different than that used to train the yardstick ANN, f_{MLP} and f_{CNN}. The validation itself occurs on the completeness evaluation with data sets with 5000 observations. On this matter, we analyzed two neural network setups. Also, since the benchmark for ML models is often the logit model, we evaluate it and compare its results as well. We reinforce that the ANN that we set for predictions, like those set as yardstick, also learn from limited data sets.

We chose to set the size of the training batch of data large, having over than thousand observations, in order to simulate the best scenario, in which the researcher has at hand a lot of observations and a balanced of observations of each type.

The MLP we trained, both the MLP to be evaluated and the MLP set as a yardstick, have 16 neurons, two hidden layers; their loss functions is set to binary crossentropy; and their optimizer function is set to adam algorithm. As for both CNN, they were built with two convolutional layers and a single pooling layer, which uses the max pooling method.

The last layer on each neural network are set to use *sigmoid* activation function instead of *softmax*. The reason behind it is that although *softmax* function is a more generalized logistic function often used on multiclass classification problems, forcing the output probabilities for a given input to sum up to 1, the sigmoid fits better our framework since the choices are not mutually exclusive. That being said, softmax activation seems better suitable for choice function modeling and not choice correspondences, since choice functions' image is singleton.

It is worth mentioning that in spite of our framework being defined as a multi-label classification problem, whose target outputs are vectors of 0 and 1, indicating if options are chosen or not, we do not use binary crossentropy or categorical crossentropy loss functions, because our predictions are not probabilities. If we use these metrics on a deterministic setup, we may end with some cases in which the distance equals $\log 0$. Thus, binary crossentropy or categorical crossentopy are suitable to calculate distance between a mapping generated from a neural network and another map, since its predictions are probabilities. We do so on a separated section.

This way, for the deterministic setup, we set the loss function to be the euclidean distance, since the psychological metric of the models being evaluated are chosen to be the same metric. However, instead of euclidean loss function, for example, one may also use mean squared error as well.

3.4 Data

Since our setting comprises 4 options, data is a vector $o \in \mathbb{R}^{12}$, such that $o = (c(S,T), S, T)$ and $c(S,T), S, T \in \mathbb{R}^4$. Each coordinate of o receive 1 to indicate the presence of an option on the $c(S,T)$, S or T and 0 indicates the absence of it. A comprehensive visualization of the data structure is presented on Fig. (3.2).

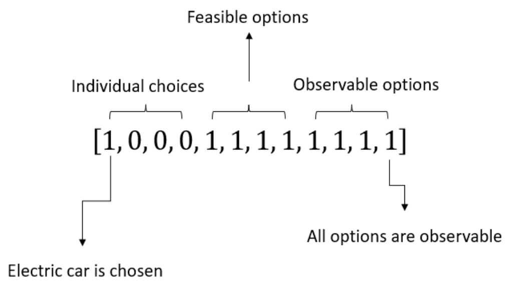


Figure 3.2: Comprehensive visualization of each data observation o.

Now we provide a better image of what a prediction mapping $f \in \mathcal{F}$ looks like on our setup. It is basically a function which maps all choice problems $(S,T) \in \mathbb{R}^8$ into $c(S,T) \in \mathbb{R}^4$.

Under our restricted setting, we have roughly 65 choice problems. This way, we can see models parametrizations and neural networks like prediction mappings as well, but with a known structure behind it.

Note that meaningful choice problems are not in number of 256 (2^8), like it is for $\mathcal{F}$. We take into account only relevant problems, discarding settings in which there are no feasible or observable options. Additionally, whenever an option is not observed, it cannot be feasible as well. That restricts choice problems of $\mathcal{F}_{\mathcal{M}}$ to 65. Check Fig. (3.3) for a visual representation.

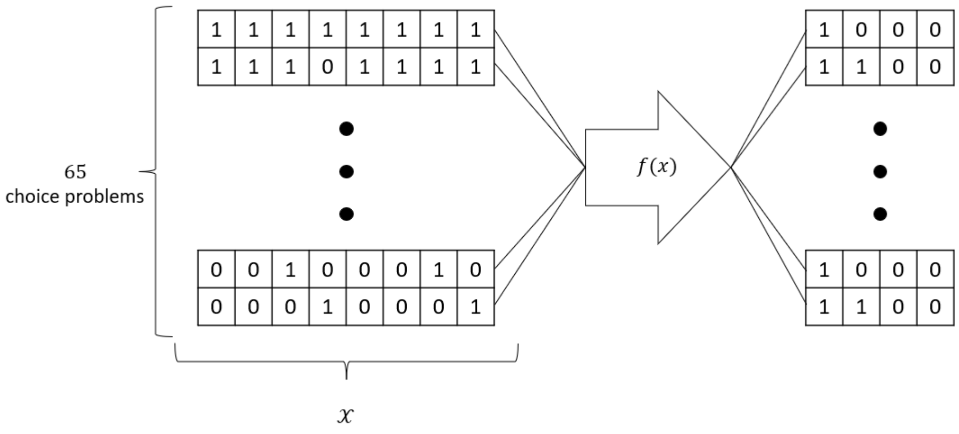


Figure 3.3: Visual representation of a mapping f structure.

Finally, the data we used for our computational simulations are randomly generated. Finding revealed preferences data itself is not a trivial task. Furthermore, such data with access to the options the agent observe, being aware of their existence, but cannot choose them becomes even more difficult. We do not go into a survey because our main contribution here is methodological, that is, we propose ANN as a feasible option to improve the ability of discovering structure on data sets, so there is no loss in using random data. Moreover, there is no loss on using random data to analyze the empirical implications of imposing some decision axioms, as a part of data preparation and cleaning.

3.5 Results

3.5.1 Deterministic Experiment

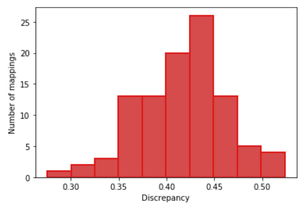
Being straightforward, regarding the ANN and logit outputs we first find that for a given $c(S, T)$ with $S \subset T$, that is, a choice problem with observable options that are not available, CNN spits out extreme low probabilities of choice for any $x \in T$ but $x \notin S$, in the order of 10^{-5} to 10^{-7}. However, MLP probabilities for those alternatives are in the order of 10^{-3} to 10^{-4}. The logit probabilities for these alternatives are relatively lower, in the order of 10^{-2}. It shows that CNN are better on detecting unavailable alternatives.This finding itself can be explored on future research topics, since we did not found any documentation of this feature on previous works. However, since we do not allow an unfeasible option to be chosen when running probabilities, this fact cannot be captured by the deterministic evaluation of models.

Although completeness lies in the unit interval, its estimator (3.3) proposed by Fudenberg, Gao & Liang (2020), which is based on the out-of-sample error CV may be negative in some cases. Conducting some analysis, we found out that that happens due to overfitting or when the test set is not a good replica of the training set. On such cases, it turns out that the minimal error mapping on the training set shows a higher error on the test set. Precisely on our case, since we are working with synthetic data, we found out some negative values for completeness. However, on our simulations, all the negative values are not statistically significant according to the statistics and standard errors proposed in Fudenberg, Gao & Liang (2020), even accounting for a level of confidence of 5%.

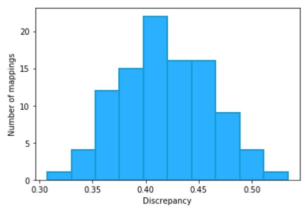
Finally, in the next series of tables, the standard errors are displayed in parenthesis.

	Restrictiveness	Completeness (Complete data)	Completeness (Incomplete data)	Completeness (Ordinary data)
Silva & Riella (2020)	0.417 (0.005)	**0.871** (0.057)	**0.835** (0.055)	-0.072*[1] (0.103)
Guney, Richter & Tsur (2018)	0.418 (0.004)	**0.856** (0.057)	**0.832** (0.055)	-0.023* (0.103)
MLP	0.496 (0.004)	0.744 (0.057)	0.718 (0.061)	-0.069* (0.103)
$MLP_{\text{reflexive}}$	0.418 (0.004)	**0.860** (0.052)	**0.841** (0.056)	-0.018* (0.103)
CNN	**0.678** (0.004)	0.474 (0.063)	0.420 (0.067)	-0.065* (0.102)
$CNN_{\text{reflexive}}$	0.531 (0.004)	0.711 (0.059)	0.684 (0.062)	-0.093* (0.103)
Logit	0.471 (0.004)	0.791 (0.056)	0.714 (0.060)	0.123* (0.103)

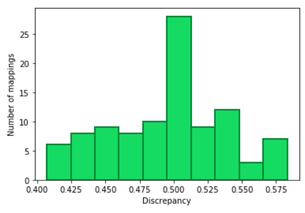
Table 3.1: Deterministic models assessment setting the yardstick map to f_{nothing}. Standard errors are displayed in parenthesis.



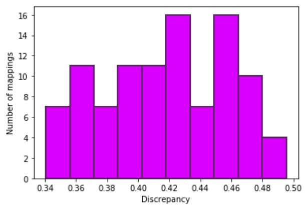
(a) Silva & Riella (2020)



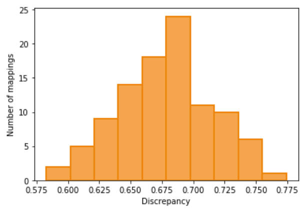
(b) Guney, Richter & Tsur (2018)



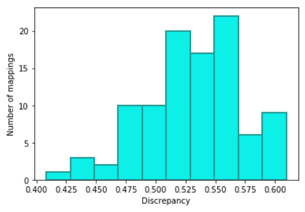
(c) MLP



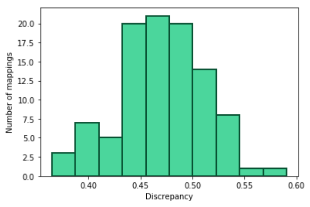
(d) $MLP_{reflexive}$



(e) CNN



(f) $CNN_{reflexive}$



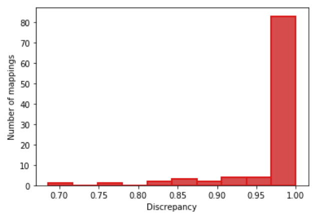
(g) Logit

Figure 3.4: Distribution of the δ-discrepancy of 100 random mappings, regarding the yardstick map f_{nothing}.

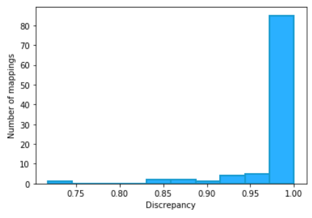
[1]The symbol * indicates that the values are not statistically significant for relevant values of significance of the test. The negative values of completeness occurs due to the fact that the validation group is not a good replica of the training set on the k-fold crossvalidation procedure. The problem may also happen due to overfitting, which might require some treatment of the data.

	Restrictiveness	**Completeness (Complete data)**	**Completeness (Incomplete data)**	**Completeness (Ordinary data)**
Silva & Riella (2020)	0.979 (0.005)	0.000 (0.099)	-0.103* (0.089)	0.000 (0.107)
Guney, Richter & Tsur (2018)	0.985 (0.004)	0.000 (0.099)	-0.105* (0.089)	0.000 (0.107)
MLP	0.999 (0.001)	0.000 (0.099)	0.000 (0.090)	0.000 (0.107)
$MLP_{\text{reflexive}}$	0.994 (0.002)	0.000 (0.099)	-0.053* (0.088)	0.000 (0.107)
CNN	1.000 (0.000)	0.000 (0.099)	0.000 (0.090)	-0.120* (0.107)
$CNN_{\text{reflexive}}$	1.000 (0.000)	0.000 (0.099)	0.000 (0.090)	0.000 (0.107)
Logit	0.999 (0.001)	0.000 (0.099)	0.000 (0.090)	-0.003* (0.106)

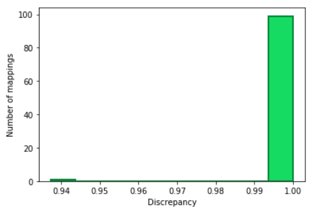
Table 3.2: Deterministic models assessment setting the yardstick map to f_{all}. Standard errors are displayed in parenthesis.



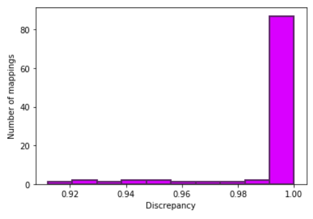
(a) Silva & Riella (2020)



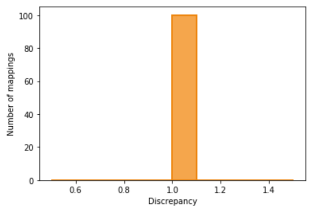
(b) Guney, Richter & Tsur (2018)



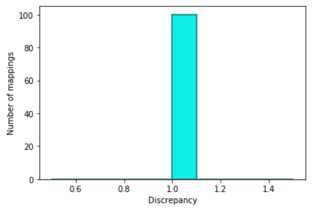
(c) MLP



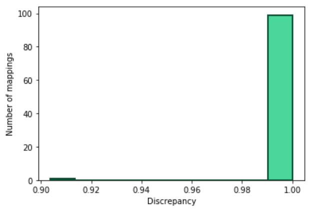
(d) MLP$_{\text{reflexive}}$



(e) CNN



(f) CNN$_{\text{reflexive}}$

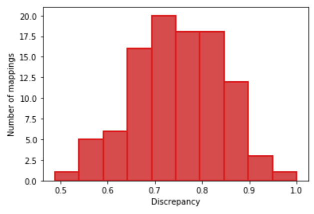


(g) Logit

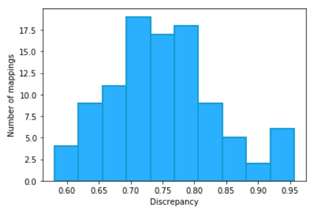
Figure 3.5: Distribution of the δ-discrepancy of 100 random mappings, regarding the yardstick map f_{all}.

	Restrictiveness	Completeness (Complete data)	Completeness (Incomplete data)	Completeness (Ordinary data)
Silva & Riella (2020)	0.749 (0.010)	**0.617** (0.075)	**0.592** (0.070)	-0.090* (0.104)
Guney, Richter & Tsur (2018)	0.751 (0.008)	**0.573** (0.075)	**0.586** (0.069)	-0.076* (0.103)
MLP	**0.888** (0.009)	0.238 (0.075)	0.303 (0.076)	-0.197* (0.104)
$MLP_{\text{reflexive}}$	0.760 (0.009)	0.577 (0.068)	**0.594** (0.071)	0.072* (0.103)
CNN	***0.999*** (0.001)	*0.000* (0.079)	*0.000* (0.080)	-0.064* (0.103)
$CNN_{\text{reflexive}}$	0.939 (0.006)	0.127 (0.077)	0.191 (0.079)	-0.014* (0.103)
Logit	0.844 (0.008)	0.379 (0.073)	0.294 (0.076)	0.099* (0.104)

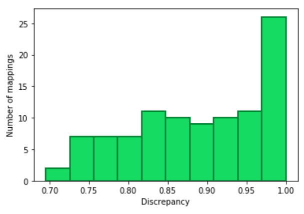
Table 3.3: Deterministic models assessment setting the yardstick map to f_{MLP}. Standard errors are displayed in parenthesis.



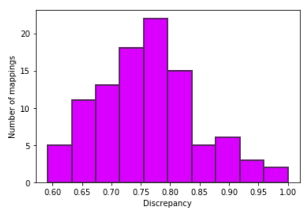
(a) Silva & Riella (2020)



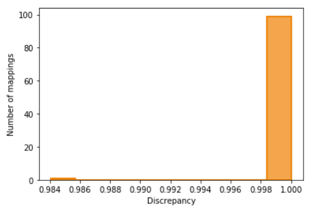
(b) Guney, Richter & Tsur (2018)



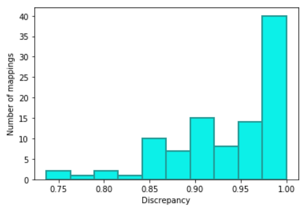
(c) MLP



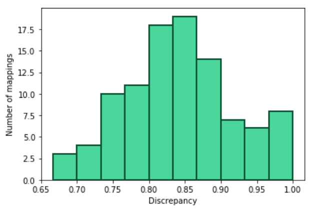
(d) $MLP_{reflexive}$



(e) CNN



(f) $CNN_{reflexive}$

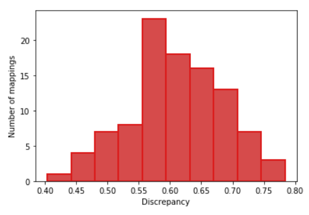


(g) Logit

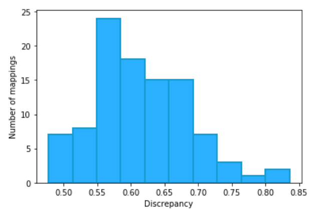
Figure 3.6: Distribution of the δ-discrepancy of 100 random mappings, regarding the yardstick map f_{MLP}.

	Restrictiveness	**Completeness (Complete data)**	**Completeness (Incomplete data)**	**Completeness (Ordinary data)**
Silva & Riella (2020)	0.610 (0.007)	**0.757** (0.067)	**0.713** (0.064)	-0.058* (0.105)
Guney, Richter & Tsur (2018)	0.613 (0.007)	**0.729** (0.067)	**0.709** (0.063)	-0.067* (0.104)
MLP	**0.728** (0.008)	0.516 (0.067)	0.510 (0.070)	-0.103* (0.104)
$MLP_{\text{reflexive}}$	0.566 (0.007)	**0.772** (0.058)	**0.765** (0.062)	-0.139* (0.104)
CNN	***0.965*** (0.004)	-0.019* (0.074)	-0.045* (0.076)	-0.168* (0.104)
$CNN_{\text{reflexive}}$	0.718 (0.006)	0.530 (0.066)	0.531 (0.069)	0.000 (0.104)
Logit	0.689 (0.007)	0.605 (0.065)	0.503 (0.069)	-0.018* (0.105)

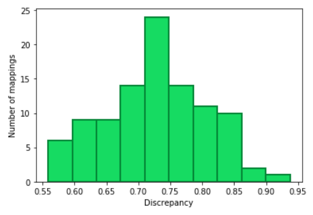
Table 3.4: Deterministic models assessment setting the yardstick map to f_{CNN}. Standard errors are displayed in parenthesis.



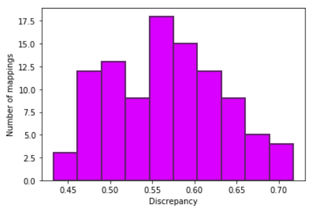
(a) Silva & Riella (2020)



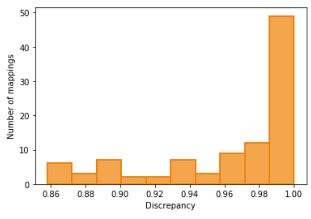
(b) Guney, Richter & Tsur (2018)



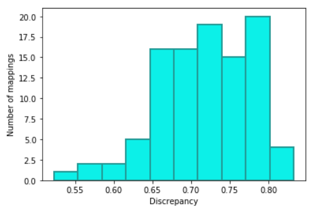
(c) MLP



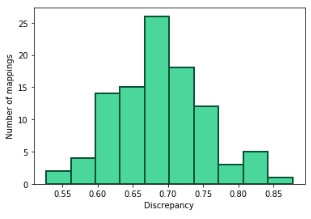
(d) $MLP_{\text{reflexive}}$



(e) CNN



(f) CNN



(g) Logit

Figure 3.7: Distribution of the δ-discrepancy of 100 random mappings, regarding the yardstick map f_{CNN}.

Given the results presented on the Tables and Figures above, we summarize the main findings below:

- Setting the yardstick to f_{nothing} does not help to reveal much structure on data, since all models are evaluated as flexible models with high completeness. That is expected, since flexible models fit almost all conceivable data without capturing structure specific to the observed data;
- ANN seems like the best yardstick option to reveal data structure. On this matter, MLP yardstick appears as the option that shows the best pair restrictiveness and completeness, with slightly restrictive models but a meaningful completeness level;
- Comparing theoretical models with ANN models when setting the yardstick to an ANN, economic models reveal more structure, with a higher completeness evaluation. Although economic models' restrictiveness are a bit lower than ANN ones, their completeness with both complete and incomplete data are considerably higher;
- On the comparison between aspirational models in Guney, Richter & Tsur (2018) and in Silva & Riella (2020), the findings indicate that considering the existence of a attraction region of the reference point, for the synthetic data we used, is in fact a good way to approach data, since it leads to a small loss of restrictiveness and a higher gain of completeness. This way, if such a finding repeats for data collected from real world, it shows that the effect of the attraction region plays an important role in the decision making under phantom and decoy effects;
- Economic models overall are better tools to identify structure on data. This result shows that evaluating economic models with the use of ML foster that economic models serve as a meaning compass to behavioral researches;
- CNN shows for almost all experiments the higher restrictiveness level. This indicates the convolutional learning process is able to capture some particular behavioral patterns that cannot accommodate many modeling strategies. However, those identified regularities are not present on the batch of data we analyzed, due to the low completeness evaluations;

- Considering only the ANN and *logit* models, MLP had way better results than CNN and even the usual benchmark, that is the *logit* model, with considerably high completeness without being loose models able to comprise any data set;
- Imposing complete preferences on data does not seem to impair the evaluation of models. Although the assessment of models with complete data is different from that with incomplete data, they are pretty similar, without any significant change. This finding indicates that removing indecisiveness from choice data may not be a part of data preparation when dealing with choice data, fostering that indecisiveness is a core part of individual's behavior;
- Regarding the choice data set the researcher intends to investigate, our results show that data preparation is a fundamental part of the data analysis, even dealing with choice data. Looking at all the completeness evaluations for ordinary data, which can be seen as a raw data with many noises, like a sparse matrix, their results are not different from zero from a statistical view and hence not meaningful. This result also shows the relevance of imposing reflexivity on choice theory, even for empirical analysis. Reflexive data set used to train ANN shows a gain of structure for both MLP and CNN. All results, aside from those setting yardstick to f_{all}, shows that reflexive train data sets make ANN a bit more flexible and then improve their completeness on data, may it be complete or incomplete.

3.5.2 Stochastic Experiment

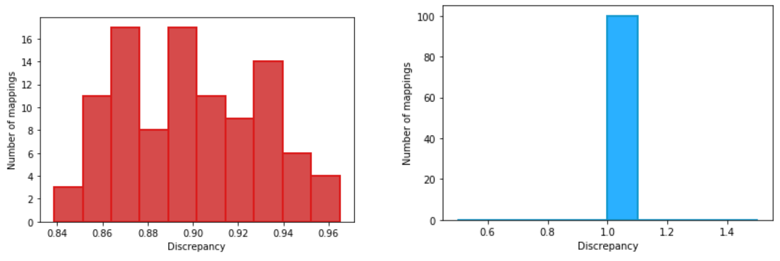
Now we analyze the trained neural networks on a stochastic choice domain, keeping their output probabilities. Doing so, we can investigate if there is any significant difference on the assessment of both MLP and the CNN, specially because their probabilities for unavailable options slightly differ. Here we change the loss function to a more adequate option for this analysis, that is, Binary Crossentropy (BCE). We still fix $M = 100$ and the yardstick map is set to the *logit* map, which is the usual benchmark for stochastic models. The random mappings f_m are still kept as deterministic mappings.

	Restrictiveness	**Completeness (Complete data)**	**Completeness (Incomplete data)**	**Completeness (Ordinary data)**
MLP	**0.899** (0.003)	**0.429** (0.042)	**0.494** (0.041)	0.147 (0.069)
CNN	*1.000* (0.000)	*0.000* (0.047)	*0.000* (0.049)	*0.000* (0.065)

Table 3.5: Assessment of ANN trained with a balanced set of complete, incomplete and ordinary data setting the yardstick map to f_{logit}. Standard errors are displayed in parenthesis.

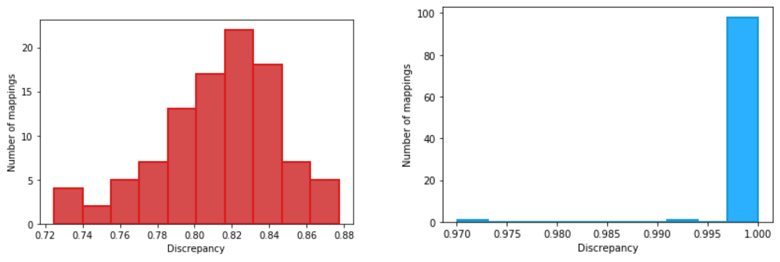
	Restrictiveness	**Completeness (Complete data)**	**Completeness (Incomplete data)**	**Completeness (Ordinary data)**
MLP	**0.814** (0.003)	**0.861** (0.044)	**0.833** (0.044)	0.000 (0.068)
CNN	*0.999* (0.000)	*0.000* (0.054)	*0.068* (0.057)	0.000 (0.068)

Table 3.6: Assessment of ANN trained with a balanced set of complete and incomplete data setting the yardstick map to f_{logit}. Standard errors are displayed in parenthesis.



(a) MLP *(b) CNN*

Figure 3.8: Distribution of the δ-discrepancy of 100 random mappings, regarding the yardstick map f_{logit}, considering the training set a balanced set with complete, incomplete and ordinary data.

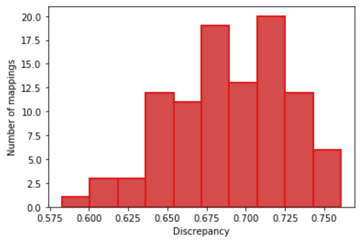


(a) MLP *(b) CNN*

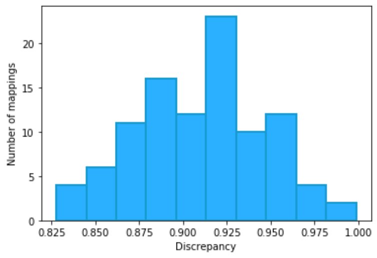
Figure 3.9: Distribution of the δ-discrepancy of 100 random mappings, regarding the yardstick map f_{logit}, considering the training set a balanced set with complete and incomplete data.

	Restrictiveness	**Completeness (Complete data)**	**Completeness (Incomplete data)**	**Completeness (Ordinary data)**
MLP	0.689 (0.004)	**0.893** (0.038)	**0.691** (0.051)	0.000 (0.061)
CNN	**0.911** (0.004)	0.244 (0.053)	0.320 (0.049)	0.030 (0.066)

Table 3.7: Assessment of ANN trained with a balanced set of complete data setting the yardstick map to f_{logit}. Standard errors are displayed in parenthesis.



(a) MLP



(b) CNN

Figure 3.10: Distribution of the δ-discrepancy of 100 random mappings, regarding the yardstick map f_{logit}, considering the training set a balanced set with complete data.

According to the results presented on Tables and Figures below, the main conclusions are the following:

- Even on the stochastic environment, imposing reflexivity on choice data still remains as a crucial part of data preparation in order to get better and meaningful results;
- Training ANN with data after imposing reflexivity still remains a good strategy to better evaluate ANN, specially in the case of MLP which showed a significant gain of completeness followed by a timid drop on its restrictiveness;
- Comparing Tables 3.6 and 3.7, we conclude that imposing the axioms of complete preferences when training ANN does not show any gains in terms of the networks performance, since the MLP trained with both complete and incomplete data shows higher restrictiveness and completeness levels than the one that is trained only with complete data;
- MLP appears to have a higher degree of compatibility with decision theory analysis, in which data use a binary representation for choices.

3.5.3 A More Flexible Framework

Here, we build a wider experimental setup for deterministic choice problems, basically allowing more options in order to investigate how much of previous results are related to the restrictive setup with only four alternatives. Doing so, we are able to compare the rational model with the covered bounded rationality models. The model we refer as rational under this framework is the standard utility maximizer in which individual does not take into account any $x \in T \setminus S$ when choosing from S, simply picking $\arg\max_{x\in S} u(x)$. The options the agent can choose are shown in Fig. (3.11).

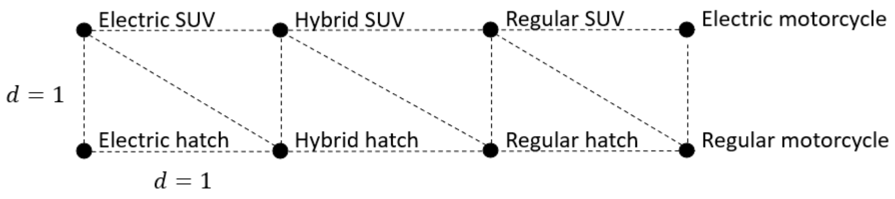


Figure 3.11: Spatial arrangement of alternatives the individual can choose on a framework with 8 options.

We keep all parametrizations set before. When the agent observes all options and can choose any of them, we set the satisfaction ordering (with decreasing prices) as being electric SUV $\succ$ electric hatch $\succ$ hybrid SUV $\succ$ hybrid hatch $\succ$ regular SUV $\succ$ regular hatch $\succ$ electric motorcycle $\succ$ regular motorcycle. To get a glimpse on models predictions on this framework, suppose the agent observes all options and the only unavailable option is the electric car, which turns out to be her aspiration. In this case, the rational model says she chooses the electric hatch and aspiration-only model shows indifference between the references, hybrid SUV and electric hatch. As for the aspiration-reference model, after identifying the two references, her attention lies on them and on the hybrid hatch and regular SUV, leading to a regular SUV choice, since it is cheaper. Furthermore, note that under this framework, the only model that allows indifference is the aspiration-only model.

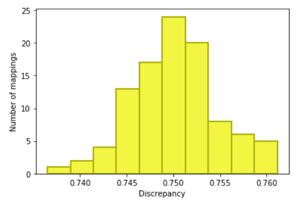
Regarding the structure of the yardsticks ANN, we had find 16 neurons per hidden layer appears to be a better setting for uncovering data structure, instead of 32 neurons per layer like set in the four alternatives case. It speaks to the fact that under a setting with more alternatives, dropping the amount of neurons reduce the amount of parameters of the ANN, which reduces over-fitting effects and improve the generalization capabilities of the ANN.

About the number of parameters of the ANN used on this experiment, we set all of them to be trainable, with MLP containing 680 parameters and CNN containing 1624 parameters. The findings here are pretty much consistent with those previously found. However, since we are analyzing completeness on a random data set, allowing individuals to choose more options significantly drop models' completeness evaluation.

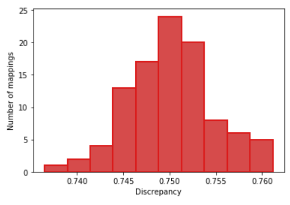
Although not much different from each other, completeness levels for the random data sets showed a slightly better degree of fitness of the aspiration-only model in relation with the other models.

	Restrictiveness	**Completeness (Complete data)**	**Completeness (Incomplete data)**	**Completeness (Ordinary data)**
Rational	0.750 (0.000)	**0.565** (0.069)	0.443 (0.072)	0.022* (0.081)
Silva & Riella (2020)	0.750 (0.000)	0.546 (0.069)	0.448 (0.072)	0.043* (0.082)
Guney, Richter & Tsur (2018)	0.748 (0.000)	**0.567** (0.069)	**0.456** (0.072)	0.038* (0.081)
MLP	0.778 (0.000)	0.512 (0.066)	0.030 (0.073)	0.073* (0.082)
$MLP_{\text{reflexive}}$	0.778 (0.001)	0.512 (0.066)	0.315 (0.073)	-0.002* (0.081)
CNN	**0.814** (0.000)	0.419 (0.066)	0.242 (0.073)	-0.018* (0.079)
$CNN_{\text{reflexive}}$	0.805 (0.000)	0.451 (0.066)	0.236 (0.074)	0.022* (0.082)
Logit	0.785 (0.001)	0.477 (0.066)	0.265 (0.073)	0.096* (0.082)

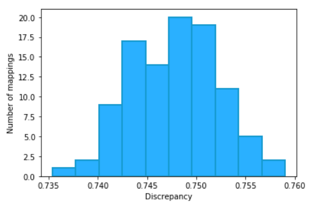
Table 3.8: Deterministic models assessment setting the yardstick map to f_{nothing}. Standard errors are displayed in parenthesis.



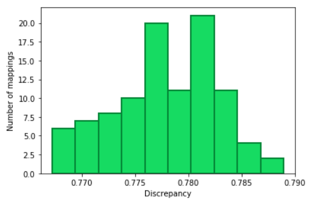
(a) Rational



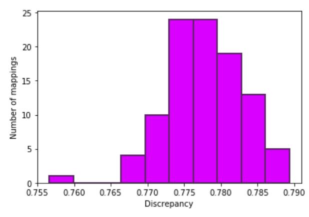
(b) Silva & Riella (2020)



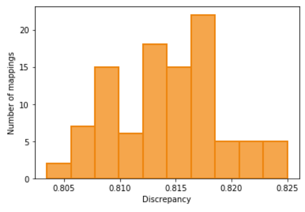
(c) Guney, Richter & Tsur (2018)



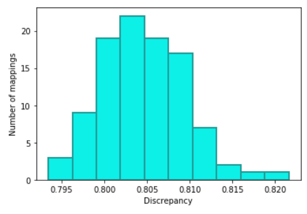
(d) MLP



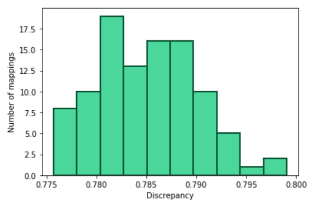
(e) MLP$_{\text{reflexive}}$



(f) CNN



(g) CNN$_{\text{reflexive}}$

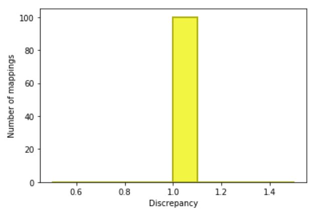


(h) Logit

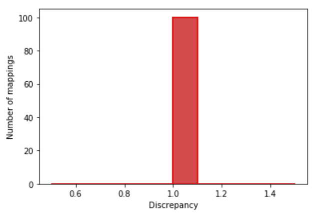
Figure 3.12: Distribution of the δ-discrepancy of 100 random mappings, regarding the yardstick map f_{nothing}.

	Restrictiveness	**Completeness (Complete data)**	**Completeness (Incomplete data)**	**Completeness (Ordinary data)**
Rational	1.000 (0.000)	0.000 (0.078)	0.054* (0.082)	0.000 (0.084)
Aspiration-Reference	1.000 (0.000)	0.000 (0.078)	0.063* (0.082)	-0.041* (0.082)
Aspiration-only	1.000 (0.000)	0.000 (0.078)	0.076* (0.082)	0.000 (0.082)
MLP	1.000 (0.000)	0.000 (0.078)	0.000 (0.082)	0.051* (0.081)
$MLP_{\text{reflexive}}$	1.000 (0.000)	0.000 (0.078)	0.000 (0.082)	0.030* (0.080)
CNN	1.000 (0.000)	0.000 (0.078)	0.000 (0.082)	0.087* (0.082)
$CNN_{\text{reflexive}}$	1.000 (0.000)	0.000 (0.078)	0.000 (0.082)	0.071* (0.081)
Logit	1.000 (0.000)	0.000 (0.078)	0.000 (0.082)	0.042* (0.080)

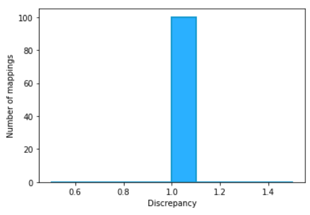
Table 3.9: Deterministic models assessment setting the yardstick map to f_{all}. Standard errors are displayed in parenthesis.



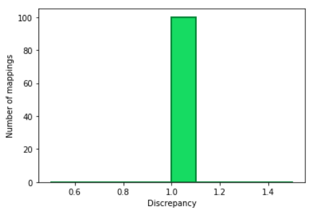
(a) Rational



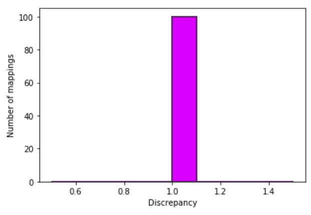
(b) Silva & Riella (2020)



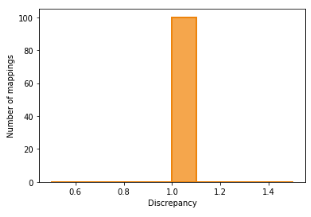
(c) Guney, Richter & Tsur (2018)



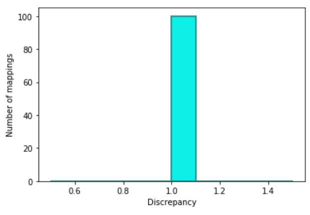
(d) MLP



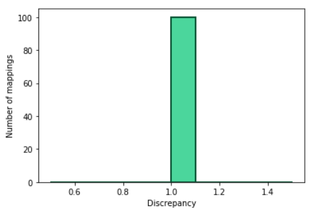
(e) MLP$_{\text{reflexive}}$



(f) CNN



(g) CNN$_{\text{reflexive}}$

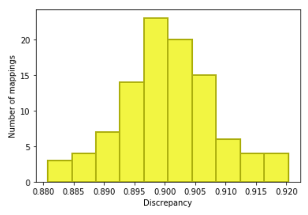


(h) Logit

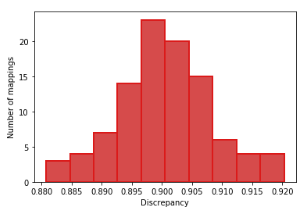
Figure 3.13: Distribution of the δ-discrepancy of 100 random mappings, regarding the yardstick map f_{all}.

	Restrictiveness	**Completeness (Complete data)**	**Completeness (Incomplete data)**	**Completeness (Ordinary data)**
Rational	0.900 (0.001)	**0.312** (0.078)	0.288 (0.076)	-0.020* (0.082)
Silva & Riella (2020)	0.901 (0.001)	0.281 (0.078)	**0.295** (0.076)	-0.023* (0.082)
Guney, Richter & Tsur (2018)	0.898 (0.001)	**0.315** (0.078)	**0.305** (0.077)	-0.035* (0.082)
MLP	0.934 (0.001)	0.228 (0.074)	0.107 (0.078)	0.036* (0.082)
$MLP_{\text{reflexive}}$	0.934 (0.001)	0.228 (0.074)	0.126 (0.077)	-0.042* (0.082)
CNN	**0.978** (0.001)	0.080 (0.074)	0.033 (0.078)	0.000 (0.082)
$CNN_{\text{reflexive}}$	**0.966** (0.001)	0.130 (0.074)	0.024 (0.078)	-0.020* (0.082)
Logit	0.942 (0.001)	0.172 (0.074)	0.061 (0.078)	0.061* (0.083)

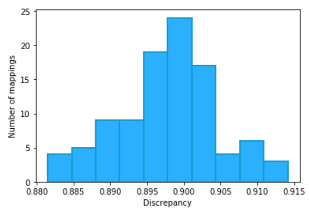
Table 3.10: Deterministic models assessment setting the yardstick map to f_{MLP}. Standard errors are displayed in parenthesis.



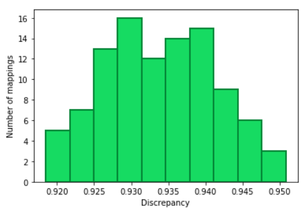
(a) Rational



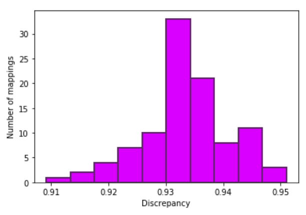
(b) Silva & Riella (2020)



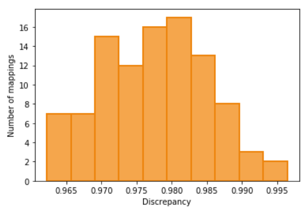
(c) Guney, Richter & Tsur (2018)



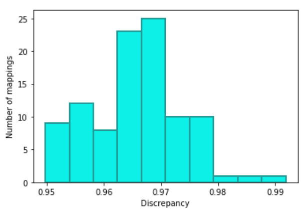
(d) MLP



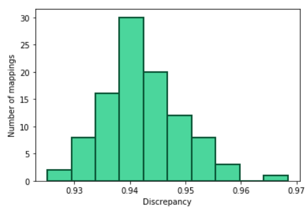
(e) MLP$_{\text{reflexive}}$



(f) CNN



(g) CNN$_{\text{reflexive}}$

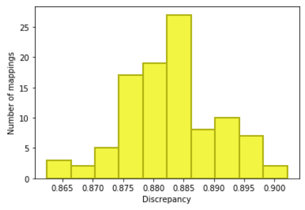


(h) Logit

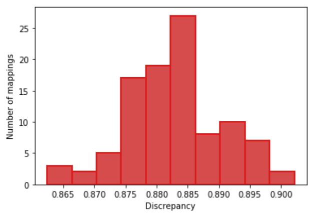
Figure 3.14: Distribution of the δ-discrepancy of 100 random mappings, regarding the yardstick map f_{MLP}.

	Restrictiveness	Completeness (Complete data)	Completeness (Incomplete data)	Completeness (Ordinary data)
Rational	0.882 (0.001)	0.326 (0.078)	0.305 (0.076)	-0.043* (0.081)
Silva & Riella (2020)	0.883 (0.001)	0.296 (0.077)	0.312 (0.076)	0.026* (0.082)
Guney, Richter & Tsur (2018)	0.880 (0.001)	**0.329** (0.077)	**0.321** (0.076)	-0.003* (0.082)
MLP	0.916 (0.001)	0.244 (0.074)	0.128 (0.078)	0.056* (0.082)
$MLP_{\text{reflexive}}$	0.915 (0.001)	0.244 (0.074)	0.146 (0.077)	-0.017* (0.082)
CNN	**0.958** (0.001)	0.100 (0.073)	0.056 (0.077)	0.000 (0.081)
$CNN_{\text{reflexive}}$	0.947 (0.001)	0.148 (0.074)	0.048 (0.078)	-0.012* (0.082)
Logit	0.924 (0.001)	0.189 (0.074)	0.084 (0.077)	0.080* (0.083)

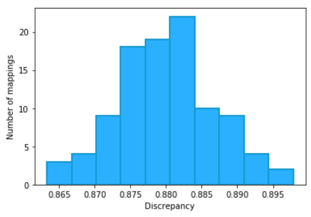
Table 3.11: Deterministic models assessment setting the yardstick map to f_{CNN}. Standard errors are displayed in parenthesis.



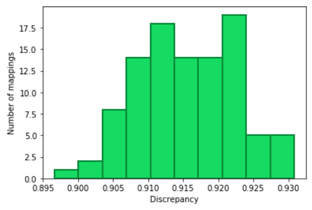
(a) Rational



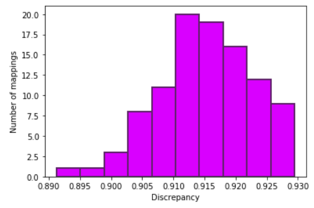
(b) Silva & Riella (2020)



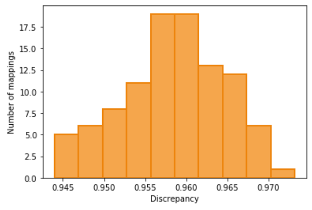
(c) Guney, Richter & Tsur (2018)



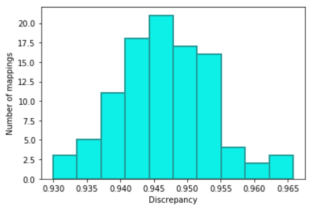
(d) MLP



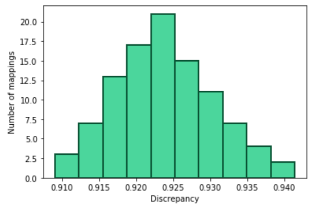
(e) $MLP_{\text{reflexive}}$



(f) CNN



(g) $CNN_{\text{reflexive}}$



(h) Logit

Figure 3.15: Distribution of the δ-discrepancy of 100 random mappings, regarding the yardstick map f_{CNN}.

3.6 Bridging Rationality and Real World Behavior: A Numerical Approach

The main goal of this section is to use the metrics of restrictiveness and completeness to explore the joint potential of a pair of models, one deterministic and another stochastic, to explain real world conjunctures in terms or behavior and uncertainty. In this sense, the deterministic model reveals its compatibility with data behavior and the stochastic models reveals how much of the data uncertainty can be understood under its setup. This way, we explore the advantages of both modeling strategies, namely the clear notion of rationality under deterministic setups, and the flexibility of probabilistic models.

Previously we investigated the relationship between models and data, using ANN to better understand it. Now we investigate the relationship between models, using data. By the end of the day, the goal is the same as before, namely to identify structure on data. However, now we give a step forward. After finding how much structure a deterministic model reveals for some data set, we ask how much of the remaining uncertainty of data can be understood under a stochastic model structure.

We propose a numerical method based on Fudenberg, Gao & Liang (2020) completeness measure instead of an axiomatic approach, like developed in Ok & Tserenjigmid (2019), Ok & Tserenjigmid (2021) and Ok & Tserenjigmid (2022). Although the approaches have a similar purpose, seeking rationality, they slightly differ. While the authors attempt to compare the rationality degree of stochastic models or look for some rational consistency on these models, we attempt to numerically identify if the flexibility allowed by a probabilistic model helps to explain the uncertainty in data that a deterministic model cannot explain.

By doing so, we may be able to get additional structure for the preferences underlying some data set, since the stochastic model provides a more data-friendly way to improve deterministic models comprehension, allowing to understand randomness on choices using a stochastic model grounded on behavioral hypothesis as well.

Our approach is based on the following proposed measure, which is a score based on completeness evaluation of the stochastic model, that we call it Deterministic-Completeness, presented in the Eq. (3.9). For example, Deterministic-Completeness is referred as rational-completeness if for a given stochastic model we evaluate its completeness setting the yardstick to the rational model, or aspirational-completeness, if we set the yardstick to be some aspirational model:

$$\kappa = \frac{MSE_{\text{yardstick}} - MSE_{\text{model}}}{MSE_{\text{yardstick}} - MSE_{\text{best}}} \tag{3.9}$$

Completeness definition is based on errors, that is, the errors of the mappings on some data. As previously said, errors can be calculated in different manners, like euclidean distance, mean squared error (MSE), binary crossentropy, categorical binary crossentropy and so on. So far, we had used euclidean distance to calculate deterministic models' errors and binary crossentropy for stochastic models, but that is to be decided by the researcher according to the nature of her problem. However, for the bridging approach, we propose using MSE since it is meaningful for both deterministic and stochastic models. That being said, the only difference from Eq. (3.9) and the completeness definition is that we specify the function used to evaluate the error, which turned out to be the MSE.

On Eq. (3.9) we set the yardstick as being the deterministic model, the best model to be a MLP over fitting analyzed data as did before and the model being evaluated as a set with the stochastic model and the deterministic model, hence, the yardstick.

Note that for a deterministic model, regardless of its completeness level on a data set, if the stochastic model shows a higher MSE than the deterministic model, its deterministic-completeness is null. It shows that the flexibility allowed by that specific stochastic model does not help understanding uncertainty on that data set, which may come to many factors, like indecisiveness, experimentation or indifference. However, if a stochastic model shows a not-null deterministic-completeness, we can understand some degree of its uncertainty regarding the deterministic model under the theoretical structure that underlies that stochastic model.

To turn the matter more comprehensible, suppose a stochastic model, let's say a RUM, shows a high rational-completeness $\kappa^{\text{rational}}_{\text{RUM}}$, that is, setting yardstick to be the rational model for a given data set. Moreover, setting the yardstick to a MLP, we found an average completeness κ for the rational model. Then we conclude that the rational model captures data structure at the degree of its completeness κ, and the portion that is regarded as error is explained by RUM at the extent of $\kappa^{\text{rational}}_{\text{RUM}}$.

Although stochastic models mitigate the clear notion of rationality as we cannot see it through probabilities, they are built on the rational basis, that is, the economic theory. As for the RUM, there is a set of preferences and some of them shows x as a maximal element. So the probability of observing x chosen is simply the probability of the agent choose according to a preference in which x is a maximal element. That being said, we identify how much of the structure of a deterministic model is pointed out by the data and how much of its uncertainty is captured by the stochastic model, in such a way both of them we can explain on rational terms.

Doing so, we propose a way to further develop the completeness measure to better understand data, building a bridge between deterministic and stochastic models grounded on data. Henceforth, having two economic models, one deterministic and another stochastic, a researcher can link the economic assumptions behind the two of them to better explain revealed preferences without getting rid of a sturdy theoretical basis.

Since the method applies for a general stochastic model, it also applies to ANN. Thus, next we show the results of our method applied to investigate how some MLP and CNN (trained with 1000 observations from complete data and 1000 observation from incomple data), which play the role of stochastic models, can further improve rational, Silva & Riella (2020) and Guney, Richter & Tsur (2018) models fit to some random generated data. The results are presented on Tab. (3.12), in which the yardstick is set to be a $MLP_{\text{yardstick}}$ trained with a balanced data set with 50 observations from complete, incomplete and ordinary data and completeness is evaluated on a complete data set with 5000 observations.

Note that, again we set the size of the training batch of data to be over 1000 observations to simulate the best scenario in which the researcher have a balanced set of data with many observations. The obtained results for this experiment are summarized on the following Table and next discussions.

	Restrictiveness	Completeness	Deterministic-Completeness of MLP[2]	Deterministic-Completeness of CNN[3]
Rational	0.916 (0.001)	0.289 (0.078)	91.04%	80.05%
Silva & Riella (2020)	0.915 (0.001)	0.253 (0.078)	91.23%	80.47%
Guney, Richter & Tsur (2018)	0.913 (0.001)	0.285 (0.078)	91.06%	80.08%

Table 3.12: Results of the evaluation of different Deterministic-Completeness of a MLP and a CNN. Standard errors are displayed in parenthesis.

According to Table 3.12, Deterministic-Completeness evaluations of MLP shows that its potential of explaining residual uncertainty behind the data set when accounting for the considered deterministic models is around 91%. It turns out that the same potential of the CNN is around 80%. This results are compatible with the high completeness evaluations of the ANN, although they show high levels of restrictiveness. In fact, the completeness of the MLP is higher than that of the CNN, thus is expected that the MLP potential to comprise the uncertainty is higher as well.

Furthermore, simulating the deterministic-completeness of a stochastic model whose probabilities are drawn from uniform distribution, we found that the values were all null. It shows that a random uniform probabilistic model does not help at all to understand the uncertainty of the data we had. However, as pointed out by Table 3.12, ANN indeed help to understand a considerable portion of it, as far as its own structure is known.

3.7 Conclusion

In this paper we explore the model assessment method developed in Fudenberg, Gao & Liang (2020). We do experiments regarding the best decision for what they call a naive mapping, but we refer to it as a yardstick mapping, since one can vary it according to the research topic of interest. The yardstick plays a crucial role on evaluating restrictiveness and completeness of a model, which are measures closely related to data structure.

[2]The MLP being evaluated here shows a restrictiveness of 0.739 (standard error of 0.000) and completeness of 0.839 (standard error of 0.040), when setting the yardstick to be another MLP trained with complete, incomplete and ordinary data.

[3]The CNN being evaluated here shows a restrictiveness of 0.803 (standard error of 0.000) and completeness of 0.616 (standard error of 0.046), when setting the yardstick to be another MLP trained with complete, incomplete and ordinary data.

Our main findings indicate that ML techniques can be used together with economic models to uncover structure on data, rather than being a tool that replaces theoretical modeling at all. This way, setting the yardstick to be a MLP trained with a small balanced data set, showed the best assessment results. Furthermore, we reinforce that data preparation is a fundamental stage on choice data analysis and we conclude that imposing reflexivity on choice data shows a high gain on the ability to identify structure, but imposing completeness does not affect models' evaluation. Finally, we use completeness measure to build a bridge between deterministic and stochastic models, honing the ability to understand underlying preferences and uncertainty inherent to a real world conjuncture.

Since the contribution is mostly methodological, future works may evaluate economic models on real world data. Henceforth, one can identify the models that better unfold an environment behavior which is extremely handful to conduct policy analysis and counterfactual simulations based on preferences. Furthermore, next works may explore the potential of CNN on better identifying unfeasiable alternatives, when compared to MLP or Logit models.

Chapter 4

Updating Inner Perspectives about the World in a Subjective Manner

4.1 Introduction

Many of our daily choices involve a two-staged decision process, first we choose where to shop or in which restaurant to dine and then, once there, we choose what to buy or which meal to order. This first stage characterizes a choice over menus of alternatives and the second one a choice over the alternatives contained in the chosen menu. Since people often present multiple tastes, meaning the same person will sometimes order pasta and sometimes order salad, depending on the mood of the day, this second choice has a stochastic nature. It is reasonable, in this context, to expect some consistency of the decision maker-henceforth DM- when choosing between menus and between alternatives. As an example, if a menu with a single additional alternative is strictly preferred to one without it, it must be because the DM finds this additional alternative compelling and we ought to expect they would choose it with positive probability.

Ahn (2013) develop this connection between choice over menus and stochastic choice in the DLR-GP representation, that links the Dekel, Lipman & Rustichini (2001) framework of preferences over menus of lotteries to the Gul & Pesendorfer (2006) random expected utility representation for the DM's stochastic choice. They characterize the alignment between the two choice procedures with two very intuitive conditions, the first of which described in the example above.

Under these conditions the DM would perfectly anticipate their future preferences. It is reasonable to expect, however, that the DM may receive new information between the two stages of the decision process, meaning that either some expected states never realize or some unexpected states do. In the first case we could see some previously relevant alternatives never being chosen and, in the second, formerly irrelevant alternatives are chosen with positive probability. Ahn (2013) explicitly model this latter case in the *unforeseen contingencies* representation while the former is a straightforward result of their work that we formalize in Section 4.4.

This learning process, in which the underlying set of states is updated, is not restricted to the transition from preferences over menus to random choice. Indeed, Riella (2013) discusses a similar updating transition but between two preferences over menus. Intuitively, suppose the DM is choosing where to dine in a given night and reveal a preference during the morning and a different one after lunch. When do these two preferences reveal that the agent has learned new information about her future preferences and dropped some subjective states during lunch? They answer this question with the property of Flexibility Consistency, which uses a set of additional alternatives to identify which subjective states were dropped.

In this paper we study some remaining processes of learning. Particularly we focus on recognizing updates in random choice rules that follow a Finite Random Expected Utility procedure, an adaptation of the model in Gul & Pesendorfer (2006) with finite state spaces. As done in Ahn (2013), the finiteness of the state space requires the use of a tie-breaking rule with infinite support. We also study the conditions under which a collection of random choice rules may be understood as emerging from a partition of the state space that defines either another Finite Random Expected Utility model or a Preference Over Menus that admits a Dekel, Lipman & Rustichini (2001)-henceforth DLR- representation.

Our results allow us to characterize when different choice behaviors emerge from an updating process between them. Suppose the DM decides to avoid meat at mondays. When will this new revealed pattern, which we understand as a new random choice rule, be a Bayesian update from the previously revealed pattern? Meaning that some alternatives will be chosen less often, but the relative probabilities between unaffected choices remain unchanged.

For the main theorem in this paper the order of the updating is not really relevant, meaning we can explain both when the learning process leads to a drop of states or to the enrichment of the original set of states. In this sense, the theorem accommodates both the traditional notion of Bayesian Updating, in which the state space shrinks, and that of Reverse Bayesianism, discussed in Karni & Vierø (2013), in which the state space expands.

The remainder of this paper is organized as follow. In the next section we discuss the primitives and main definitions necessary to our work. In Section 4.3 we pose our main results, characterizing the updating between RCRs. We also provide the conditions under which a collection of RCRs build a partition of the original one after the updating. In Section 4.4 we study the regular updating from Menus to Random Choice and provide a partitioning result in this setting, similar to that of the previous section. In Section 4.5 we discuss the connections between the updating under the different frameworks. The last section presents our conclusions and the proofs are presented in Appendix A.2.

4.2 Setup

Let Z be a finite set with $|Z| \geq 2$, $\Delta(Z)$ be the space of probability measures on Z and $\mathcal{A} \subset \Delta(Z)$ be the collection of all nonempty, finite subsets of $\Delta(Z)$ endowed with the Hausdorff metric:

$$d_h(A, B) := \max\left\{\max_{p\in A}\min_{q\in B} d(p, q), \max_{q\in B}\min_{p\in A} d(p, q)\right\}.$$

We call an arbitrary element $A \in \mathcal{A}$ a choice problem or, similarly, a menu. Let $\Delta(\Delta(Z))$ denote the space of all probability distributions over $\Delta(Z)$. We will denote by S the set of possible states, be them subjective or objective, that will influence the DM's preferences over $\Delta(Z)$. Let $n := |Z|$, since $\Delta(Z) \subset \mathbb{R}^n$ we use the euclidean distance over lotteries and denote by $B_\epsilon(z)$ the open ball with radius ϵ and center z and by $\overline{B_\epsilon}(z)$ its closure. We denote by int$A$ the interior of a set A and by ri$A$ its relative interior.

Given a function (or a vector) $u \in \mathbb{R}^Z$ we denote the expected utility of the lottery $p \in \Delta(Z)$ with respect to u interchangeably as $u(p)$ and $u \cdot p$. We define the set of normalized (nonconstant) expected-utility functions by

$$\mathcal{U} := \left\{ u \in \mathbb{R}^Z : \sum_{z \in Z} u_z = 0, \sum_{z \in Z} u_z^2 = 1 \right\}.$$

We note that every nontrivial expected utility preference over $\Delta(Z)$ is represented by one and only one function $u \in \mathcal{U}$. By $\Delta^f(\mathcal{U})$ we mean the space of finitely additive probability measures on $\mathcal{U}$. Given an expected-utility function $u \in \mathbb{R}^Z$, we let $M(A, u)$ denote the maximizers of u in A:

$$M(A, u) := \left\{ p \in A : u(p) = \max_{q \in A} u(q) \right\}.$$

4.2.1 Random Choice Rules

We begin with the definition of a random choice rule:

Definition 1. A *random choice rule* (RCR) is a function $\rho : \mathcal{A} \to \Delta(\Delta Z)$ that associates to each choice problem A a probability measure ρ^A on A, meaning, for any $A \in \mathcal{A}$, $\rho^A(A) = 1$ and, if $B, C \in \mathcal{A}$ are such that $B \cap C = \emptyset$, then $\rho^A(B \cup C) = \rho^A(B) + \rho^A(C)$.

Let $U : S \times \Delta(Z) \mapsto \mathbb{R}$ be the agent's utility function across states, such that $U_s \in \mathbb{R}^Z$ is the expected-utility function that represents the agents preferences over $\Delta(Z)$ upon the realization of state $s \in S$. If we have that, for some $A \in \mathcal{A}$, $|M(A, U_s)| = 1$ for every $s \in S$, then the Finite Random Expected Utility representation of ρ on A would be resumed to

$$\rho^A(p) = \mu\left(\{s \in S : p \in M(A, U_s)\}\right),$$

where μ is a probability distribution over S.

As we work with a finite state space, though, we will need a rule to deal with situations where a state with positive probability of realization leads to a tie among available alternatives. In the original set-up of Gul & Pesendorfer (2006) this problem is averted as the authors show that it is always possible to achieve an infinite state space representation where each individual state has zero probability.[1] We follow Ahn (2013) by defining a tie-breaking rule.

[1]They deal with nonregular random utility functions, requiring a tie-breaking rule in the supplemental material to their paper.

Definition 2. Given a finite state space S, a *tie-breaking rule* for S is a map $\tau : S \to \Delta^f(\mathcal{U})$ that satisfies the following regularity condition for all $A \in \mathcal{A}$, $p \in A$ and $s \in S$:

$$\tau_s\left(\{u \in \mathcal{U} : u(p) > u(q), \forall q \in A \setminus \{p\}\}\right) = \tau_s\left(\left\{u \in \mathcal{U} : u(p) = \max_{q \in A} u(q)\right\}\right).$$

Note that, despite having finite states in S, the regularity condition implies that the tie-breaking rule τ_s cannot have a finite support on $\mathcal{U}$, otherwise it would itself lead back into ties among lotteries.

With this we can define the Finite Random Expected Utility representation.

Definition 3. A *Finite Random Expected Utility (FREU) representation* is a tuple (S, U, μ, τ), where S is a finite state space, $U : S \times \Delta(Z) \to \mathbb{R}$, μ is a probability distribution on S, and τ is a tie-breaking rule over S such that the following statements hold:

(i) For every $A \in \mathcal{A}$ and $p \in A$,

$$\rho^A(p) = \sum_{s \in S} \mu(s)\tau_s\left(\{u \in \mathcal{U} : p \in M(M(A, U_s), u)\}\right)$$

(ii) For any two distinct states $s, s' \in S$, U_s and $U_{s'}$ do not represent the same von Neumann-Morgestern (vNM) preference on $\Delta(Z)$.

(iii) For every $s \in S$, $\mu(s) > 0$ and U_s is nonconstant.

In this setup every FREU representation is essentially unique. Meaning that, if (S, U, μ, τ) and (S', U', μ', τ') represent the same random choice rule, then it must be the case that for any $s \in S$ there is a unique $s' \in S'$ such that, for every $A \subset \Delta(Z)$, $\arg\max U_s(A) = \arg\max U'_{s'}(A)$, $\mu(s) = \mu'(s')$ and $\tau_s = \tau'_{s'}$, meaning that, essentially, $S = S'$. Through the remainder of this paper, whenever we say two subjective states, s, s' are equal ($s = s'$), we mean that the utilities they imply, say U_s and $U'_{s'}$, represent the same vNM preferences.

Whenever two RCRs, ρ and ρ', have FREU representations (S, U, μ, τ) and (S', U', μ', τ') such that $S' \subseteq S$ and, for every $s \in S'$, $\mu'(s) = \frac{\mu(s)}{\mu(S')}$ and $\tau'_s = \tau_s$, we abuse notation by saying $(S', U, \mu_{S'}, \tau)$ is a FREU representation of ρ'. Note that $S' \subseteq S$ already implies that, for every $s \in S'$, U_s and U'_s represent the same vNM preferences over $\Delta(Z)$.

4.2.2 Preferences over Menus

In sections 4.4 and 4.5 we work with preferences over menus and its relations to RCRs. For that we use the canonical DLR representation with a finite subjective state space.

Definition 4. A *preference over menus* is a binary relation $\succsim \subseteq \mathcal{A} \times \mathcal{A}$.

Definition 5. A preference over menus $\succsim$ has a DLR representation if there is a tuple (S, U, μ), where S is a finite state space, $U : S \times \Delta(Z) \to \mathbb{R}$ is a state-dependent expected-utility function, and μ is a probability distribution on S, such that the following statements hold:

(i) $A \succsim B$ if and only if $V(A) \geq V(B)$, where $V : \mathcal{A} \to \mathbb{R}$ is defined by $V(A) = \sum_{s \in S} \mu(s) \max_{p \in A} U_s(p)$.

(ii) For any two distinct states $s, s' \in S$, U_s and $U_{s'}$ do not represent the same von Neumann-Morgenstern preference on $\Delta(Z)$.

(iii) For every $s \in S$, $\mu(s) > 0$ and U_s is nonconstant.

4.3 Updating Finite Random Expected Utility Representations

4.3.1 Main Result: Updating Between FREU representations

We proceed by stating our result of updating between FREU representations. Let ρ_1 and ρ_2 be two RCRs, (S, U, μ, τ) and (T, U', μ', τ') its respective FREU representations. Our main theorem is based upon the axiom of Random Consistency.

Axiom 1 (Random Consistency). *For any choice problem $A \in \mathcal{A}$ and $p, q \in A$, if $\rho_1^A(p)\rho_2^A(q) > \rho_1^A(q)\rho_2^A(p)$, then there exists a set $B \in \mathcal{A}$ and a radius $\delta > 0$ such that, $\rho_2^{A \cup B \cup \{p^\delta\}}(p^\delta) = 0$ for every $p^\delta \in B_\delta(p) \cap \Delta(Z)$, but $\rho_1^{A \cup B}(p) > 0$.*

What this axiom implies is that, given any menu $A \in \mathcal{A}$, whenever we see a difference in the relative probability of choice among two alternatives $p, q \in A$ between ρ_1 and ρ_2, it must be the case that there is some other set $B \in \mathcal{A}$ that inhibits the choice of p, and every neighboring alternative, by ρ_2 but not by ρ_1. The intuition is that some of the subjective states that led to the choice of p must have been dropped in the transition from ρ_1 to ρ_2, otherwise we should not see any change in relative probability among p and q. Therefore, by carefully selecting the set $B \in \mathcal{A}$, we can offer the DM some specific alternatives that will please them more than p in the remaining states that lead to its choice in ρ_2, but that are not as attractive in the states that were dropped due to the updating, meaning we still have $\rho_1(p) > 0$.

Clearly, this axiom depends on the space of alternatives being rich enough so we can always find such set B. Still, we can craft an example in a simplified setup, with states of nature that are external to the DM, that may help explicit the idea underlying the axiom. Let's say the DM is planning a barbecue with friends on a given Sunday. Suppose the options of drinks at the barbecue will be $A = \{beer,\ wine\}$. On the day of the barbecue two independent events may happen with $1/2$ probability, the DM may play soccer and the day might be sunny. Either if the day is sunny or the DM plays soccer they will prefer $beer$ over $wine$, meaning that, beforehand, the DM sees their probabilities of choice as $\rho_1^A(beer) = \frac{3}{4}$ and $\rho_1^A(wine) = \frac{1}{4}$. On friday, though, the DM starts feeling some knee pain, implying that playing soccer is out of the question. After learning this, their choice probabilities become $\rho_2^A(beer) = \rho_2^A(wine) = \frac{1}{2}$, depending only on the day being sunny or not. What Random Consistency implies is that, seeing this change in relative probabilities of choice, we must be able to find a new set of drinks B so that we have $\rho_2^{A\cup B}(beer) = 0$, but $\rho_1^{A\cup B}(beer) > 0$. Suppose the DM prefers the drink $caipirinha$ to $beer$ but only when it is sunny and they have not played soccer, and make $B = \{caipirinha\}$. Now we see that $\rho_2^{A\cup B}(beer) = 0$ and $\rho_1^{A\cup B}(beer) = \frac{2}{4} > 0$.

With Random Consistency we can state the main theorem of our paper.

Theorem 1. *Let ρ_1 and ρ_2 be two RCRs that admit FREU representations. The following statements are equivalent:*

1. *The stochastic choice functions ρ_1 and ρ_2 satisfy Random Consistency;*

2. *either $S \cap T = \emptyset$ or $T \subseteq S$ and (T, U, μ_T, τ) is a random expected utility representation of ρ_2, where μ_T is the Bayesian update of μ after the observation of T.*

As stated, theorem 1, implies that when ρ_1 and ρ_2 satisfy Random Consistency and have FREU representations, then either ρ_1 and ρ_2 share none of their subjective states or ρ_2 is the update of ρ_1 when the DM observes T. Random Consistency, therefore, can say very little about the agents behavior when $S \cap T = \emptyset$. If we want to make sure that at least one state is shared among S and T, meaning $\mu(T) > 0$, we can add the following very straightforward axiom:

Axiom 2. *For every $A \in \mathcal{A}$, $supp(\rho_1^A) \cap supp(\rho_2^A) \neq \emptyset$.*

Axiom 2 only says that, in any choice problem $A \in \mathcal{A}$, there will be at least one alternative for which both ρ_1 and ρ_2 attribute a positive probability of choice. Given the construction of the FREU representations, this implies that $S \cap T \neq \emptyset$.

Corollary 1. *Let ρ_1 and ρ_2 be two RCRs with FREU representations, then ρ_1 and ρ_2 satisfy Random Consistency and Axiom 2 if, and only if, $T \subseteq S$, μ' is the Bayesian update of μ after the observation of T and they share the same tie-breaking rule ($\tau = \tau'$).*

With Corollary 1 we have, therefore, the desired result of the necessary and sufficient conditions for ρ_2 to be the update of ρ_1 after the observation of T. Notice that, though we number the RCRs ρ_1 and ρ_2, there is no necessary order implied in the axioms or theorems, meaning the result may also imply that ρ_1 is the update of ρ_2 after the DM learns that there are some unforeseen states ($S \setminus T$) that might actually realize. This interpretation relates to the notion of Reverse Bayesianism, discussed in Karni & Vierø (2013).

4.3.2 Multiple Signals and Partitions

Suppose now that the information received comes from a set o signals that is sufficiently informative so that each subjective state is only realized after one possible signal, though the same signal may still lead to different subjective states in the second stage. In this case the collection of RCRs after the updating build a partition of the broader RCR from the first stage.

To characterize the relations between the original RCR an the collection formed after the signal in this setting, consider a finite collection of $I + 1$ random choice rules, ρ and $\{\rho_i\}_{i \in I}$, with FREU representations (S, U, μ, τ) and $(S_i, U^i, \mu^i, \tau^i)$, such that, for each $i \in I$, ρ and ρ_i satisfy Random Consistency, Axiom 2 and the following axioms.

Axiom 3. *For every $i, j \in I$, and $A \in \mathcal{A}$, if, for some $p \in ri\Delta(Z)$ and $\delta > 0$, we have $\rho_i^{A \cup \{p^\delta\}}(p^\delta) > 0$ for every $p^\delta \in B_\delta(p) \cap \Delta(Z)$, then there is $D \in \mathcal{A}$ with $\rho_i^{A \cup D \cup \{p^\delta\}}(p^\delta) > 0$, but $\rho_j^{A \cup D \cup \{p^\delta\}}(p^\delta) = 0$.*

Axiom 4. *For any choice problem $A \in \mathcal{A}$, supp(ρ^A) $= \bigcup_{i \in I}$ supp(ρ_i^A).*

Axiom 3 has an interpretation related to that of Random Consistency. It implies that, if ρ_i and ρ_j are the RCRs after different signals are received by the DM, then if all the alternatives in the neighborhood of some lottery p are chosen with positive probability from $A \in \mathcal{A}$ by ρ_i, it must be the case that there is a set $D \in \mathcal{A}$ that would prevent the choice of the neighborhood of p from $A \cup D$ to ρ_j but wouldn't prevent it to ρ_i. Lemma 1 states that this condition is sufficient to guarantee that the FREU representations of the RCRs in $\{\rho_i\}_{i \in I}$ have mutually disjoint state spaces. Axiom 4 is fairly simpler and only makes sure that every alternative chosen with positive probability before any signal keeps being chosen after at least one of the possible signals.

Lemma 1. *A collection of RCRs $\{\rho_i\}_{i \in I}$ with FREU representations $\{(S_i, U^i, \mu^i, \tau^i)\}_{i \in I}$ satisfies Axiom 3 if, and only if, for any $i, j \in I, i \neq j$, we have $S_i \cap S_j = \{\emptyset\}$.*

With the axioms and lemma above we can now state our result of partitioning for RCRs.

Proposition 1. *Let I be a finite set of indices and suppose ρ and $\{\rho_i\}_{i \in I}$ are random choice rules with FREU representations such that, for each $i \in I$, ρ and ρ_i satisfy Random Consistency and Axiom 2. Then, the collection $\{\rho_i\}_{i \in I}$ satisfy Axioms 3 and 4 if, and only if, the collection $\{S_i\}_{i \in I}$ is a partition of S and each ρ_i has a FREU representation $(S_i, U, \mu_{S_i}, \tau)$.*

4.4 Updating from Menus to Random Choice Rules

Here we develop the traditional Bayesian updating direction between preference over menus and random choice rules. This is the opposite direction of the *unforeseen contingencies* representations from Ahn (2013) and a straightforward application of the second part of the Proposition 2 from their paper. Beyond closing this small gap in the literature on the transition from menus to Random Choice, this result will be useful for the discussion on Section 4.5. We also explore the partitioning of a Preference Over Menus into a collection of RCRs, similarly to what we have done in the previous section between RCRs.

The following axiom is a restatement of Axiom 2 in Ahn (2013).

Axiom 5 (Sarver's Axiom 2)**.** *Let $A \in \mathcal{A}$ be a menu and $p \in \Delta(Z) \setminus A$ an arbitrary lottery. If there is $\delta > 0$ such that $\rho^{D \cup \{p^\delta\}}(p^\delta) > 0$ for every $p^\delta \in B_\delta(p) \cap \Delta(Z)$ and $D \in \mathcal{A}$ with $d_h(A, D) < \delta$, then $A \cup \{p\} \succ A$.*

Proposition 2. *Let ρ be a RCR that admits a FREU representation and $\succsim$ a preference over menus that admits a DLR representation, then ρ and $\succsim$ satisfy axiom 5 if, and only if, $\succsim$ has a DLR representation (S, U, μ) such that (T, U, μ_T, τ), $T \subseteq S$, is a FREU representation of ρ.*

We now turn to the question of when a collection of random choice rules, $\{\rho_i\}_{i \in I}$, with FREU representations $(S_i, U^i, \mu^i, \tau^i)$, is a partition of the subjective state space from a DLR representation of a preference over menus $\succsim$.

Axiom 6. *$A \cup \{p\} \succ A$ if, and only if, there is $i \in I$ and $\delta > 0$ such that, $\rho_i^{D \cup \{p^\delta\}}(p^\delta) > 0$ for every $p^\delta \in B_\delta(p) \cap \Delta(Z)$ and $D \in \mathcal{A}$ with $d_h(A, D) < \delta$.*

Proposition 3. *Let $\succsim$ be a preference over menus that admits a DLR representation and $\{\rho_i\}_{i \in I}$ be a collection of random choice rules with FREU representations. Then, $\{\rho_i\}_{i \in I}$ satisfies Axiom 3 and $(\succsim, \{\rho_i\}_{i \in I})$ satisfy Axiom 6 if, and only if, $\succsim$ has a DLR representation (S, U, μ) such that, for each $i \in I$, $(S_i, U, \mu_{S_i}, \tau_i)$, is a FREU representation of ρ_i and $\{S_i\}_{i \in I}$ is a partition of S.*

4.5 Connections of Updating Among Frameworks

In this section we discuss the connections between the axioms that characterize the updating among preferences over menus and random choice rules. Before anything else, lets define the *DLR-GP representation* as presented in Ahn (2013).

Definition 6 (DLR-GP representation)**.** Let $\succsim$ be a preference over menus and ρ a RCR. A *DLR-GP representation* of $(\succsim, \rho)$ is a tuple (S, U, μ, τ), where (S, U, μ) is a DLR representation of $\succsim$ and (S, U, μ, τ) is a FREU representation of ρ.

Now, let $\succsim_1$ and $\succsim_2$ be two preferences over menus and ρ_1 and ρ_2 be two RCRs such that $(\succsim_1, \rho_1)$ and $(\succsim_2, \rho_2)$ admit DLR-GP representations (S, U, μ, τ) and (T, U', μ', τ'), respectively. To present the connections of the updating between random choice rules and preferences over menus, we restate the Axiom 1 from Ahn (2013):

Axiom 7 (Sarver's Axiom 1)**.** *For any menu $A \in \mathcal{A}$ and lottery $q \in \Delta(Z)$, $A \cup \{q\} \succ A$ implies that $\rho^{A\cup\{q\}}(q) > 0$.*

With this we can now proceed to Corollary 2.

Corollary 2. *Let $\succsim_1$ and $\succsim_2$ be two preferences over menus and ρ_1 and ρ_2 be two random choice rules such that the pairs $(\succsim_1, \rho_1)$ and $(\succsim_2, \rho_2)$ admit DLR-GP representations. If ρ_1 and ρ_2 satisfy Random Consistency and Axiom 2, then:*

(i) $\succsim_2$ and ρ_1 satisfy Sarver's Axiom 1;

(ii) $\succsim_1$ and ρ_2 satisfy Sarver's Axiom 2.

What this corollary implies is that the connection built by Random Consistency and Axiom 2 among the RCRs from $(\succsim_1, \rho_1)$ and $(\succsim_2, \rho_2)$ transcribes into different relations between the RCRs and the preferences over menus beyond that of the DLR-GP representation. This result implies that $(\succsim_2, \rho_1)$ admits an unforeseen contingencies representation (T, S, U, μ, τ), which is the same as saying that there is a reverse updating from $\succsim_2$ to ρ_1, and that ρ_2 is an update from $\succsim_1$, as described in Proposition 2.

As relevant as what is stated in Corollary 2 is what is missing. We cannot establish the same relation between $\succsim_1$ and $\succsim_2$ without imposing additional conditions specifically over the preferences over menus. That happens because, even with all the structure imposed by the connection between ρ_1 and ρ_2, there may still be a considerable degree of variation of scale between the state dependent utilities U and U'. Though for each $t \in T$ we have that U_t and U'_t represent the same vNM preferences over $\Delta(Z)$, the variations in scale among states are enough to ensure we could have $A \succ_1 B$ and $B \succsim_2 A$, for some $A, B \in \mathcal{A}$, and not only because of the changes in the subjective state spaces from S to T. Therefore we cannot imply that the axiom of Flexibility Consistency presented on Riella (2013) will be satisfied and cannot characterize $\succsim_2$ as a proper Bayesian update from $\succsim_1$.

Though this means the implications of Theorem 1 do not propagate entirely to the relations among preferences over menus, we believe it opens up a discussion on what these subjective states mean in the FREU and in the DLR representations. What we see is that, apart from the tie-breaking rule, the FREU representation is perfectly defined by the expected probability of a given state, $\mu(s)$, and by the vNM preference given by U_s, while in the DLR representation, the scale of U_s with respect to the other states still carries relevant information that cannot be delimited from the perspective of the RCR implied by the DLR-GP representation.

4.6 Conclusion

In this paper we extended the theory of Bayesian and Reverse Bayesian updating to the learning revealed between random choice rules. We worked in a framework of Finite Random Expected Utilities already developed in Gul & Pesendorfer (2006) and Ahn (2013). We proposed the property of Random Consistency that is closely related to the Axiom 2 in Ahn (2013) and to the property of Flexibility Consistency in Riella (2013), which apply to the transition from menus to random choice and between menus, respectively. We developed the characterization of when a collection of random choice rules represents a partition of the state space from another broader random choice rule or from a Preference Over Menus. Finally we discussed the connections between our results and those in the related frameworks mentioned above.

Appendix A

Appendix

A.1 Codes

A.1.1 Four Alternatives Setup

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jul 16 09:24:38 2022

@author: mathe
OBS: verbose = 0 hides the keras.sequential.fit process on console
"""
import tensorflow
from tensorflow import keras
from tensorflow.keras import layers
#from sklearn.model_selection import KFold
#from sklearn.preprocessing import minmax_scale
#from sklearn.metrics import log_loss
from sklearn.metrics import auc
from itertools import product
from random import randint
from random import uniform
from random import random
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
#next line of code must be executed on older versions of tensorflow
tensorflow.enable_eager_execution()
bce = tensorflow.keras.losses.BinaryCrossentropy()

class restrictiveness():
    """
    A class to evaluate restrictiveness and completeness of a parametric aspiration-reference model. Generates random
    data, random non-parametric mappings. The general framework is set to have four alternatives such that there is a
    strict complete preorder ranking them.

    """

    def __init__(self, m):
        """
        Initialize an object of a class with valid, feasible options must be observable, and relevant, there are at
        least one observable option as well an feasible option, choice problemes and random mappings from them to
        choices.

        Parameters
        ----------
        m : int
            the number of random mappings to generate.

        Returns
        -------
        self.choice_problems : list of valid and relevant choice problems.

        self.mappings : list of m random mappings.

        """

        self.choice_problems_list = []
        t = list(product(range(2), repeat = 4))
        t.pop(0)
        for i in range(0,len(t)):
            s = list(product(range(2), repeat = 4))
            s.pop(0)
            remove_list = []
            for j in range (0,4):
                for k in range(0,len(s)):
                    if t[i][j] == 0 and s[k][j] == 1:
                        remove_list.append(s[k])
            list_to_append = [x for x in s if x not in remove_list]
            for k in range(0,len(list_to_append)):
                self.choice_problems_list.append(list(list_to_append[k]+t[i]))
        self.choice_problems = pd.DataFrame(self.choice_problems_list, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'])
        self.mappings = self.mapping_generate(m)
        self.asp_ref_model = self.f_theta()
        self.asp_only_model = self.f_theta_no_ref()
        self.all_model = self.f_naive_all()
        self.nothing_model = self.f_naive_nothing()
        self.rational = self.f_rational()

    def data_generate(self, n):
        """
        Generates data for valid choice problems considering a complete preorder. The individual must choose something.

        Parameters
        ----------
        n : int
            the number of observations of hypothetical data.

        Returns
        -------
        A data frame with n rows and 12 columns, the first four denotes the choices, the middle four denotes the feasible
        options, the last four denotes the observable options.

        """
        # The first 4 positions on the data are the choices
        choice_data_list = []
        for h in range(0,n):
            obs_temp = (0, 0, 0, 0) + tuple(self.choice_problems_list[randint(0,len(self.choice_problems_list)-1)])
            obs = list(obs_temp)
            while sum(obs[0:4]) < 1:
                for i in range(0,4):
                    if obs[i+4] == 1:
                        obs[i] = randint(0,1)
            choice_data_list.append(obs)
        choice_data = pd.DataFrame(choice_data_list, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L'])
        return(choice_data)

    def data_generate_ord_bin(self, n):
        """
        Generates data for valid choice problems, comprising individuals who choose nothing when there are two or more
        options to choose.

        Parameters
        ----------
        n : int
            the number of observations of hypothetical data.

        Returns
        -------
        A data frame with n rows and 12 columns, the first four denotes the choices, the middle four denotes the feasible
        options, the last four denotes the observable options.

        """
        # The first 4 positions on the data are the choices
        choice_data_list = []
        for h in range(0,n):
            obs_temp = (0, 0, 0, 0) + tuple(self.choice_problems_list[randint(0,len(self.choice_problems_list)-1)])
            obs = list(obs_temp)
            for i in range(0,4):
                if obs[i+4] == 1:
                    obs[i] = randint(0,1)
            choice_data_list.append(obs)
        choice_data = pd.DataFrame(choice_data_list, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L'])
        return(choice_data)

    def data_generate_inc_pref(self, n):
        """
        Generates data for valid choice problems from a ordinary binary relation, that could not be reflexive.

        Parameters
        ----------
        n : int
            the number of observations of hypothetical data.

        Returns
        -------
        A data frame with n rows and 12 columns, the first four denotes the choices, the middle four denotes the feasible
        options, the last four denotes the observable options.

        """
        # The first 4 positions on the data are the choices
        choice_data_list = []
        for h in range(0,n):
            obs_temp = (0, 0, 0, 0) + tuple(self.choice_problems_list[randint(0,len(self.choice_problems_list)-1)])
            obs = list(obs_temp)
            if sum(obs[4:8]) == 1:
                while sum(obs[0:4]) < 1:
                    for i in range(0,4):
                        if obs[i+4] == 1:
                            obs[i] = randint(0,1)
            else:
                for i in range(0,4):
                    if obs[i+4] == 1:
                        obs[i] = randint(0,1)
            choice_data_list.append(obs)
        choice_data = pd.DataFrame(choice_data_list, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L'])
        return(choice_data)
```

```
def mapping_generate(self, m, deterministic = True):
    """
    Generate random mappings from valid and relevant choice problems to actual choices.
    Mappings are choice correspondences, thus allowing multiple choices.
    Each mapping has 65 rows and 12 columns, the first four denotes feasible options, the middle four denotes observa
    ble alternatives and last four denotes the choices.

    Parameters
    ----------
    m : int
        the number of random mappings to generate.

    Returns
    -------
    A list of m random mappings.

    """
    # The last 4 positions on the mapping are the choices
    mapping_list = [] # A list of dataframes 65x12
    for i in range(0,m):
        mapping_predictions = []
        for j in range(0,len(self.choice_problems_list)):
            if deterministic == True:
                prediction_temp = tuple(self.choice_problems_list[j]) + (0, 0, 0, 0)
                prediction = list(prediction_temp)
                while sum(prediction[8:12]) < 1:
                    for h in range(0,4):
                        if prediction[h] == 1:
                            prediction[h+8] = randint(0,1)
            else:
                prediction_temp = tuple(self.choice_problems_list[j]) + (uniform(0, 1), uniform(0, 1), uniform(0, 1), uniform(0, 1))
                prediction = list(prediction_temp)
            mapping_predictions.append(prediction)
        mapping = pd.DataFrame(mapping_predictions, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L'])
        mapping_list.append(mapping)
    return(mapping_list)

def perf_round(self, x, gamma):
    """
    Round values greater than 0.54 to 1. Used to make probabilities of choice into choices for a neural network.

    Parameters
    ----------
    x : float
        value to be rounded.

    Returns
    -------
    Rounded number.

    """

    if x - int(x) >= gamma:
        x = int(x) + 1
    else:
        x = int(x)
    return(x)

def aspiration(self, T):
    """
    Get the index of the aspiration.

    Parameters
    ----------
```

```
        T : list
            a list of length 4 and elements 0 or 1.

        Returns
        -------
        i : int
            index of the aspiration.

        """
        for i in range(0,4):
            if T[i] == 1:
                return i

    def u_S(self, S, T):
        a = self.aspiration(T)
        alternatives = [np.array((0,3)), np.array((0,2)), np.array((0,1)), np.array((0,0))]
        dist = [int(np.linalg.norm(alternatives[a]-alternatives[x])) for x in range(0,4)]
        for i in range(0,len(S)):
            if S[i] == 0:
                dist[i] = 100
        reference = dist.index(min(dist))
        dist_ref = [int(np.linalg.norm(alternatives[reference]-alternatives[x])) for x in range(0,4)]
        for j in range(0,len(S)):
            if dist_ref[j] > int(np.linalg.norm(alternatives[a]-alternatives[reference])):
                S[j] = 0
        if reference != 3:
            if S[reference+1] == 1:
                choice = reference + 1
            else:
                choice = reference
        else:
            choice = reference
        return(choice)

    def u_S_no_ref(self, S, T):
        a = self.aspiration(T)
        alternatives = [np.array((0,3)), np.array((0,2)), np.array((0,1)), np.array((0,0))]
        dist = [int(np.linalg.norm(alternatives[a]-alternatives[x])) for x in range(0,4)]
        for i in range(0,len(S)):
            if S[i] == 0:
                dist[i] = 100
        reference = dist.index(min(dist))
        return(reference)

    def f_rational(self):
        """
        Creates a mapping from the choice problems of the 4 alternatives framework into choices
        according to the rational model, without observable but non feasible options. For this setup,
        it coincides with the aspiration-based choice model.

        Returns
        -------
        A data frame corresponding to the rational model, for each choice problem.

        """
        naive_list = []
        for i in range(0,len(self.choice_problems_list)):
            S = self.choice_problems_list[i][0:4]
            choice_index = self.aspiration(S)
            temp = [0, 0, 0, 0]
            temp[choice_index] = 1
            naive_predict_temp = tuple(self.choice_problems_list[i]) + tuple(temp)
            naive_predict = list(naive_predict_temp)
            naive_list.append(naive_predict)
        naive = pd.DataFrame(naive_list, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L'])
        return(naive)

    def f_naive_all(self):
        """
        Creates a mapping from the choice problems of the 4 alternatives framework into choices
        in such a way everything that is available to choose is chosen.

        Returns
        -------
        A data frame corresponding to the naive predictions, for each choice problem.

        """
        naive_list = []
        for i in range(0,len(self.choice_problems_list)):
            S = self.choice_problems_list[i][0:4]
            naive_predict_temp = tuple(self.choice_problems_list[i]) + tuple(S)
            naive_predict = list(naive_predict_temp)
            naive_list.append(naive_predict)
        naive = pd.DataFrame(naive_list, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L'])
        return(naive)

    def f_naive_nothing(self):
        """
        Creates a mapping from the choice problems of the 4 alternatives framework into choices
        in such a way nothing is chosen.

        Returns
        -------
        A data frame corresponding to the naive predictions, for each choice problem.

        """
        naive_list = []
        for i in range(0,len(self.choice_problems_list)):
            naive_predict_temp = tuple(self.choice_problems_list[i]) + (0, 0, 0, 0)
            naive_predict = list(naive_predict_temp)
            naive_list.append(naive_predict)
        naive = pd.DataFrame(naive_list, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L'])
        return(naive)

    def f_theta(self):
        """
        Creates a mapping from the choice problems of the 4 alternatives framework into choices
        according to aspiration-based reference dependance model.

        Returns
        -------
        A data frame corresponding to aspiration-based reference dependance model's predictions
        for each choice problem.

        """
        model_list = []
        for i in range(0,len(self.choice_problems_list)):
            S = self.choice_problems_list[i][0:4]
            T = self.choice_problems_list[i][4:8]
            if S == T:
                choice_index = self.aspiration(T)
            else:
                choice_index = self.u_S(S, T)
            temp = [0, 0, 0, 0]
            temp[choice_index] = 1
            model_predict_temp = tuple(self.choice_problems_list[i]) + tuple(temp)
            model_predict = list(model_predict_temp)
            model_list.append(model_predict)
        model = pd.DataFrame(model_list, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L'])
        return(model)

    def f_theta_no_ref(self):
        """
        Creates a mapping from the choice problems of the 4 alternatives framework into choices
        according to aspiration-based choice model.

        Returns
        -------
        A data frame corresponding to aspiration-based choice model's predictions, for each
        choice problem.

        """
        model_list = []
        for i in range(0,len(self.choice_problems_list)):
            S = self.choice_problems_list[i][0:4]
            T = self.choice_problems_list[i][4:8]
            if S == T:
                choice_index = self.aspiration(T)
            else:
                choice_index = self.u_S_no_ref(S, T)
            temp = [0, 0, 0, 0]
            temp[choice_index] = 1
            model_predict_temp = tuple(self.choice_problems_list[i]) + tuple(temp)
            model_predict = list(model_predict_temp)
            model_list.append(model_predict)
        model = pd.DataFrame(model_list, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L'])
        return(model)

    def f_logit(self, data, deterministic = True):
        x_train = data[['E', 'F', 'G', 'H', 'I', 'J', 'K', 'L']]
        y_train = data[['A', 'B', 'C', 'D']]
        logit_model = keras.Sequential([
            keras.Input(shape=(8)),
            layers.Dense(4, activation = 'sigmoid')])
        logit_model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
        logit_model.fit(x_train, y_train, batch_size = 32, epochs = 100)

        predict_df = logit_model.predict(self.choice_problems)
        predict_df = pd.DataFrame(predict_df)

        if deterministic == True:
            #predict_df = predict_df.applymap(self.perf_round)
            for i in range(0,len(self.choice_problems)):
                for j in range(0,4):
                    if self.choice_problems.iat[i,j] == 0:
                        predict_df.iat[i,j] = 0
                    else:
                        predict_df.iat[i,j] = int(np.random.choice([0,1], 1, p = [1 - predict_df.iat[i,j], predict_df.iat[i,j]]))
        nn = pd.concat([self.choice_problems, predict_df], axis=1)
        return(nn)

    def f_NN(self, data, N, deterministic = True):
        """
        Creates a mapping from the choice problems of the 4 alternatives framework into choices
        according to a neural network. By default, the neural net has two hidden layers, each
        with same amount of neurons.

        Parameters
        ----------
        data : pandas.DataFrame
            Data used to train the neural net.
        N : int
```

```
            Number of neurons for each hidden layer of the neural net.

        Returns
        -------
        A data frame corresponding to the treined neural net's predictions, for each choice problem.

        """
        x_train = data[['E', 'F', 'G', 'H', 'I', 'J', 'K', 'L']]
        y_train = data[['A', 'B', 'C', 'D']]
        one_hidden_layer_nn = keras.Sequential([
            keras.Input(shape=(8)),
            layers.Dense(N, activation = 'relu'),
            layers.Dense(N, activation = 'relu'),
            layers.Dense(4, activation = 'sigmoid')])
        one_hidden_layer_nn.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
        one_hidden_layer_nn.fit(x_train, y_train, batch_size = 32, epochs = 100)

        predict_df = one_hidden_layer_nn.predict(self.choice_problems)
        predict_df = pd.DataFrame(predict_df)

        if deterministic == True:
            #predict_df = predict_df.applymap(self.perf_round)

            for i in range(0,len(self.choice_problems)):
                for j in range(0,4):
                    if self.choice_problems.iat[i,j] == 0:
                        predict_df.iat[i,j] = 0
                    else:
                        predict_df.iat[i,j] = int(np.random.choice([0,1], 1, p = [1 - predict_df.iat[i,j], predict_df.iat[i,j]]))

        nn = pd.concat([self.choice_problems, predict_df], axis=1)
        return(nn)

    def f_CNN(self, data, N, deterministic = True):
        x_train = data[['E', 'F', 'G', 'H', 'I', 'J', 'K', 'L']]
        x_train = np.array(x_train)
        x_train = x_train.reshape(len(x_train), 8, 1)
        y_train = data[['A', 'B', 'C', 'D']]
        #y_train = np.array(y_train)
        #y_train = y_train.reshape(len(y_train), 8, 1)
        convolutional_nn = keras.Sequential([
            layers.Conv1D(N, kernel_size=3, activation='relu', input_shape=(8, 1)),
            layers.Conv1D(N, kernel_size=3, activation='relu'),
            layers.Dropout(0.5),
            layers.MaxPooling1D(pool_size=2),
            layers.Flatten(),
            layers.Dense(4, activation = 'sigmoid')])
        convolutional_nn.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
        convolutional_nn.fit(x_train, y_train, batch_size = 32, epochs = 100)

        x_test = np.array(self.choice_problems)
        x_test = x_test.reshape(len(x_test), 8, 1)
        predict_df = convolutional_nn.predict(x_test)
        predict_df = pd.DataFrame(predict_df)

        if deterministic == True:
            #predict_df = predict_df.applymap(self.perf_round)

            for i in range(0,len(self.choice_problems)):
                for j in range(0,4):
                    if self.choice_problems.iat[i,j] == 0:
                        predict_df.iat[i,j] = 0
                    else:
                        predict_df.iat[i,j] = int(np.random.choice([0,1], 1, p = [1 - predict_df.iat[i,j], predict_df.iat[i,j]]))
        nn = pd.concat([self.choice_problems, predict_df], axis=1)
        return(nn)

    def model_fit(self, data, modelo):
        """
        Generate a model's prediction, choices, based on input data, choice problem.

        Parameters
        ----------
        data : pandas.DataFrame
            A data frame with n rows and 12 columns, with the last 8 being the choice problem, feasible and observable
            options).
        modelo : pandas.DataFrame
            A data frame, corresponding to a mapping or a model, with 12 columns, being the first eight the choice
            problem and the last 4 the choice correspondent to the choice problem.

        Returns
        -------
        A data frame containing the predictions of the model for the data input.

        """
        model_fit = pd.merge(data, modelo,  how='left', left_on=['E', 'F', 'G', 'H', 'I', 'J', 'K', 'L'],
                                         right_on = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'])
        model_fit = model_fit.iloc[:,20:24]
        return(model_fit)

    def distance_map(self, f, F_Theta, deterministic = True):
        """
        Evaluates distance between mappings, according to the usual euclidean distance.

        Parameters
        ----------
        f : pandas.DataFrame
            Mapping from choice problems into choices, 65 rows and 12 columns.
        F_Theta : list
            A list containing the mappings corresponding to the parametrizations of the model, data frames with 65 rows
            and 12 columns. On the simplest case, it comprises a parametrization of interest and the naive mapping.

        Returns
        -------
        The minimum distance of model parametrizations and a given map.

        """
        dist_list = []
        #f_func = f[['I', 'J', 'K', 'L']]
        if deterministic == True:
            f_func = np.array(f)
            for i in range(0,len(F_Theta)):
                #parametrization_pred = F_Theta[i][['I', 'J', 'K', 'L']]
                parametrization_pred = np.array(F_Theta[i])
                dist = [np.linalg.norm(f_func[j]-parametrization_pred[j]) for j in range(0,len(f))]
                dist = np.array(dist)
                mean_dist = dist.mean()
                dist_list.append(mean_dist)
        else:
            f_func = f[['I', 'J', 'K', 'L']]
            f_func = np.array(f_func)
            for i in range(0,len(F_Theta)):
                parametrization_pred = F_Theta[i][[0, 1, 2, 3]]
                parametrization_pred = np.array(parametrization_pred)
                dist = [bce(f_func[j], parametrization_pred[j]).numpy() for j in range(0,len(f))]
                dist = np.array(dist)
                mean_dist = dist.mean()
                dist_list.append(mean_dist)
        dist_list = np.array(dist_list)
        error = dist_list.min()
        return(error)

    def best_map(self, data, det = True):
        """
        Iterates over 1000 maps to find the map that minimizes the expected loss function for the given data.

        Parameters
        ----------
        data : pandas.DataFrame
            Data in which the function iterates over random mappings to find the best one.

        Returns
        -------
        The best map for the given data.

        """
        if det == True:
            maps = self.mapping_generate(1000)
        else:
            maps = self.mapping_generate(1000, deterministic = det)
            """
            for k in range(0,len(maps)):
                maps[k][['I','J','K','L']] = maps[k][['I','J','K','L']].replace(0, 0.001)
                maps[k][['I','J','K','L']] = maps[k][['I','J','K','L']].replace(1, 0.999)
            """
        dist_list = []
        Y = data[['A', 'B', 'C', 'D']]
        Y = np.array(Y)
        for i in range(0,len(maps)):
            map_fit = self.model_fit(data, maps[i])
            map_fit = np.array(map_fit)
            if det == True:
                dist = [np.linalg.norm(Y[j] - map_fit[j]) for j in range(0,len(Y))]
            else:
                dist = [bce(Y[j], map_fit[j]).numpy() for j in range(0,len(Y))]
            dist = np.array(dist)
            mean_dist = dist.mean()
            dist_list.append(mean_dist)
        dist_list = np.array(dist_list)
        optimal_map = np.argmin(dist_list)
        return(maps[optimal_map])

    def mean_discrepancy(self, F_Theta, deterministic = True):
        """
        Evaluates the restrictiveness of a model, taking into account the respective naive mapping.

        Parameters
        ----------
        F_Theta : list
            A list containing data frames representing model's parametrizations mappings, with 65 rows and 12
            columns. By the defaut, the last element must be the naive mapping.

        Returns
        -------
        A numpy.array with the discrepancy for each random generated mapping.

        """
        delta_list = []
        #F_theta_predictions = []
        f_naive = []
        #Y = data[['A','B','C','D']]
        """
        for j in range(0,len(F_Theta)):
            predictions = self.model_fit(data, F_Theta[j])
            F_theta_predictions.append(predictions)
        """
        f_naive.append(F_Theta[-1])
        for i in range(0,len(self.mappings)):
            #mapping_fit = self.model_fit(data, self.mappings[i])
            if deterministic == True:
                delta = self.distance_map(self.mappings[i], F_Theta, deterministic = True)/self.distance_map(self.mappings[i],
                                                                                   f_naive, deterministic = True)
            else:
                delta = self.distance_map(self.mappings[i], F_Theta, deterministic = False)/self.distance_map(self.mappings[i],
                                                                                   f_naive, deterministic = False)
            delta_list.append(delta)
        delta_list = np.array(delta_list)
        return(delta_list)

    def mean_discrepancy_std_error(self, F_Theta, deterministic = True):
        """
        Evaluates the restrictiveness of a model, taking into account the respective naive mapping.

        Parameters
        ----------
        F_Theta : list
            A list containing data frames representing model's parametrizations mappings, with 65 rows and 12
            columns. By the defaut, the last element must be the naive mapping.

        Returns
        -------
        A numpy.array with the discrepancy for each random generated mapping.

        """
        delta_list = []
        #F_theta_predictions = []
        f_naive = []
        #Y = data[['A','B','C','D']]
        """
        for j in range(0,len(F_Theta)):
            predictions = self.model_fit(data, F_Theta[j])
            F_theta_predictions.append(predictions)
        """
        f_naive.append(F_Theta[-1])
        for i in range(0,len(self.mappings)):
            #mapping_fit = self.model_fit(data, self.mappings[i])
            if deterministic == True:
                delta = self.distance_map(self.mappings[i], F_Theta, deterministic = True)/self.distance_map(self.mappings[i],
                                                                                   f_naive, deterministic = True)
            else:
                delta = self.distance_map(self.mappings[i], F_Theta, deterministic = False)/self.distance_map(self.mappings[i],
                                                                                   f_naive, deterministic = False)
            delta_list.append(delta)
        delta_list = np.array(delta_list)
        delta_list = np.square(delta_list - delta_list.mean())
        ep = np.sqrt(delta_list.mean()/len(self.mappings))
        return(ep)

    def data_split(self, data, k):
        """
        Shuffles data and splits it into groups. The first k-1 groups have the same length. The last k
        group only has the same size as other groups if the division of rows of data by k is an integer.

        Parameters
        ----------
        data : pandas.DataFrame
            Data to be splitted.
        k : int
            Number of groups to be formed.

        Returns
        -------
        A list with k data frames.

        """
        group_size = int(len(data)/k)
        folds = []
        data = data.sample(frac=1).reset_index()
        del data['index']
        for i in range(0,k):
            if i < k-1:
                paper = data.iloc[i*group_size:(i+1)*group_size,]
                folds.append(paper)
            else:
                paper = data.iloc[i*group_size:,]
                folds.append(paper)
        return(folds)

    def k_fold(self, data_splitted, F, deterministic = True):
        """
        Perform the k-fold cross validation of a model, evaluating the error for each group of the data.

        Parameters
        ----------
        data_splitted : list
            A list of groups of data.
        F : list
            A list of mappings, may be random or a set of a model's parametrizations.

        Returns
        -------
        The mean error of the k groups.

        """
        group_size = len(data_splitted[0])
        error_list = []
        for i in range(0,len(data_splitted)):
            dist_list = []
            temp = [x for x in data_splitted if not(x.equals(data_splitted[i]))]
            test = data_splitted[i]
            Y_test = test[['A', 'B', 'C', 'D']]
            Y_test = np.array(Y_test)
            train = pd.concat(temp, ignore_index = True)
            Y_train = train[['A', 'B', 'C', 'D']]
            Y_train = np.array(Y_train)
            for j in range(0, len(F)):
                mapping_fit = self.model_fit(train, F[j])
                mapping_fit = np.array(mapping_fit)
                if deterministic == True:
                    dist = [np.linalg.norm(Y_train[k]-mapping_fit[k]) for k in range(0,len(Y_train))]
                else:
                    dist = [bce(Y_train[k], mapping_fit[k]) for k in range(0,len(Y_train))]
                dist = np.array(dist)
                dist_sum = dist.sum()
                dist_list.append(dist_sum)
            dist_list = np.array(dist_list)
            dist_list = (1/len(train))*dist_list
            f_k = np.argmin(dist_list)
            optimal_fit = self.model_fit(test, F[f_k])
            optimal_fit = np.array(optimal_fit)
            if deterministic == True:
                optimal_dist = [np.linalg.norm(Y_test[h]-optimal_fit[h]) for h in range(0,len(Y_test))]
            else:
                optimal_dist = [bce(Y_test[h], optimal_fit[h]) for h in range(0,len(Y_test))]
            optimal_dist = np.array(optimal_dist)
            optimal_dist_sum = optimal_dist.sum()
            error = (1/group_size)*optimal_dist_sum
            error_list.append(error)
        error_list = np.array(error_list)
        error_list_mean = error_list.mean()
        return(error_list_mean)

    def completeness_std_error(self, data_splitted, F, f_best, deterministic = True):
        """
        Perform the k-fold cross validation of a model, evaluating the error for each group of the data.

        Parameters
        ----------
        data_splitted : list
            A list of groups of data.
        F : list
            A list of mappings, may be random or a set of a model's parametrizations.

        Returns
        -------
        The mean error of the k groups.

        """
        group_size = len(data_splitted[0])
        var_list = []
        for i in range(0,len(data_splitted)):
            dist_list = []
            temp = [x for x in data_splitted if not(x.equals(data_splitted[i]))]
            test = data_splitted[i]
            Y_test = test[['A', 'B', 'C', 'D']]
            Y_test = np.array(Y_test)
            train = pd.concat(temp, ignore_index = True)
            Y_train = train[['A', 'B', 'C', 'D']]
            Y_train = np.array(Y_train)
            for j in range(0, len(F)):
                mapping_fit = self.model_fit(train, F[j])
                mapping_fit = np.array(mapping_fit)
                if deterministic == True:
                    dist = [np.linalg.norm(Y_train[k]-mapping_fit[k]) for k in range(0,len(Y_train))]
                else:
                    dist = [bce(Y_train[k], mapping_fit[k]) for k in range(0,len(Y_train))]
                dist = np.array(dist)
                dist_sum = dist.sum()
                dist_list.append(dist_sum)
            dist_list = np.array(dist_list)
            dist_list = (1/len(train))*dist_list
            f_k = np.argmin(dist_list)
            optimal_fit = self.model_fit(test, F[f_k])
            optimal_fit = np.array(optimal_fit)
            best_fit = self.model_fit(test, f_best)
            best_fit = np.array(best_fit)
            if deterministic == True:
                optimal_dist = [np.linalg.norm(Y_test[h]-optimal_fit[h]) for h in range(0,len(Y_test))]
                best_dist = [np.linalg.norm(Y_test[h]-best_fit[h]) for h in range(0,len(Y_test))]
            else:
                optimal_dist = [bce(Y_test[h], optimal_fit[h]) for h in range(0,len(Y_test))]
                best_dist = [bce(Y_test[h], best_fit[h]) for h in range(0,len(Y_test))]
            optimal_dist = np.array(optimal_dist)
```

```
            best_dist = np.array(best_dist)
            delta_z = optimal_dist - best_dist
            delta_z_sum = delta_z.sum()
            error = (1/group_size)*delta_z_sum
            delta_list = np.square(delta_z - error)
            var_sum = delta_list.sum()
            var = (1/(group_size-1))*var_sum
            ### IMPORTANTE ###
            var_list.append(var)
        var_list = np.array(var_list)
        var_list_sum = var_list.sum()
        f_star = []
        f_star.append(f_best)
        f_naive = []
        f_naive.append(F[-1])
        cv_best = self.k_fold(data_splitted, f_star, deterministic = deterministic)
        cv_f_naive = self.k_fold(data_splitted, f_naive, deterministic = deterministic)
        std_variation = np.sqrt(var_list_sum/(cv_f_naive-cv_best))
        ep = np.sqrt(std_variation/group_size)
        return(ep)

    def completeness(self, data_list, F_Theta, f_best, det = True):
        """
        Evaluates the completeness of a model, according to the naive mapping.

        Parameters
        ----------
        data_list : list
            A list of groups of data..
        F_Theta : list
            A list of model's parametrizations. By default, the last element must be the naive mapping.
        f_best : pandas.DataFrame
            A data frame corresponding to the best mapping for the data correspondent to data_list.

        Returns
        -------
        The completeness level of the model.

        """
        #data = pd.concat(data_list, ignore_index = True)
        f_star = []
        f_star.append(f_best)
        f_naive = []
        f_naive.append(F_Theta[-1])
        cv_best = self.k_fold(data_list, f_star, deterministic = det)
        cv_F_Theta = self.k_fold(data_list, F_Theta, deterministic = det)
        cv_f_naive = self.k_fold(data_list, f_naive, deterministic = det)
        #completeness_level = (cv_F_Theta - cv_mappings)/(cv_f_naive - cv_mappings)
        completeness_level = (cv_f_naive - cv_F_Theta)/(cv_f_naive - cv_best)
        return(completeness_level)

    def similarity(self, det_model, sto_model, n):
        restrictiveness_list = []
        threshold = np.arange(0,1.05,0.05)
        round_functions = []
        for i in range(0,len(threshold)):
            def pf(x, gamma = threshold[i]):
                if x - int(x) >= gamma:
                    x = int(x) + 1
                else:
                    x = int(x)
                return(x)
            round_functions.append(pf)
        sto_predict = sto_model[[0,1,2,3]]
        run_list = []
        for i in range(0,n):
            sto_temp = pd.DataFrame(np.zeros((65,4)))
            naive_tp = pd.DataFrame(np.zeros((65,4)))
            for k in range(0,len(self.choice_problems)):
                for j in range(0,4):
                    if self.choice_problems.iat[k,j] == 0:
                        sto_temp.iat[k,j] = 0
                        naive_tp.iat[k,j] = 0
                    else:
                        sto_temp.iat[k,j] = int(np.random.choice([0,1], 1, p = [1 - sto_predict.iat[k,j], sto_predict.iat[k,j]]))
                        naive_tp.iat[k,j] = sto_predict.iat[k,j]
            sto_temp = pd.concat([self.choice_problems, sto_temp], axis=1)
            run_list.append(sto_temp)
        for v in range(0,len(round_functions)):
            naive = naive_tp.applymap(round_functions[v])
            naive = pd.concat([self.choice_problems, naive], axis=1)
            delta_list = []
            det_list = []
            f_naive = []
            det_list.append(det_model)
            det_list.append(naive)
            f_naive.append(naive)
            for i in range(0,n):
                delta = self.distance_map(run_list[i], det_list, deterministic = True)/self.distance_map(run_list[i], f_naive, deterministic = True)
                delta_list.append(delta)
            delta_list = np.array(delta_list)
            restrictiveness_list.append(delta_list.mean())
        threshold = pd.DataFrame(threshold)
        restrictiveness_list = pd.DataFrame(restrictiveness_list)
        graph = pd.concat([threshold, restrictiveness_list], axis=1, ignore_index=True)
        return(graph)

    def curve(self, data, sto_model, n):
        Y_data = data[['A', 'B', 'C', 'D']]
        Y_data = np.array(Y_data)
        restrictiveness_list = []
        threshold = np.arange(0.01,1.01,0.01)
        round_functions = []
        for i in range(0,len(threshold)):
            def pf(x, gamma = threshold[i]):
                if x - int(x) >= gamma:
                    x = int(x) + 1
                else:
                    x = int(x)
                return(x)
            round_functions.append(pf)
        sto_predict = sto_model[[0,1,2,3]]
        run_list = []
        for i in range(0,n):
            sto_temp = pd.DataFrame(np.zeros((65,4)))
            naive_tp = pd.DataFrame(np.zeros((65,4)))
            for k in range(0,len(self.choice_problems)):
                for j in range(0,4):
                    if self.choice_problems.iat[k,j] == 0:
                        sto_temp.iat[k,j] = 0
                        naive_tp.iat[k,j] = 0
                    else:
                        sto_temp.iat[k,j] = int(np.random.choice([0,1], 1, p = [1 - sto_predict.iat[k,j], sto_predict.iat[k,j]]))
                        naive_tp.iat[k,j] = sto_predict.iat[k,j]
            sto_temp = pd.concat([self.choice_problems, sto_temp], axis=1)
            run_list.append(sto_temp)
        for v in range(0,len(round_functions)):
            naive = naive_tp.applymap(round_functions[v])
            naive = pd.concat([self.choice_problems, naive], axis=1)
            naive_fit = self.model_fit(data, naive)
            naive_fit = np.array(naive_fit)
            naive_dist = [np.linalg.norm(Y_data[h]-naive_fit[h]) for h in range(0,len(Y_data))]
            naive_dist = np.array(naive_dist)
            naive_error = naive_dist.mean()
            delta_list = []
            for i in range(0,n):
                realization_fit = self.model_fit(data, run_list[i])
                realization_fit = np.array(realization_fit)
                realization_dist = [np.linalg.norm(Y_data[h]-realization_fit[h]) for h in range(0,len(Y_data))]
                realization_dist = np.array(realization_dist)
                realization_error = realization_dist.mean()
                realization_error = min(realization_error,naive_error)
                kappa = (naive_error-realization_error)/naive_error
                delta_list.append(kappa)
            delta_list = np.array(delta_list)
            restrictiveness_list.append(delta_list.mean())
        threshold = pd.DataFrame(threshold)
        restrictiveness_list = pd.DataFrame(restrictiveness_list)
        graph = pd.concat([threshold, restrictiveness_list], axis=1, ignore_index=True)
        return(graph)
```

A.1.2 Eight Alternatives Setup

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jul 16 09:24:38 2022

@author: mathe
OBS: verbose = 0 hides the keras.sequential.fit process on console
"""
import tensorflow
from tensorflow import keras
from tensorflow.keras import layers
#from sklearn.model_selection import KFold
#from sklearn.preprocessing import minmax_scale
#from sklearn.metrics import
from sklearn.metrics import mean_squared_error as mse
#from sklearn.metrics import auc
from itertools import product
from random import randint
from random import uniform
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
#next line of code must be executed on older versions of tensorflow
tensorflow.enable_eager_execution()
bce = tensorflow.keras.losses.BinaryCrossentropy()

class restrictiveness_2dgrid():
    """
    A class to evaluate restrictiveness and completeness of a parametric aspiration-reference model. Generates random
    data, random non-parametric mappings. The general framework is set to have four alternatives such that there is a
    strict complete preorder ranking them.

    """

    def __init__(self, m):
        """
        Initialize an object of a class with valid, feasible options must be observable, and relevant, there are at
        least one observable option as well an feasible option, choice problemes and random mappings from them to
        choices.

        Parameters
        ----------
        m : int
            the number of random mappings to generate.

        Returns
        -------
        self.choice_problems : list of valid and relevant choice problems.

        self.mappings : list of m random mappings.

        """

        self.choice_problems_list = []
        t = list(product(range(2), repeat = 8))
        t.pop(0)
        for i in range(0,len(t)):
            s = list(product(range(2), repeat = 8))
            s.pop(0)
            remove_list = []
            for j in range (0,8):
                for k in range(0,len(s)):
                    if t[i][j] == 0 and s[k][j] == 1:
                        remove_list.append(s[k])
            list_to_append = [x for x in s if x not in remove_list]
            for k in range(0,len(list_to_append)):
                self.choice_problems_list.append(list(list_to_append[k]+t[i]))
        self.choice_problems = pd.DataFrame(self.choice_problems_list,
                                            columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',
                                                     'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P'])
        self.mappings = self.mapping_generate(m)
        self.asp_ref_model = self.f_theta()
        self.asp_only_model = self.f_theta_no_ref()
        self.all_model = self.f_naive_all()
        self.nothing_model = self.f_naive_nothing()
        self.rational = self.f_rational()

    def data_generate(self, n):
        """
        Generates data for valid choice problems considering a complete preorder. The individual must choose something.

        Parameters
        ----------
        n : int
            the number of observations of hypothetical data.

        Returns
        -------
        A data frame with n rows and 12 columns, the first four denotes the choices, the middle four denotes the feasible
        options, the last four denotes the observable options.

        """
        # The first 4 positions on the data are the choices
        choice_data_list = []
        for h in range(0,n):
            obs_temp = (0, 0, 0, 0, 0, 0, 0, 0) + tuple(self.choice_problems_list[randint(0,len(self.choice_problems_list)-1)])
            obs = list(obs_temp)
            while sum(obs[0:8]) < 1:
                for i in range(0,8):
                    if obs[i+8] == 1:
                        obs[i] = randint(0,1)
            choice_data_list.append(obs)
    choice_data = pd.DataFrame(choice_data_list,
                                        columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
                                                        'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'X', 'Y'])
    return(choice_data)

def data_generate_ord_bin(self, n):
    """
    Generates data for valid choice problems, comprising individuals who choose nothing when there are two or more
    options to choose.

    Parameters
    ----------
    n : int
        the number of observations of hypothetical data.

    Returns
    -------
    A data frame with n rows and 12 columns, the first four denotes the choices, the middle four denotes the feasible
    options, the last four denotes the observable options.

    """
    # The first 4 positions on the data are the choices
    choice_data_list = []
    for h in range(0,n):
        obs_temp = (0, 0, 0, 0, 0, 0, 0, 0) + tuple(self.choice_problems_list[randint(0,len(self.choice_problems_list)-1)])
        obs = list(obs_temp)
        for i in range(0,8):
            if obs[i+8] == 1:
                obs[i] = randint(0,1)
        choice_data_list.append(obs)
    choice_data = pd.DataFrame(choice_data_list,
                                        columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
                                                        'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'X', 'Y'])
    return(choice_data)

def data_generate_inc_pref(self, n):
    """
    Generates data for valid choice problems from a ordinary binary relation, that could not be reflexive.

    Parameters
    ----------
    n : int
        the number of observations of hypothetical data.

    Returns
    -------
    A data frame with n rows and 12 columns, the first four denotes the choices, the middle four denotes the feasible
    options, the last four denotes the observable options.

    """
    # The first 4 positions on the data are the choices
    choice_data_list = []
    for h in range(0,n):
        obs_temp = (0, 0, 0, 0, 0, 0, 0, 0) + tuple(self.choice_problems_list[randint(0,len(self.choice_problems_list)-1)])
        obs = list(obs_temp)
        if sum(obs[8:16]) == 1:
            while sum(obs[0:8]) < 1:
                for i in range(0,8):
                    if obs[i+8] == 1:
                        obs[i] = randint(0,1)
        else:
            for i in range(0,8):
                if obs[i+8] == 1:
                    obs[i] = randint(0,1)
        choice_data_list.append(obs)
    choice_data = pd.DataFrame(choice_data_list,
                                        columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
                                                'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'X', 'Y'])
    return(choice_data)

def mapping_generate(self, m, deterministic = True):
    """
    Generate random mappings from valid and relevant choice problems to actual choices.
    Mappings are choice correspondences, thus allowing multiple choices.
    Each mapping has 65 rows and 12 columns, the first four denotes feasible options, the middle four denotes observa
    ble alternatives and last four denotes the choices.

    Parameters
    ----------
    m : int
        the number of random mappings to generate.

    Returns
    -------
    A list of m random mappings.

    """
    # The last 4 positions on the mapping are the choices
    mapping_list = [] # A list of dataframes 65x12
    for i in range(0,m):
        mapping_predictions = []
        for j in range(0,len(self.choice_problems_list)):
            if deterministic == True:
                prediction_temp = tuple(self.choice_problems_list[j]) + (0, 0, 0, 0, 0, 0, 0, 0)
                prediction = list(prediction_temp)
                while sum(prediction[16:24]) < 1:
                    for h in range(0,8):
                        if prediction[h] == 1:
                            prediction[h+16] = randint(0,1)
            else:
                prediction_temp = tuple(self.choice_problems_list[j]) + (uniform(0, 1), uniform(0, 1),
                                                                          uniform(0, 1), uniform(0, 1),
                                                                          uniform(0, 1), uniform(0, 1),
                                                                          uniform(0, 1), uniform(0, 1))
                prediction = list(prediction_temp)
            mapping_predictions.append(prediction)
        mapping = pd.DataFrame(mapping_predictions, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
                                                            'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R',
                                                            'S', 'T', 'U', 'V', 'X', 'Y'])
        mapping_list.append(mapping)
    return(mapping_list)

def perf_round(self, x, gamma):
    """
    Round values greater than 0.54 to 1. Used to make probabilities of choice into choices for a neural network.

    Parameters
    ----------
    x : float
        value to be rounded.

    Returns
    -------
    Rounded number.

    """

    if x - int(x) >= gamma:
        x = int(x) + 1
    else:
        x = int(x)
    return(x)

def aspiration(self, T):
    """
    Get the index of the aspiration.

    Parameters
    ----------
    T : list
        a list of length 4 and elements 0 or 1.

    Returns
    -------
    i : int
        index of the aspiration.

    """
    for i in range(0,8):
        if T[i] == 1:
            return i

def u_S(self, S, T):
    a = self.aspiration(T)
    alternatives = [np.array((0,1)), np.array((0,0)), np.array((1,1)), np.array((1,0)),
                            np.array((2,1)), np.array((2,0)), np.array((3,1)), np.array((3,0))]
    dist = [float(np.linalg.norm(alternatives[a]-alternatives[x])) for x in range(0,8)]
    for i in range(0,len(S)):
        if S[i] == 0:
            dist[i] = 100
    #reference = dist.index(min(dist))
    references = list(np.where(np.array(dist) == min(dist))[0])
    dist_ref_list = []
    for v in range(0,len(references)):
        dist_ref = [float(np.linalg.norm(alternatives[references[v]]-alternatives[x])) for x in range(0,8)]
        dist_ref_list.append(dist_ref)
    for j in range(0,len(S)):
        k=0
        while k < len(references):
            if dist_ref_list[k][j] > float(np.linalg.norm(alternatives[a]-alternatives[references[0]])):
                if k == len(references)-1:
                    S[j] = 0
                k=k+1
            else:
                k = len(references)
    for y in range(0,8):
        if S[-y-1] == 1:
            choice = 7-y
            return(choice)

def u_S_no_ref(self, S, T):
    a = self.aspiration(T)
    alternatives = [np.array((0,1)), np.array((0,0)), np.array((1,1)), np.array((1,0)), np.array((2,1)),
                                              np.array((2,0)), np.array((3,1)), np.array((3,0))]
    dist = [float(np.linalg.norm(alternatives[a]-alternatives[x])) for x in range(0,8)]
    for i in range(0,len(S)):
        if S[i] == 0:
            dist[i] = 100
    references = list(np.where(np.array(dist) == min(dist))[0])
    return(references)

def f_rational(self):
    """
    Creates a mapping from the choice problems of the 4 alternatives framework into choices
    according to the rational model, without observable but non feasible options. For this setup,
    it coincides with the aspiration-based choice model.

    Returns
    -------
    A data frame corresponding to the rational model, for each choice problem.

    """
    naive_list = []
    for i in range(0,len(self.choice_problems_list)):
        S = self.choice_problems_list[i][0:8]
        choice_index = self.aspiration(S)
        temp = [0, 0, 0, 0, 0, 0, 0, 0]
        temp[choice_index] = 1
        naive_predict_temp = tuple(self.choice_problems_list[i]) + tuple(temp)
        naive_predict = list(naive_predict_temp)
        naive_list.append(naive_predict)
    naive = pd.DataFrame(naive_list, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
                                              'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q',
                                              'R', 'S', 'T', 'U', 'V', 'X', 'Y'])
    return(naive)

def f_naive_all(self):
    """
    Creates a mapping from the choice problems of the 4 alternatives framework into choices
    in such a way everything that is available to choose is chosen.

    Returns
    -------
    A data frame corresponding to the naive predictions, for each choice problem.

    """
    naive_list = []
    for i in range(0,len(self.choice_problems_list)):
        S = self.choice_problems_list[i][0:8]
        naive_predict_temp = tuple(self.choice_problems_list[i]) + tuple(S)
        naive_predict = list(naive_predict_temp)
        naive_list.append(naive_predict)
    naive = pd.DataFrame(naive_list, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
                                              'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'X', 'Y'])
    return(naive)

def f_naive_nothing(self):
    """
    Creates a mapping from the choice problems of the 4 alternatives framework into choices
    in such a way nothing is chosen.

    Returns
    -------
    A data frame corresponding to the naive predictions, for each choice problem.

    """
    naive_list = []
    for i in range(0,len(self.choice_problems_list)):
        naive_predict_temp = tuple(self.choice_problems_list[i]) + (0, 0, 0, 0, 0, 0, 0, 0)
        naive_predict = list(naive_predict_temp)
        naive_list.append(naive_predict)
    naive = pd.DataFrame(naive_list, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
                                              'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'X', 'Y'])
    return(naive)

def f_theta(self):
    #theta e o modelo do silva
    """
    Creates a mapping from the choice problems of the 4 alternatives framework into choices
    according to aspiration-based reference dependance model.

    Returns
    -------
    A data frame corresponding to aspiration-based reference dependance model's predictions
    for each choice problem.

    """
    model_list = []
    for i in range(0,len(self.choice_problems_list)):
        S = self.choice_problems_list[i][0:8]
        T = self.choice_problems_list[i][8:16]
        if S == T:
            choice_index = self.aspiration(T)
        else:
            choice_index = self.u_S(S, T)
        temp = [0, 0, 0, 0, 0, 0, 0, 0]
        temp[choice_index] = 1
        model_predict_temp = tuple(self.choice_problems_list[i]) + tuple(temp)
        model_predict = list(model_predict_temp)
        model_list.append(model_predict)
    model = pd.DataFrame(model_list, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
                                              'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'X', 'Y'])
    return(model)

def f_theta_no_ref(self):
    #theta_no_ref e o modelo do guney
    """
    Creates a mapping from the choice problems of the 4 alternatives framework into choices
    according to aspiration-based choice model.

    Returns
    -------
    A data frame corresponding to aspiration-based choice model's predictions, for each
    choice problem.

    """
    model_list = []
    for i in range(0,len(self.choice_problems_list)):
        S = self.choice_problems_list[i][0:8]
        T = self.choice_problems_list[i][8:16]
        if S == T:
            choice_index = [self.aspiration(T)]
        else:
            choice_index = self.u_S_no_ref(S, T)
        temp = [0, 0, 0, 0, 0, 0, 0, 0]
        for v in range(0,len(choice_index)):
            temp[choice_index[v]] = 1
        model_predict_temp = tuple(self.choice_problems_list[i]) + tuple(temp)
        model_predict = list(model_predict_temp)
        model_list.append(model_predict)
    model = pd.DataFrame(model_list, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
                                          'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'X', 'Y'])
    return(model)

def f_logit(self, data, deterministic = True):
    x_train = data[['I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'X', 'Y']]
    y_train = data[['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']]
    logit_model = keras.Sequential([
        keras.Input(shape=(16)),
        layers.Dense(8, activation = 'sigmoid')])
    logit_model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
    logit_model.fit(x_train, y_train, batch_size = 32, epochs = 100)

    predict_df = logit_model.predict(self.choice_problems)
    predict_df = pd.DataFrame(predict_df)

    if deterministic == True:
        #predict_df = predict_df.applymap(self.perf_round)
        for i in range(0,len(self.choice_problems)):
            for j in range(0,8):
                if self.choice_problems.iat[i,j] == 0:
                    predict_df.iat[i,j] = 0
                else:
                    predict_df.iat[i,j] = int(np.random.choice([0,1], 1, p = [1 - predict_df.iat[i,j], predict_df.iat[i,j]]))
    nn = pd.concat([self.choice_problems, predict_df], axis=1)
    return(nn)

def f_NN(self, data, N, deterministic = True):
    """
    Creates a mapping from the choice problems of the 4 alternatives framework into choices
    according to a neural network. By default, the neural net has two hidden layers, each
    with same amount of neurons.

    Parameters
    ----------
    data : pandas.DataFrame
        Data used to train the neural net.
    N : int
        Number of neurons for each hidden layer of the neural net.

    Returns
    -------
    A data frame corresponding to the treined neural net's predictions, for each choice problem.

    """
    x_train = data[['I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'X', 'Y']]
    y_train = data[['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']]
    one_hidden_layer_nn = keras.Sequential([
        keras.Input(shape=(16)),
        layers.Dense(N, activation = 'relu'),
        layers.Dense(N, activation = 'relu'),
        layers.Dense(8, activation = 'sigmoid')])
    one_hidden_layer_nn.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
    one_hidden_layer_nn.fit(x_train, y_train, batch_size = 32, epochs = 100)

    predict_df = one_hidden_layer_nn.predict(self.choice_problems)
    predict_df = pd.DataFrame(predict_df)

    if deterministic == True:
        #predict_df = predict_df.applymap(self.perf_round)
        for i in range(0,len(self.choice_problems)):
            for j in range(0,8):
                if self.choice_problems.iat[i,j] == 0:
                    predict_df.iat[i,j] = 0
                else:
                    predict_df.iat[i,j] = int(np.random.choice([0,1], 1, p = [1 - predict_df.iat[i,j], predict_df.iat[i,j]]))
    nn = pd.concat([self.choice_problems, predict_df], axis=1)
    return(one_hidden_layer_nn.summary())

def f_CNN(self, data, N, deterministic = True):
    x_train = data[['I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'X', 'Y']]
    x_train = np.array(x_train)
    x_train = x_train.reshape(len(x_train), 16, 1)
    y_train = data[['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']]
```

```
        #y_train = np.array(y_train)
        #y_train = y_train.reshape(len(y_train), 8, 1)
        convolutional_nn = keras.Sequential([
            layers.Conv1D(N, kernel_size=3, activation='relu', input_shape=(16, 1)),
            layers.Conv1D(N, kernel_size=3, activation='relu'),
            layers.Dropout(0.5),
            layers.MaxPooling1D(pool_size=2),
            layers.Flatten(),
            layers.Dense(8, activation = 'sigmoid')])
        convolutional_nn.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
        convolutional_nn.fit(x_train, y_train, batch_size = 32, epochs = 100)

        x_test = np.array(self.choice_problems)
        x_test = x_test.reshape(len(x_test), 16, 1)
        predict_df = convolutional_nn.predict(x_test)
        predict_df = pd.DataFrame(predict_df)

        if deterministic == True:
            #predict_df = predict_df.applymap(self.perf_round)
            for i in range(0,len(self.choice_problems)):
                for j in range(0,8):
                    if self.choice_problems.iat[i,j] == 0:
                        predict_df.iat[i,j] = 0
                    else:
                        predict_df.iat[i,j] = int(np.random.choice([0,1], 1, p = [1 - predict_df.iat[i,j], predict_df.iat[i,j]]))
        nn = pd.concat([self.choice_problems, predict_df], axis=1)
        return(convolutional_nn.summary())

    def model_fit(self, data, modelo):
        """
        Generate a model's prediction, choices, based on input data, choice problem.

        Parameters
        ----------
        data : pandas.DataFrame
            A data frame with n rows and 12 columns, with the last 8 being the choice problem, feasible and observable
            options).
        modelo : pandas.DataFrame
            A data frame, corresponding to a mapping or a model, with 12 columns, being the first eight the choice
            problem and the last 4 the choice correspondent to the choice problem.

        Returns
        -------
        A data frame containing the predictions of the model for the data input.

        """
        model_fit = pd.merge(data, modelo, how='left', left_on=['I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
                                                                  'Q', 'R', 'S', 'T', 'U', 'V', 'X', 'Y'],
                                                         right_on = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',
                                                                  'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P'])
        model_fit = model_fit.iloc[:,40:48]
        return(model_fit)

    def distance_map(self, f, F_Theta, deterministic = True):
        """
        Evaluates distance between mappings, according to the usual euclidean distance.

        Parameters
        ----------
        f : pandas.DataFrame
            Mapping from choice problems into choices, 65 rows and 12 columns.
        F_Theta : list
            A list containing the mappings corresponding to the parametrizations of the model, data frames with 65 rows
            and 12 columns. On the simplest case, it comprises a parametrization of interest and the naive mapping.
        Returns
        -------
        The minimum distance of model parametrizations and a given map.

        """
        dist_list = []
        #f_func = f[['I', 'J', 'K', 'L']]
        if deterministic == True:
            f_func = np.array(f)
            for i in range(0,len(F_Theta)):
                #parametrization_pred = F_Theta[i][['I', 'J', 'K', 'L']]
                parametrization_pred = np.array(F_Theta[i])
                dist = [np.linalg.norm(f_func[j]-parametrization_pred[j]) for j in range(0,len(f))]
                dist = np.array(dist)
                mean_dist = dist.mean()
                dist_list.append(mean_dist)
        else:
            f_func = f[['Q', 'R', 'S', 'T', 'U', 'V', 'X', 'Y']]
            f_func = np.array(f_func)
            for i in range(0,len(F_Theta)):
                parametrization_pred = F_Theta[i][[0, 1, 2, 3, 4, 5, 6, 7]]
                parametrization_pred = np.array(parametrization_pred)
                dist = [bce(f_func[j], parametrization_pred[j]).numpy() for j in range(0,len(f))]
                dist = np.array(dist)
                mean_dist = dist.mean()
                dist_list.append(mean_dist)
        dist_list = np.array(dist_list)
        error = dist_list.min()
        return(error)

    def best_map(self, data, det = True):
        """
        Iterates over 1000 maps to find the map that minimizes the expected loss function for the given data.

        Parameters
        ----------
        data : pandas.DataFrame
            Data in which the function iterates over random mappings to find the best one.

        Returns
        -------
        The best map for the given data.

        """
        if det == True:
            maps = self.mapping_generate(1000)
        else:
            maps = self.mapping_generate(1000, deterministic = det)
            """
            for k in range(0,len(maps)):
                maps[k][['I','J','K','L']] = maps[k][['I','J','K','L']].replace(0, 0.001)
                maps[k][['I','J','K','L']] = maps[k][['I','J','K','L']].replace(1, 0.999)
            """
        dist_list = []
        Y = data[['A', 'B', 'C', 'D']]
        Y = np.array(Y)
        for i in range(0,len(maps)):
            map_fit = self.model_fit(data, maps[i])
            map_fit = np.array(map_fit)
            if det == True:
                dist = [np.linalg.norm(Y[j] - map_fit[j]) for j in range(0,len(Y))]
            else:
                dist = [bce(Y[j], map_fit[j]).numpy() for j in range(0,len(Y))]
            dist = np.array(dist)
            mean_dist = dist.mean()
            dist_list.append(mean_dist)
        dist_list = np.array(dist_list)
        optimal_map = np.argmin(dist_list)
        return(maps[optimal_map])

    def mean_discrepancy(self, F_Theta, deterministic = True):
        """
        Evaluates the restrictiveness of a model, taking into account the respective naive mapping.

        Parameters
        ----------
        F_Theta : list
            A list containing data frames representing model's parametrizations mappings, with 65 rows and 12
            columns. By the defaut, the last element must be the naive mapping.

        Returns
        -------
        A numpy.array with the discrepancy for each random generated mapping.

        """
        delta_list = []
        #F_theta_predictions = []
        f_naive = []
        #Y = data[['A','B','C','D']]
        """
        for j in range(0,len(F_Theta)):
            predictions = self.model_fit(data, F_Theta[j])
            F_theta_predictions.append(predictions)
        """
        f_naive.append(F_Theta[-1])
        for i in range(0,len(self.mappings)):
            #mapping_fit = self.model_fit(data, self.mappings[i])
            if deterministic == True:
                delta = self.distance_map(self.mappings[i], F_Theta, deterministic = True)/self.distance_map(
                                                            self.mappings[i], f_naive, deterministic = True)
            else:
                delta = self.distance_map(self.mappings[i], F_Theta, deterministic = False)/self.distance_map(
                                                            self.mappings[i], f_naive, deterministic = False)
            delta_list.append(delta)
        delta_list = np.array(delta_list)
        return(delta_list)

    def mean_discrepancy_std_error(self, F_Theta, deterministic = True):
        """
        Evaluates the restrictiveness of a model, taking into account the respective naive mapping.

        Parameters
        ----------
        F_Theta : list
            A list containing data frames representing model's parametrizations mappings, with 65 rows and 12
            columns. By the defaut, the last element must be the naive mapping.

        Returns
        -------
        A numpy.array with the discrepancy for each random generated mapping.

        """
        delta_list = []
        #F_theta_predictions = []
        f_naive = []
        #Y = data[['A','B','C','D']]
        """
        for j in range(0,len(F_Theta)):
            predictions = self.model_fit(data, F_Theta[j])
            F_theta_predictions.append(predictions)
        """
        f_naive.append(F_Theta[-1])
        for i in range(0,len(self.mappings)):
            #mapping_fit = self.model_fit(data, self.mappings[i])
            if deterministic == True:
                delta = self.distance_map(self.mappings[i], F_Theta, deterministic = True)/self.distance_map(
                                                                    self.mappings[i], f_naive, deterministic = True)
            else:
                delta = self.distance_map(self.mappings[i], F_Theta, deterministic = False)/self.distance_map(
                                                                    self.mappings[i], f_naive, deterministic = False)
            delta_list.append(delta)
        delta_list = np.array(delta_list)
        delta_list = np.square(delta_list - delta_list.mean())
        ep = np.sqrt(delta_list.mean()/len(self.mappings))
        return(ep)

    def data_split(self, data, k):
        """
        Shuffles data and splits it into groups. The first k-1 groups have the same length. The last k
        group only has the same size as other groups if the division of rows of data by k is an integer.

        Parameters
        ----------
        data : pandas.DataFrame
            Data to be splitted.
        k : int
            Number of groups to be formed.

        Returns
        -------
        A list with k data frames.

        """
        group_size = int(len(data)/k)
        folds = []
        data = data.sample(frac=1).reset_index()
        del data['index']
        for i in range(0,k):
            if i < k-1:
                paper = data.iloc[i*group_size:(i+1)*group_size,]
                folds.append(paper)
            else:
                paper = data.iloc[i*group_size:,]
                folds.append(paper)
        return(folds)

    def k_fold(self, data_splitted, F, deterministic = True):
        """
        Perform the k-fold cross validation of a model, evaluating the error for each group of the data.

        Parameters
        ----------
        data_splitted : list
            A list of groups of data.
        F : list
            A list of mappings, may be random or a set of a model's parametrizations.

        Returns
        -------
        The mean error of the k groups.
        """
        group_size = len(data_splitted[0])
        error_list = []
        for i in range(0,len(data_splitted)):
            dist_list = []
            temp = [x for x in data_splitted if not(x.equals(data_splitted[i]))]
            test = data_splitted[i]
            Y_test = test[['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']]
            Y_test = np.array(Y_test)
            train = pd.concat(temp, ignore_index = True)
            Y_train = train[['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']]
            Y_train = np.array(Y_train)
            for j in range(0, len(F)):
                mapping_fit = self.model_fit(train, F[j])
                mapping_fit = np.array(mapping_fit)
                if deterministic == True:
                    dist = [np.linalg.norm(Y_train[k]-mapping_fit[k]) for k in range(0,len(Y_train))]
                else:
                    dist = [bce(Y_train[k], mapping_fit[k]) for k in range(0,len(Y_train))]
                dist = np.array(dist)
                dist_sum = dist.sum()
                dist_list.append(dist_sum)
            dist_list = np.array(dist_list)
            dist_list = (1/len(train))*dist_list
            f_k = np.argmin(dist_list)
            optimal_fit = self.model_fit(test, F[f_k])
            optimal_fit = np.array(optimal_fit)
            if deterministic == True:
                optimal_dist = [np.linalg.norm(Y_test[h]-optimal_fit[h]) for h in range(0,len(Y_test))]
            else:
                optimal_dist = [bce(Y_test[h], optimal_fit[h]) for h in range(0,len(Y_test))]
            optimal_dist = np.array(optimal_dist)
            optimal_dist_sum = optimal_dist.sum()
            error = (1/group_size)*optimal_dist_sum
            error_list.append(error)
        error_list = np.array(error_list)
        error_list_mean = error_list.mean()
        return(error_list_mean)

    def completeness_std_error(self, data_splitted, F, f_best, deterministic = True):
        """
        Perform the k-fold cross validation of a model, evaluating the error for each group of the data.

        Parameters
        ----------
        data_splitted : list
            A list of groups of data.
        F : list
            A list of mappings, may be random or a set of a model's parametrizations.

        Returns
        -------
        The mean error of the k groups.

        """
        group_size = len(data_splitted[0])
        var_list = []
        for i in range(0,len(data_splitted)):
            dist_list = []
            temp = [x for x in data_splitted if not(x.equals(data_splitted[i]))]
            test = data_splitted[i]
            Y_test = test[['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']]
            Y_test = np.array(Y_test)
            train = pd.concat(temp, ignore_index = True)
            Y_train = train[['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']]
            Y_train = np.array(Y_train)
            for j in range(0, len(F)):
                mapping_fit = self.model_fit(train, F[j])
                mapping_fit = np.array(mapping_fit)
                if deterministic == True:
                    dist = [np.linalg.norm(Y_train[k]-mapping_fit[k]) for k in range(0,len(Y_train))]
                else:
                    dist = [bce(Y_train[k], mapping_fit[k]) for k in range(0,len(Y_train))]
                dist = np.array(dist)
                dist_sum = dist.sum()
                dist_list.append(dist_sum)
            dist_list = np.array(dist_list)
            dist_list = (1/len(train))*dist_list
            f_k = np.argmin(dist_list)
            optimal_fit = self.model_fit(test, F[f_k])
            optimal_fit = np.array(optimal_fit)
            best_fit = self.model_fit(test, f_best)
            best_fit = np.array(best_fit)
            if deterministic == True:
                optimal_dist = [np.linalg.norm(Y_test[h]-optimal_fit[h]) for h in range(0,len(Y_test))]
                best_dist = [np.linalg.norm(Y_test[h]-best_fit[h]) for h in range(0,len(Y_test))]
            else:
                optimal_dist = [bce(Y_test[h], optimal_fit[h]) for h in range(0,len(Y_test))]
                best_dist = [bce(Y_test[h], best_fit[h]) for h in range(0,len(Y_test))]
            optimal_dist = np.array(optimal_dist)
            best_dist = np.array(best_dist)
            delta_z = optimal_dist - best_dist
            delta_z_sum = delta_z.sum()
            error = (1/group_size)*delta_z_sum
            delta_list = np.square(delta_z - error)
            var_sum = delta_list.sum()
            var = (1/(group_size-1))*var_sum
            var_list.append(var)
        var_list = np.array(var_list)
        var_list_sum = var_list.sum()
        f_star = []
        f_star.append(f_best)
        f_naive = []
        f_naive.append(F[-1])
        cv_best = self.k_fold(data_splitted, f_star, deterministic = deterministic)
        cv_f_naive = self.k_fold(data_splitted, f_naive, deterministic = deterministic)
        std_variation = np.sqrt(var_list_sum/(cv_f_naive-cv_best))
        ep = np.sqrt(std_variation/group_size)
        return(ep)

    def completeness(self, data_list, F_Theta, f_best, det = True):
        """
        Evaluates the completeness of a model, according to the naive mapping.

        Parameters
        ----------
        data_list : list
            A list of groups of data..
        F_Theta : list
            A list of model's parametrizations. By default, the last element must be the naive mapping.
        f_best : pandas.DataFrame
            A data frame corresponding to the best mapping for the data correspondent to data_list.

        Returns
        -------
        The completeness level of the model.

        """
```

```
        #data = pd.concat(data_list, ignore_index = True)
        f_star = []
        f_star.append(f_best)
        f_naive = []
        f_naive.append(F_Theta[-1])
        cv_best = self.k_fold(data_list, f_star, deterministic = det)
        cv_F_Theta = self.k_fold(data_list, F_Theta, deterministic = det)
        cv_f_naive = self.k_fold(data_list, f_naive, deterministic = det)
        #completeness_level = (cv_F_Theta - cv_mappings)/(cv_f_naive - cv_mappings)
        completeness_level = (cv_f_naive - cv_F_Theta)/(cv_f_naive - cv_best)
        return(completeness_level)

    def k_mse(self, data, F):
        Y_data = data[['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']]
        Y_data = np.array(Y_data)
        dist_list = []
        dist_modelo = []
        for j in range(0, len(F)):
            mapping_fit = self.model_fit(data, F[j])
            mapping_fit = np.array(mapping_fit)
            dist = [mse(Y_data[k], mapping_fit[k]) for k in range(0,len(Y_data))]
            dist = np.array(dist)
            dist_mean = dist.mean()
            dist_list.append(dist_mean)
        dist_best = dist_list[0]
        dist_modelo.append(dist_list[1])
        dist_modelo.append(dist_list[2])
        dist_naive = dist_list[-1]
        det_completeness = (dist_naive-min(dist_modelo))/(dist_naive-dist_best)
        return(det_completeness)

    def similarity(self, det_model, sto_model, n):
        restrictiveness_list = []
        threshold = np.arange(0.01,1.01,0.01)
        round_functions = []
        for i in range(0,len(threshold)):
            def pf(x, gamma = threshold[i]):
                if x - int(x) >= gamma:
                    x = int(x) + 1
                else:
                    x = int(x)
                return(x)
            round_functions.append(pf)
        sto_predict = sto_model[[0,1,2,3,4,5,6,7]]
        run_list = []
        for i in range(0,n):
            sto_temp = pd.DataFrame(np.zeros((len(self.choice_problems),8)))
            naive_tp = pd.DataFrame(np.zeros((len(self.choice_problems),8)))
            for k in range(0,len(self.choice_problems)):
                for j in range(0,8):
                    if self.choice_problems.iat[k,j] == 0:
                        sto_temp.iat[k,j] = 0
                        naive_tp.iat[k,j] = 0
                    else:
                        sto_temp.iat[k,j] = int(np.random.choice([0,1], 1, p = [1 - sto_predict.iat[k,j], sto_predict.iat[k,j]]))
                        naive_tp.iat[k,j] = sto_predict.iat[k,j]
            sto_temp = pd.concat([self.choice_problems, sto_temp], axis=1)
            run_list.append(sto_temp)
        for v in range(0,len(round_functions)):
            naive = naive_tp.applymap(round_functions[v])
            naive = pd.concat([self.choice_problems, naive], axis=1)
            delta_list = []
            f_naive = []
            f_naive.append(naive)
        for i in range(0,n):
            model_list = []
            model_list.append(run_list[i])
            model_list.append(naive)
            delta = self.distance_map(det_model, model_list, deterministic = True)/self.distance_map(
                                                                                     det_model, f_naive, deterministic = True)
            delta_list.append(delta)
        delta_list = np.array(delta_list)
        restrictiveness_list.append(delta_list.mean())
    threshold = pd.DataFrame(threshold)
    restrictiveness_list = pd.DataFrame(restrictiveness_list)
    graph = pd.concat([threshold, restrictiveness_list], axis=1, ignore_index=True)
    return(graph)

def curve(self, data, sto_model, n):
    Y_data = data[['A', 'B', 'C', 'D','E','F','G','H']]
    Y_data = np.array(Y_data)
    restrictiveness_list = []
    threshold = np.arange(0.01,1.01,0.01)
    round_functions = []
    for i in range(0,len(threshold)):
        def pf(x, gamma = threshold[i]):
            if x - int(x) >= gamma:
                x = int(x) + 1
            else:
                x = int(x)
            return(x)
        round_functions.append(pf)
    sto_predict = sto_model[[0,1,2,3,4,5,6,7]]
    run_list = []
    for i in range(0,n):
        sto_temp = pd.DataFrame(np.zeros((len(sto_model),8)))
        naive_tp = pd.DataFrame(np.zeros((len(sto_model),8)))
        for k in range(0,len(self.choice_problems)):
            for j in range(0,8):
                if self.choice_problems.iat[k,j] == 0:
                    sto_temp.iat[k,j] = 0
                    naive_tp.iat[k,j] = 0
                else:
                    sto_temp.iat[k,j] = int(np.random.choice([0,1], 1, p = [1 - sto_predict.iat[k,j], sto_predict.iat[k,j]]))
                    naive_tp.iat[k,j] = sto_predict.iat[k,j]
        sto_temp = pd.concat([self.choice_problems, sto_temp], axis=1)
        run_list.append(sto_temp)
    for v in range(0,len(round_functions)):
        naive = naive_tp.applymap(round_functions[v])
        naive = pd.concat([self.choice_problems, naive], axis=1)
        naive_fit = self.model_fit(data, naive)
        naive_fit = np.array(naive_fit)
        naive_dist = [np.linalg.norm(Y_data[h]-naive_fit[h]) for h in range(0,len(Y_data))]
        naive_dist = np.array(naive_dist)
        naive_error = naive_dist.mean()
        delta_list = []
        for i in range(0,n):
            realization_fit = self.model_fit(data, run_list[i])
            realization_fit = np.array(realization_fit)
            realization_dist = [np.linalg.norm(Y_data[h]-realization_fit[h]) for h in range(0,len(Y_data))]
            realization_dist = np.array(realization_dist)
            realization_error = realization_dist.mean()
            realization_error = min(realization_error,naive_error)
            kappa = (naive_error-realization_error)/naive_error
            delta_list.append(kappa)
        delta_list = np.array(delta_list)
        restrictiveness_list.append(delta_list.mean())
    threshold = pd.DataFrame(threshold)
```

```
restrictiveness_list = pd.DataFrame(restrictiveness_list)
graph = pd.concat([threshold, restrictiveness_list], axis=1, ignore_index=True)
return(graph)
```

A.2 Proofs

A.2.1 Proof of Theorem 1

Proof Suppose ρ_1 and ρ_2 satisfy Random Consistency and $S \cap T \neq \emptyset$, so that $\mu(T) > 0$. Fix any menu A and $p \in A$. Without loss of generality, we may assume that $A \subseteq \text{ri}\Delta(Z)$.[1] Let $E \subseteq \text{ri}\Delta(Z)$ be any sphere. Let C be the subset of E that includes only the maximizers of s for all $s \in S \setminus T$ and D be the subset of the maximizers of s for all $s \in T$. For each $\lambda \in (0,1)$, define $A_\lambda := C \cup (\lambda D + (1-\lambda)A)$. Let λ be large enough so that $\arg\max U(A_\lambda, s) \subseteq C$ for every $s \in S \setminus T$ and $\arg\max U_s(A_\lambda) \subseteq \lambda D + (1-\lambda)A$ for every $s \in T$.[2] Suppose $q \in \lambda D + (1-\lambda)A$ is such that $\rho_2^{A^\lambda}(q) > 0$ and fix $p \in \lambda D + (1-\lambda)A$ with $\rho_1^{A^\lambda}(p) > 0$.[3] We note that this implies that there exist unique $s, s' \in T$ with $p \in \arg\max U(A^\lambda, s)$ and $q \in \arg\max U(A^\lambda, s')$. Suppose now that either $\rho_1^{A^\lambda}(q) = 0$ or $\rho_2^{A^\lambda}(p) = 0$. By Random Consistency, there must exist a finite set $B \subseteq \Delta(Z)$ and $\delta > 0$ such that $\rho_1^{A^\lambda \cup B}(p) > 0$, but $\rho_2^{A^\lambda \cup B \cup \{p^\delta\}}(p^\delta) = 0$ for every $p^\delta \in B_\delta(p) \cap \Delta(Z)$. However, $\rho_2^{A^\lambda \cup B \cup \{p^\delta\}}(p^\delta) = 0$ for every $p^\delta \in B_\delta(p)$ can happen only if $\max_{r \in B} U(r,s) > U(p,s)$, which would imply that $\rho_1^{A^\lambda \cup B}(p) = 0$. We conclude that, for any $p \in \lambda D + (1-\lambda)A$, $\rho_1^{A^\lambda}(p) > 0$ if and only if $\rho_2^{A^\lambda}(p) > 0$. We note that this implies that $T \subseteq S$. Fix any $p, q \in \lambda D + (1-\lambda)A$ with $\rho_1^{A^\lambda}(p) > 0$ and $\rho_1^{A^\lambda}(q) > 0$. Assume, without loss of generality, that $\rho_1^{A^\lambda}(p)\rho_2^{A^\lambda}(q) \geq \rho_1^{A^\lambda}(q)\rho_2^{A^\lambda}(p)$. There must exist a unique $s \in T$ such that $p \in \arg\max_{r \in A^\lambda} U(r,s)$. But then, there exists a finite set B and $\delta > 0$ with $\rho_2^{A^\lambda \cup B \cup \{p^\delta\}}(p^\delta) = 0$ for every $p^\delta \in B_\delta(p)$ only if $\max_{r \in B} U(r,s) > U(p,s)$. This implies that $\rho_1^{A^\lambda \cup B}(p) = 0$, so that, by Random Consistency, we must have $\rho_1^{A^\lambda}(p)\rho_2^{A^\lambda}(q) = \rho_1^{A^\lambda}(q)\rho_2^{A^\lambda}(p)$. We conclude that, for

[1]Otherwise, just work with $\frac{1}{2}A + \frac{1}{2}\{p\}$, where p is any lottery with full support.

[2]Such λ must exist since, if we took $\lambda = 1$, then $A_\lambda = C \cup D$, and, for any $s \in S \cup T$, $|\arg\max U_s(A_\lambda)| = 1$. If $s \in S \setminus T$, then $\arg\max U_s(A_\lambda) \subseteq C$, and $\arg\max U_s(A_\lambda) \subseteq D$, otherwise. Therefore, for every $\lambda \in (0,1)$ close enough to 1, $\max U_s(C) > \max U_s(\lambda D + (1-\lambda)A)$, if $s \in S \setminus T$, and $\max U_s(C) < \max U_s(\lambda D + (1-\lambda)A)$, if $s \in T$.

Note that, in this context, if $a \in \arg\max U_s(A)$ and $q_s = \arg\max U_s(D)$ for some $s \in T$, then $U_s(\lambda q_s + (1-\lambda)a) > U_s(\lambda q + (1-\lambda)a)$, for any $q \in D \setminus \{q_s\}$, $U_s(\lambda q_s + (1-\lambda)a) \geq U_s(\lambda q_s + (1-\lambda)a')$ for any $a' \in A \setminus \{a\}$ and $U_s(\lambda q_s + (1-\lambda)a) = U_s(\lambda q_s + (1-\lambda)a')$ if, and only if, $\{a, a'\} \subseteq \arg\max U_s(A)$.

[3]Such a p is guaranteed to exist because $\mu(T) > 0$.

any $p, q \in \lambda D + (1-\lambda)A$ with $\rho_1^{A^\lambda}(q) > 0$, we must have

$$\frac{\rho_1^{A^\lambda}(p)}{\rho_1^{A^\lambda}(q)} = \frac{\rho_2^{A^\lambda}(p)}{\rho_2^{A^\lambda}(q)}.$$

Now note that

$$\sum_{p \in \lambda D + (1-\lambda)A} \rho_1(p) = \mu(T)$$

and

$$\sum_{p \in \lambda D + (1-\lambda)A} \rho_2(p) = \mu'(T) = 1.$$

But then, for any $p \in \lambda D + (1-\lambda)A$,

$$\begin{aligned} \frac{\rho_1^{A^\lambda}(p)}{\mu(T)} &= \frac{\rho_1^{A^\lambda}(p)}{\sum_{q \in \lambda D + (1-\lambda)A} \rho_1(q)} \\ &= \frac{\rho_2^{A^\lambda}(p)}{\sum_{q \in \lambda D + (1-\lambda)A} \rho_2(q)} \\ &= \rho_2^{A^\lambda}(p) \end{aligned}$$

And, therefore, for any $p \in A$,

$$\begin{aligned} \rho_2^A(p) &= \sum_{q \in D} \rho_2^{A^\lambda}(\lambda q + (1-\lambda)p) \\ &= \frac{1}{\mu(T)} \sum_{q \in D} \rho_1^{A^\lambda}(\lambda q + (1-\lambda)p) \\ &= \frac{1}{\mu(T)} \sum_{q \in D} \sum_{s \in S} \mu(s) \tau_s \left(\left\{u \in \mathcal{U} : \lambda q + (1-\lambda)p \in M(M(A^\lambda, U_s), u)\right\}\right) \\ &= \frac{1}{\mu(T)} \sum_{q \in D} \sum_{s \in T} \mu(s) \tau_s \left(\left\{u \in \mathcal{U} : \lambda q + (1-\lambda)p \in M(M(A^\lambda, U_s), u)\right\}\right) \\ &= \frac{1}{\mu(T)} \sum_{s \in T} \mu(s) \tau_s \left(\{u \in \mathcal{U} : p \in M(M(A, U_s), u)\}\right) \end{aligned}$$

This proves statement 2. Conversely, suppose there exists a set $T \subseteq S$, such that (T, U, μ_T, τ) is a FREU representation of ρ_2. Fix a menu $A \subseteq \text{ri}\Delta(Z)$ and $p, q \in A$ with $\rho_1^A(p)\rho_2^A(q) > \rho_1^A(q)\rho_2^A(p)$. For that to happen, we must have $\rho_2^A(q) > 0$, which also implies that $\rho_1^A(q) > 0$.

Therefore, the previous condition can be written as

$$\frac{\rho_1^A(p)}{\rho_1^A(q)} > \frac{\rho_2^A(p)}{\rho_2^A(q)}.$$

It is clear that this can happen only if there exists $s^* \in S \backslash T$ with $\tau_{s^*}\left(\{u \in \mathcal{U} : p \in M(M(A, U_{s^*}), u)\}\right) > 0$. Following the same steps as in the proof of the main result in Riella (2013), we can find a finite set B such that $\max_{q \in B} U(q, s) > \max_{q \in A} U(q, s)$ for every $s \in S \setminus \{s^*\}$, but $U(p, s^*) > \max_{q \in B} U(q, s^*)$. Let δ be small enough so that $\max_{q \in B} U(q, s) > U(p^\delta, s)$ for every $s \in S \setminus \{s^*\}$ and $p^\delta \in B_\delta(p)$. Note that this implies that $\rho_2^{A \cup B \cup \{p^\delta\}}(p^\delta) = 0$ for every $p^\delta \in B_\delta(p)$, but $\rho_1^{A \cup B}(p) > 0$. That is, ρ_1 and ρ_2 satisfy Random Consistency. Finally, if $S \cap T = \emptyset$, it is clear that $\rho_1^A(p)\rho_2^A(q) > \rho_1^A(q)\rho_2^A(p)$ implies that there exists $s^* \in S \setminus T$ with $\tau_{s^*}\left(\{u \in \mathcal{U} : p \in M(M(A, U_{s^*}), u)\}\right) > 0$. We may now follow the same steps as above to find a menu B and a $\delta > 0$ such that $\rho_2^{A \cup B \cup \{p^\delta\}}(p^\delta) = 0$ for every $p^\delta \in B_\delta(p)$, but $\rho_1^{A \cup B}(p) > 0$. Again, this shows that ρ_1 and ρ_2 satisfy Random Consistency.

A.2.2 Proof of Corollary 1

Proof Take some $p \in \text{ri}\Delta(Z)$ and $\epsilon > 0$ such that $\overline{B_\epsilon}(p) \cap \Delta(Z) \subset \text{ri}\Delta(Z)$ and define, for each $s \in S \cup T$, $q_s = \text{argmax}_{q \in \overline{B_\epsilon}(p)} U_s(q)$. Note that $q_s = q_{s'}$ implies that U_s and $U_{s'}$ represent the same vNM preference over lotteries. Take now $A := \{q_s \in \Delta(Z) : s \in S \cup T\}$. We must have that, if $supp(\rho_1^A) \cap supp(\rho_2^A) \neq \emptyset$, then $\text{argmax}_{q \in \overline{B_\epsilon}(p)} U_s(q) = \text{argmax}_{q \in \overline{B_\epsilon}(p)} U_{s'}(q)$ meaning that there are $s \in S$ and $s' \in T$ such that U_s and $U_{s'}$ represent the same vNM preferences, implying $s = s'$ and $s \in S \cap T$.

A.2.3 Proof of Lemma 1

Proof As stated in Section 4.2, $S_i \cap S_j = \{\emptyset\}$ means that for any $s \in S_i$ and $s' \in S_j$, U_s^i and $U_{s'}^j$ do not represent the same vNM preferences. Fix any arbitrary $i, j \in I$ with $i \neq j$.

[$\Longrightarrow$] Suppose $\{\rho_i, \rho_j\}$ satisfies Axiom 3. Take some $p \in$ ri$\Delta(Z)$ and $\epsilon > 0$ such that $\overline{B_\epsilon}(p) \cap \Delta(Z) \subset$ ri$\Delta(Z)$. Define $\hat{U} : S_i \cup S_j \times \Delta(Z) \mapsto \mathbb{R}$ as

$$\hat{U}_s := \begin{cases} U_s^i & \text{if } s \in S_i \\ U_s^j & \text{if } s \in S_j \setminus S_i \end{cases}.$$

For each $s \in S_i \cup S_j$, define $q_s = \text{argmax}_{q \in \overline{B_\epsilon}(p) \cap \Delta(Z)} \hat{U}_s(q)$ and let $A := \{q_s \in \Delta(Z) : s \in S_i \cup S_j\}$. Now choose some $q \in A$ such that $\rho_i^A(q) > 0$. By construction, there is an unique $s \in S_i$ with $\hat{U}_s(q) > \max_{p \in A \setminus \{q\}} \hat{U}_s(p)$ and $\hat{U}_{s'}(q) < \max_{p \in A \setminus \{q\}} \hat{U}_{s'}(p)$ for every $s' \in (S_i \cup S_j) \setminus s$. We must then have that, for some $\delta > 0$, $\rho_i^{(A \setminus \{q\}) \cup \{q^\delta\}}(q^\delta) > 0$ for every $q^\delta \in B_\delta(q) \cap \Delta(Z)$ which, by 3, implies the existence of some $D \in \mathcal{A}$ such that $\rho_i^{(A \setminus \{q\}) \cup D \cup \{q^\delta\}}(q^\delta) > 0$, but $\rho_j^{(A \setminus \{q\}) \cup D \cup \{q^\delta\}}(q^\delta) = 0$, for every $q^\delta \in B_\delta(q) \cap \Delta(Z)$. For $\rho_j^{(A \setminus \{q\}) \cup D \cup \{q^\delta\}}(q^\delta) = 0$ to be true for every $q^\delta \in B_\delta(q) \cap \Delta(Z)$, it must be the case that $\hat{U}_{s'}(q) < \max_{p \in (A \setminus \{q\}) \cup D} \hat{U}_{s'}(p)$ for every $s' \in S_j$, which can only happen if $s \notin S_j$. Since we took $s \in S_i$ arbitrarily, we must have that $S_i \cap S_j = \{\emptyset\}$.

[$\Longleftarrow$] Suppose that ρ_i and ρ_j have FREU representations such that $S_i \cap S_j = \{\emptyset\}$. Take some $A \in \mathcal{A}$, $p \in$ ri$\Delta(Z)$ and $\delta > 0$ such that $\rho_i^{A \cup \{p^\delta\}}(p^\delta) > 0$ for every $p^\delta \in B_\delta(p) \cap \Delta(Z)$. Fix some $s \in S_i$ such that, $\mu^i(s)\tau_s^i(\{u \in \mathcal{U} : p \in M(M(A, U_s^i), u)\}) > 0$. Let $\hat{v}$ be the vector in $\mathbb{R}^Z$ such that $\hat{v} \cdot q = U_s^i(q)$ for every $q \in \Delta(Z)$ and

$$v := \hat{v} - \left(\frac{1}{|Z|} \sum_{z \in Z} \hat{v}_z \right) \mathbf{1},$$

where $\mathbf{1} = (1, \ldots, 1)$ is the unit vector of size $|Z|$. Take $\epsilon > 0$ so that $\overline{B_{d(p,p-\epsilon v)}}(p - \epsilon v) \cap$ Span$(\Delta(Z)) \subset \Delta(Z)$. Now, for each $s' \in S_j$, let $q_{s'} := \arg\max_{q \in \overline{B_{d(p,p-\epsilon v)}}(p-\epsilon v)} U_{s'}^j(q)$ and $D := \{q_{s'} : s' \in S^j\}$. Note that, since $s \notin S_j$, $p \notin D$, $U_s^i(p) > \max_{q \in D} U_s^i(q)$ and, for each $s' \in S_j$, $U_{s'}^j(p) < U_{s'}^j(q_{s'})$. Therefore, choosing $\delta' \in (0, \delta]$ small enough, we must have that $\rho_i^{A \cup D \cup \{p^{\delta'}\}}(p^{\delta'}) > 0$, but $\rho_j^{A \cup D \cup \{p^{\delta'}\}}(p^{\delta'}) = 0$, for every $p^{\delta'} \in B_{\delta'}(p) \cap \Delta(Z)$, proving that Axiom 3 is satisfied.

A.2.4 Proof of Proposition 1

Proof [$\Longrightarrow$] Suppose the collection $\{\rho_i\}_{i\in I}$ satisfies Axioms 3 and 4. Since, for each $i \in I$, ρ and ρ_i satisfy Random Consistency and Axiom 2, Theorem 1 implies that $S_i \subset S$, $\mu_i = \mu_{S_i}$ and, for each $s \in S_i$, $\tau_s = \tau_s^i$. Therefore, for each $i \in I$, $(S_i, U, \mu_{S_i}, \tau)$ is a FREU representation of ρ_i. Since the collection $\{\rho_i\}_{i\in I}$ satisfies Axiom 3, Lemma 1 implies that, for each $i, j \in I, i \neq j$, $S_i \cap S_j = \{\emptyset\}$. It remains for us to show that $S \subseteq \bigcup_{i\in I} S_i$. To see that take some $p \in$ ri$\Delta(Z)$ and $\epsilon > 0$ such that $\overline{B_\epsilon}(p) \cap \Delta(Z) \subset$ ri$\Delta(Z)$ and, as we did in the proof of Lemma 1, for each $s \in S$, define $q_s =$ argmax$_{q\in\overline{B_\epsilon}(p)\cap\Delta(Z)} U_s(q)$ and let $A := \{q_s \in \Delta(Z) : s \in S\}$. Now, suppose there is $s' \in S \setminus \bigcup_{i\in I} S_i$. But then we should have $q_{s'} \in$ supp$(\rho^A) \setminus \bigcup_{i\in I}$ supp(ρ_i^A), which contradicts Axiom 4.

[$\Longleftarrow$] Conversely, suppose $\{S_i\}_{i\in I}$ is a partition of S. Axiom 4 is an immediate consequence of this fact, and Lemma 1 implies that Axiom 3 holds.

A.2.5 Proof of Proposition 2

Proof Let (S, U', μ') be a DLR representation of $\succsim$ and $(T, \hat{U}, \hat{\mu}, \tau)$ a representation of ρ. The second part of the Proposition 2 in Ahn (2013) assures us that ρ and $\succsim$ satisfy Axiom 5 if, and only if, $T \subseteq S$. This means that, for each $t \in T$ there is an unique $s \in S$ such that $\hat{U}_t$ and U'_s represent the same vNM preference over $\Delta(Z)$. Therefore, the necessity of Axiom 5 follows directly. It remains for us to show that there is a state dependent utility function U and a probability distribution over states μ such that (S, U, μ) is a DLR representation of $\succsim$ and (T, U, μ_T, τ) is a FREU representation of ρ.

For that, define

$$\mu(s) := \begin{cases} \mu'(T)\hat{\mu}(s) & \text{if } s \in T \\ \mu'(s) & \text{if } s \in S \setminus T \end{cases},$$

and

$$U_s := \frac{\mu'(s)}{\mu(s)} U'_s.$$

Notice this implies that, for any $A \in \mathcal{A}$ and $p \in A$,

$$\begin{aligned}\sum_{s\in S}\mu(s)\max_{p\in A}U_s(p) &= \sum_{s\in S\setminus T}\mu'(s)\max_{p\in A}\left[\frac{\mu'(s)}{\mu'(s)}U'_s(p)\right]+\sum_{s\in T}\mu'(T)\hat{\mu}(s)\max_{p\in A}\left[\frac{\mu'(s)}{\mu'(T)\hat{\mu}(s)}U'_s(p)\right]\\ &= \sum_{s\in S}\mu'(s)\max_{p\in A}U'_s(p),\end{aligned}$$

and

$$\begin{aligned}\rho^A(p) &= \sum_{s\in T}\hat{\mu}(s)\tau_s\left(\left\{u\in\mathcal{U}: p\in M(M(A,\hat{U}_s),u)\right\}\right)\\ &= \sum_{s\in T}\frac{\mu(s)}{\mu'(T)}\tau_s\left(\{u\in\mathcal{U}: p\in M(M(A,U_s),u)\}\right)\\ &= \sum_{s\in T}\mu_T(s)\tau_s\left(\{u\in\mathcal{U}: p\in M(M(A,U_s),u)\}\right).\end{aligned}$$

Therefore we have that (S,U,μ) is a DLR representation of $\succsim$ and (T,U,μ_T,τ) is a representation of ρ.

A.2.6 Proof of Proposition 3

Proof [$\implies$] Let (S,U',μ') be any DLR representation for $\succsim$ and for each ρ_i let $(S_i,\hat{U}^i,\hat{\mu}^i,\tau^i)$ be its FREU representation. By Lemma 1, we know that for each $i,j\in I$, with $i\neq j$, $S_i\cap S_j=\{\emptyset\}$. Fix some $i\in I$. Axiom 6 implies that, if there is some $\delta>0$ such that, $\rho_i^{D\cup\{p^\delta\}}(p^\delta)>0$ for every $p^\delta\in B_\delta(p)\cap\Delta(Z)$ and $D\in\mathcal{A}$ with $d_h(A,D)<\delta$, then we must have $A\cup\{p\}\succ A$, meaning that $\succsim$ and ρ_i satisfy Axiom 5. Therefore, by Proposition 2, for each $i\in I$, we have that $S_i\subseteq S$. Now, similarly to what we did in the proof of Proposition 1, take some $p\in$ ri$\Delta(Z)$ and $\epsilon>0$ such that $\overline{B_\epsilon}(p)\cap\Delta(Z)\subset$ ri$\Delta(Z)$ and for each $s\in S$, define $q'_s=$ argmax$_{q\in\overline{B_\epsilon}(p)\cap\Delta(Z)}U'_s(q)$ and, for each $i\in I$ and $\hat{s}\in S_i$, $q^i_{\hat{s}}=$ argmax$_{q\in\overline{B_\epsilon}(p)\cap\Delta(Z)}\hat{U}_{\hat{s}}(q)$. Let $A':=\{q_{s'}\in\Delta(Z): s'\in S\}$ and $\hat{A}:=\{q_{\hat{s}}\in\Delta(Z):\hat{s}\in S_i, i\in I\}$. Since $S_i\subseteq S$ for every $i\in I$, we must have that $\hat{A}\subseteq A'$. If there was some $s\in S\setminus\bigcup_{i\in I}S_i$, we should have that $q'_s\notin\hat{A}$, $\hat{U}^i_{\hat{s}}(q'_s)<\max_{q\in\hat{A}}\hat{U}^i_{\hat{s}}(q)$ and, for some $\delta>0$ small enough, for every $i\in I$, $\rho_i^{D\cup\{p^\delta\}}(p^\delta)=0$ for every $p^\delta\in B_\delta(q'_s)\cap\Delta(Z)$, $D\in\mathcal{A}$ with $d_h(A',D)<\delta$. This contradicts Axiom 6, since $\{q'_s\}=\arg\max_{q\in A'}U'_s(q)$, implying $A'\succ A'\setminus\{q'_s\}$. Therefore, we must have $S=\bigcup_{i\in I}S_i$ and $\{S_i\}_{i\in I}$ is a partition of S.

Now define $\mu := \sum_{i\in I} \mu'(S_i)\hat{\mu}^i$ and $U_s := \frac{\mu'(s)}{\mu(s)}U'_s$, for every $s \in S$, and note that, for any $A \in \mathcal{A}$ and $s \in \hat{S}$,

$$\begin{aligned}\mu'(s)\max_{p\in A} U'_s(p) &= \frac{\sum_{i\in I}\mu'(S'_i)\hat{\mu}^i(s)}{\sum_{i\in I}\mu'(S'_i)\hat{\mu}^i(s)}\mu'(s)\max_{p\in A} U'_s(p)\\ &= \sum_{i\in I}\mu'(S'_i)\hat{\mu}^i(s)\max_{p\in A}\frac{\mu'(s)}{\sum_{i\in I}\mu'(S'_i)\hat{\mu}^i(s)}U'_s(p)\\ &= \mu(s)\max_{p\in A} U_s(p),\end{aligned}$$

meaning that (S, U, μ) is a DLR representation for $\succsim$. Since, for each $i \in I$ and $s \in S$, $\mu_{S_i}(s) = \hat{\mu}^i(s)$ and $\hat{U}^i_s$ and U_s represent the same vNM preferences on $\Delta(Z)$, we have that, for each $i \in I$, $(S_i, U, \mu_{S_i}, \tau)$ is a FREU representation of ρ_i.

[$\Longleftarrow$] Suppose now $\succsim$ has a DLR representation (S, U, μ) such that for each $i \in I$, $(S_i, U, \mu_{S_i}, \tau)$ is a FREU representation of ρ_i and $\{S_i\}_{i\in I}$ is a partition of S. Lemma 1 assures us that Axiom 3 is satisfied, so we only need to show that Axiom 6 also holds. Fix some arbitrary $A \in \mathcal{A}$, $p \in \Delta(Z)$ and suppose $A \cup \{p\} \succ A$, this implies that, for some $s \in S$, $U_s(p) > \max_{a\in A} U_s(a)$. Since A is finite and U_s is continuous, we must have that there is $\delta > 0$ such that, for every $p^\delta \in B_\delta(p)$, $U_s(p^\delta) > \max_{a\in A} U_s(a)$. Since $s \in S_i$ for some $i \in I$ and, we must have $\mu_{S_i}(s) > 0$ and $\rho_i^{A\cup\{p^\delta\}}(p^\delta) > 0$, for every $p^\delta \in B_\delta(p)$.

A.2.7 Proof of Corollary 2

Proof Let (S, U, μ, τ) be a DLR-GP representation of $(\succsim_1, \rho_1)$ and (T, U', μ', τ') a DLR-GP representation of $(\succsim_2, \rho_2)$. Since ρ_1 and ρ_2 satisfy Random Consistency and Axiom 2, Theorem 1 and 1 imply that, for some $T \subseteq S$, (T, U, μ_T, τ) is a FREU representation of ρ_2 and, consequently, $\mu_T = \mu'$, $\tau = \tau'$ and, for each $t \in T$, U_t and U'_t represent the same vNM preferences over $\Delta(Z)$.

(i) For any menu $A \in \mathcal{A}$ and $q \in \Delta(Z)$, $A \cup \{q\} \succ_2 A$ implies there is some $t \in T$ such that $\{q\} = M(A \cup \{q\}, U'_t) = M(A \cup \{q\}, U_t)$. Since $t \in S$ and $\mu(t) > 0$ by definition, this implies $\rho_1^{A\cup\{q\}}(q) > 0$.

(ii) Take any menu $A \in \mathcal{A}$ and lottery $p \in \Delta(Z) \setminus A$ such that, for some $\delta > 0$, we have $\rho_2^{D\cup\{p^\delta\}}(p^\delta) > 0$ for every $p^\delta \in B_\delta(p) \cap \Delta(Z)$ and $D \in \mathcal{A}$ with $d_h(A, D) < \delta$. It must be the case that, for some $t \in T$, $\{p\} = M(A \cup \{p\}, U_t)$. Since $t \in S$, this implies

$$\sum_{s \in S} \mu(s) \max_{q \in A \cup \{p\}} U_s(q) > \sum_{s \in S} \mu(s) \max_{q \in A} U_s(q),$$

and, therefore, $A \cup \{p\} \succ_1 A$.

References

AGRANOV, M.; ORTOLEVA, P. Stochastic choice and preferences for randomization. *Journal of Political Economy*, University of Chicago Press, v. 125, n. 1, p. 40–68, feb 2017.

AGRAWAL, R.; IMIELIŃSKI, T.; SWAMI, A. Mining association rules between sets of items in large databases. In: *Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93*. [S.l.]: ACM Press, 1993.

AHN, T. S. D. S. Preference for flexibility and random choice. *Econometrica*, The Econometric Society, v. 81, n. 1, p. 341–361, 2013.

AHUJA, K.; CHOUDHURY, M.; DANDAPAT, S. *On the Economics of Multilingual Few-shot Learning: Modeling the Cost-Performance Trade-offs of Machine Translated and Manual Data*. [S.l.]: arXiv, 2022.

AKAIKE, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, Institute of Electrical and Electronics Engineers (IEEE), v. 19, n. 6, p. 716–723, dec 1974.

ALGABA, A. et al. ECONOMETRICS MEETS SENTIMENT: AN OVERVIEW OF METHODOLOGY AND APPLICATIONS. *Journal of Economic Surveys*, Wiley, v. 34, n. 3, p. 512–547, may 2020.

ALWOSHEEL, A.; CRANENBURGH, S. van; CHORUS, C. G. Why did you predict that? towards explainable artificial neural networks for travel demand analysis. *Transportation Research Part C: Emerging Technologies*, Elsevier BV, v. 128, p. 103143, jul 2021.

ANDINI, M. et al. Targeting with machine learning: An application to a tax rebate program in italy. *Journal of Economic Behavior &amp Organization*, Elsevier BV, v. 156, p. 86–102, dec 2018.

ANDREWS, I. et al. *The Transfer Performance of Economic Models*. [S.l.]: arXiv, 2022.

APESTEGUIA, J.; BALLESTER, M. A. Separating predicted randomness from residual behavior. *Journal of the European Economic Association*, Oxford University Press (OUP), v. 19, n. 2, p. 1041–1076, may 2020.

ARCIDIACONO, P.; MILLER, R. A. *Identifying dynamic discrete choice models off short panels*. 2013.

AVOYAN, A. et al. Planned vs. actual attention. 2022.

BARBERÀ, S. et al. Order-k rationality. *Economic Theory*, Springer Science and Business Media LLC, v. 73, n. 4, p. 1135–1153, mar 2021.

BARNICHON, R.; GARDA, P. Forecasting unemployment across countries: The ins and outs. *European Economic Review*, Elsevier BV, v. 84, p. 165–183, may 2016.

BOOTH, W. J. *Households*: on the moral architecture of the economy. [S.l.]: Cornell University Press, 1993. 305 p. ISBN 0801427916.

BOX, G.; JENKINS, G. Time series analysis: Forecasting and control. *Holden-Day, San*

Francisco, 1970.
BOX, G. E. P. Science and statistics. *Journal of the American Statistical Association*, Informa UK Limited, v. 71, n. 356, p. 791–799, dec 1976.
BRATHWAITE, T.; VIJ, A.; WALKER, J. L. *Machine Learning Meets Microeconomics: The Case of Decision Trees and Discrete Choice*. [S.l.]: arXiv, 2017.
BRäUNING, M. et al. Lexicographic preferences for predictive modeling of human decision making: A new machine learning method with an application in accounting. *European Journal of Operational Research*, Elsevier BV, v. 258, n. 1, p. 295–306, apr 2017.
BUSA-FEKETE, R.; HüLLERMEIER, E.; SZORENYI, B. Preference-based rank elicitation using statistical models: The case of mallows. *ICML*, 2014.
CHARPENTIER, A.; FLACHAIRE, E.; LY, A. Econometrics and machine learning. *Economie et Statistique / Economics and Statistics*, Institut National de la Statistique et des Etudes Economiques (INSEE), n. 505d, p. 147–169, apr 2019.
CLIPPEL, G. de; ROZEN, K. Bounded rationality in choice theory: A survey. 2022.
COHEN, M. D.; AXELROD, R. Coping with complexity: The adaptive value of changing utility. *American Economic Review*, v. 74, n. 1, p. 30–42, mar. 1984.
COSTA-GOMES, M. A. et al. Choice, deferral, and consistency. *Quantitative Economics*, The Econometric Society, v. 13, n. 3, p. 1297–1318, 2022.
CRANENBURGH, S. van et al. Choice modelling in the age of machine learning - discussion paper. *Journal of Choice Modelling*, Elsevier BV, v. 42, p. 100340, mar 2022.
DEFAYS, D. An efficient algorithm for a complete link method. *The Computer Journal*, Oxford University Press (OUP), v. 20, n. 4, p. 364–366, apr 1977.
DEKEL, E.; LIPMAN, B. L.; RUSTICHINI, A. Representing preferences with a unique subjective state space. *Econometrica*, The Econometric Society, v. 69, n. 4, p. 891–934, jul 2001.
DENG, Y. et al. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, Institute of Electrical and Electronics Engineers (IEEE), v. 28, n. 3, p. 653–664, mar 2017.
DONALDSON, D.; STOREYGARD, A. The view from above: Applications of satellite data in economics. *Journal of Economic Perspectives*, American Economic Association, v. 30, n. 4, p. 171–198, nov 2016.
DONGDONG, W. The consumer price index forecast based on ARIMA model. In: *2010 WASE International Conference on Information Engineering*. [S.l.]: IEEE, 2010.
DOSHI-VELEZ, F.; KIM, B. *Towards A Rigorous Science of Interpretable Machine Learning*. [S.l.]: arXiv, 2017.
DRITSAKIS, N.; KLAZOGLOU, P. Forecasting unemployment rates in usa using box-jenkins methodology. *International Journal of Economics and Financial Issues*, v. 8, n. 1, p. 9–20, 2018.
ELIAZ, K.; OK, E. A. Indifference or indecisiveness? choice-theoretic foundations of incomplete preferences. *Games and Economic Behavior*, Elsevier BV, v. 56, n. 1, p. 61–86, jul 2006.
ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. *United States*, dez. 1996.
EVREN, O.; OK, E. A. On the multi-utility representation of preference relations. *Journal of Mathematical Economics*, Elsevier BV, v. 47, n. 4-5, p. 554–563, aug 2011.

FRISCH, R. "editorial". *Econometrica, 1, p. 2*, 1933.
FUDENBERG, D.; GAO, W.; LIANG, A. *How Flexible is that Functional Form? Quantifying the Restrictiveness of Theories*. [S.l.]: arXiv, 2020.
FUDENBERG, D.; LIANG, A. Machine learning for evaluating and improving theories. *ACM SIGecom Exchanges*, Association for Computing Machinery (ACM), v. 18, n. 1, p. 4–11, dec 2020.
GARCÍA-GARCÍA, J. C. et al. A comparative study of machine learning, deep neural networks and random utility maximization models for travel mode choice modelling. *Transportation Research Procedia*, Elsevier BV, v. 62, p. 374–382, 2022.
GILBOA, I. et al. ECONOMICS: BETWEEN PREDICTION AND CRITICISM. *International Economic Review*, Wiley, v. 59, n. 2, p. 367–390, apr 2018.
GILBOA, I. et al. Economic theories and their dueling interpretations. 2022.
GILBOA, I. et al. Economic theory:economics, methods and methodology. 2022.
GILBOA, I.; SAMUELSON, L. What were you thinking? decision theory as coherence test. *Theoretical Economics*, The Econometric Society, v. 17, n. 2, p. 507–519, 2022.
GILPIN, L. H. et al. Explaining explanations: An overview of interpretability of machine learning. In: *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. [S.l.]: IEEE, 2018.
GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016.
GUL, F.; NATENZON, P.; PESENDORFER, W. Random choice as behavioral optimization. *Econometrica*, The Econometric Society, v. 82, n. 5, p. 1873–1912, 2014.
GUL, F.; PESENDORFER, W. Random expected utility. *Econometrica*, The Econometric Society, v. 74, n. 1, p. 121–146, jan 2006.
GUNEY, B.; RICHTER, M.; TSUR, M. Aspiration-based choice. *Journal of Economic Theory*, v. 176, n. C, p. 935–956, 2018.
HANSEN, S.; MCMAHON, M.; PRAT, A. Transparency and deliberation within the FOMC: A computational linguistics approach. *The Quarterly Journal of Economics*, Oxford University Press, v. 133, n. 2, p. 801–870, oct 2017.
HAUSMAN, J. A. Specification and estimation of simultaneous equation models. In: ____. [S.l.]: Elsevier BV, 1983. (Handbook of Econometrics, v. 1), cap. 7, p. 391–448.
HEBB, D. O. The organization of behavior: A neuropsychological theory. new york: John wiley and sons. *Science Education*, Wiley, v. 34, n. 5, p. 336–337, dec 1950.
HENDERSON, J. V.; STOREYGARD, A.; WEIL, D. N. Measuring economic growth from outer space. *American Economic Review*, American Economic Association, v. 102, n. 2, p. 994–1028, apr 2012.
HENDRY, D. F. The methodology of empirical econometric modeling: Applied econometrics through the looking-glass. In: *Palgrave Handbook of Econometrics*. [S.l.]: Palgrave Macmillan UK, 2009. p. 3–67.
HIGHHOUSE, S. Context-dependent selection: The effects of decoy and phantom job candidates. *Organizational Behavior and Human Decision Processes*, Elsevier BV, v. 65, n. 1, p. 68–76, jan 1996.
HILLEL, T. et al. A systematic review of machine learning classification methodologies for modelling passenger mode choice. *Journal of Choice Modelling*, Elsevier BV, v. 38, p. 100221, mar 2021.
HIRSCHBERG, J.; MANNING, C. D. Advances in natural language processing. *Science*,

American Association for the Advancement of Science (AAAS), v. 349, n. 6245, p. 261–266, jul 2015.

HOLLAND, J. H.; MILLER, J. H. Artificial adaptative agents n economic theory. *The American Economic Review*, v. 81, n. 2, p. 365–370, maio 1991.

HRNJIC, E.; TOMCZAK, N. *Machine learning and behavioral economics for personalized choice architecture*. [S.l.]: arXiv, 2019.

HUANG, C.-L.; CHEN, M.-C.; WANG, C.-J. Credit scoring with a data mining approach based on support vector machines. *Expert Systems with Applications*, Elsevier BV, v. 33, n. 4, p. 847–856, nov 2007.

HüYüK, A.; ZAME, W. R.; SCHAAR, M. van der. Inferring lexicographically-ordered rewards from preferences. *Proceedings of the AAAI Conference on Artificial Intelligence*, Association for the Advancement of Artificial Intelligence (AAAI), v. 36, n. 5, p. 5737–5745, jun 2022.

JEZIORSKI, P.; KRASNOKUTSKAYA, E.; CECCARINI, O. Adverse selection and moral hazard in a dynamic model of auto insurance. In: . [S.l.: s.n.], 2017.

KAHNEMAN, D. Maps of bounded rationality: Psychology for behavioral economics. *American Economic Review*, American Economic Association, v. 93, n. 5, p. 1449–1475, nov 2003.

KANG POLINA KUZNETSOVA, M. L. J. S.; YEJIN, C. Where not to eat? improving public policy by predicting hygiene inspections using online reviews. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. Stroudsburg, 2013. (Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing), p. 1443–48.

KARNI, E.; VIERØ, M.-L. "reverse bayesianism": A choice-based theory of growing awareness. *American Economic Review*, American Economic Association, v. 103, n. 7, p. 2790–2810, dec 2013.

KEANE, M. P. Structural vs. atheoretic approaches to econometrics. *Journal of Econometrics*, Elsevier BV, v. 156, n. 1, p. 3–20, May 2010.

KEANE, M. P.; TODD, P. E.; WOLPIN, K. I. The structural estimation of behavioral models: Discrete choice dynamic programming methods and applications. In: *Handbook of Labor Economics*. [S.l.]: Elsevier, 2011. p. 331–461.

KEANE, M. P.; WOLPIN, K. I. The career decisions of young men. *Journal of political Economy*, The University of Chicago Press, v. 105, n. 3, p. 473–522, 1997.

KERSTING, K. Machine learning and artificial intelligence: Two fellow travelers on the quest for intelligent behavior in machines. *Frontiers in Big Data*, Frontiers Media SA, v. 1, nov 2018.

KHARIMAH F., U. M. W. W. . E. F. A. M. Time series modeling and forecasting of the consumer price index bandar lampung. *Science International Lahore*, v. 27, n. 5, p. 4619–4624, 2015.

KIM, K. Financial time series forecasting using support vector machines. *Neurocomputing*, Elsevier BV, v. 55, n. 1-2, p. 307–319, sep 2003.

KIRIAKIDIS, M.; KARGAS, A. Greek GDP forecast estimates. *Applied Economics Letters*, Informa UK Limited, v. 20, n. 8, p. 767–772, may 2013.

KLEINBERG, J. et al. Human decisions and machine predictions. *The Quarterly Journal of Economics*, Oxford University Press (OUP), aug 2017.

KLEINBERG, J. et al. Prediction policy problems. *American Economic Review*, American Economic Association, v. 105, n. 5, p. 491–495, may 2015.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Springer Science and Business Media LLC, v. 521, n. 7553, p. 436–444, may 2015.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Institute of Electrical and Electronics Engineers (IEEE), v. 86, n. 11, p. 2278–2324, 1998.

LEE, D.; DERRIBLE, S.; PEREIRA, F. C. Comparison of four types of artificial neural network and a multinomial logit model for travel mode choice modeling. *Transportation Research Record: Journal of the Transportation Research Board*, SAGE Publications, v. 2672, n. 49, p. 101–112, sep 2018.

LESHEM, D. Retrospectives: What did the ancient greeks mean by ioikonomia/i? *Journal of Economic Perspectives*, American Economic Association, v. 30, n. 1, p. 225–238, feb 2016.

LUCE, R. D. Semiorders and a theory of utility discrimination. *Econometrica*, JSTOR, v. 24, n. 2, p. 178, apr 1956.

LUCE, R. D. Individual choice behavior: A theoretical analysis. *Journal of the Royal Statistical Society. Series A (General)*, JSTOR, v. 123, n. 4, p. 486, 1960.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, v. 1, p. 281–297, 1967.

MALLOWS, C. L. Non-null ranking models. *Biometrika*, Oxford University Press (OUP), v. 44, n. 1-2, p. 114–130, 1957.

MANSKI, C. F. The structure of random utility models. *Theory and Decision*, Springer Science and Business Media LLC, v. 8, n. 3, p. 229–254, jul 1977.

MANZINI, P.; MARIOTTI, M. CATEGORIZE THEN CHOOSE: BOUNDEDLY RATIONAL CHOICE AND WELFARE. *Journal of the European Economic Association*, Oxford University Press (OUP), v. 10, n. 5, p. 1141–1165, jun 2012.

MANZINI, P.; MARIOTTI, M. Choice by lexicographic semiorders. *Theoretical Economics*, The Econometric Society, v. 7, n. 1, p. 1–23, jan 2012.

MARSCHAK, J. Binary-choice constraints and random utility indicators. In: *Cowles Foundation Discussion Papers 74*. [S.l.]: Cowles Foundation for Research in Economics, Yale University, 1959.

MARTÍN-BAOS, J. Á.; GARCÍA-RÓDENAS, R.; RODRIGUEZ-BENITEZ, L. Revisiting kernel logistic regression under the random utility models perspective. an interpretable machine-learning approach. *Transportation Letters*, Informa UK Limited, v. 13, n. 3, p. 151–162, jan 2021.

MCFADDEN, D. Conditional logit analysis of qualitative choice behaviour. *Frontiers in Econometrics, Academic Press*, 1973.

MCFADDEN, D. The revealed preferences of a government bureaucracy: Empirical evidence. *The Bell Journal of Economics*, JSTOR, v. 7, n. 1, p. 55, 1976.

MEYER, J. R.; CONRAD, A. H. Economic theory, statistical inference, and economic history. *The Journal of Economic History*, Cambridge University Press (CUP), v. 17, n. 4, p. 524–544, dec 1957.

MEYLER, A. et al. Forecasting irish inflation using arima models (no. 3/rt/98). *Central Bank of Ireland.*, 1998.

MOLNAR, C. iml: An r package for interpretable machine learning. *Journal of Open Source Software*, The Open Journal, v. 3, n. 26, p. 786, jun 2018.

MOLNAR, C.; CASALICCHIO, G.; BISCHL, B. Interpretable machine learning – a brief history, state-of-the-art and challenges. arXiv, 2020.
MONTAVON, G.; SAMEK, W.; MüLLER, K.-R. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, Elsevier BV, v. 73, p. 1–15, feb 2018.
MOSAVI, A. et al. Comprehensive review of deep reinforcement learning methods and applications in economics. *Mathematics*, MDPI AG, v. 8, n. 10, p. 1640, sep 2020.
MULLAINATHAN, S.; SPIESS, J. Machine learning: An applied econometric approach. *Journal of Economic Perspectives*, American Economic Association, v. 31, n. 2, p. 87–106, may 2017.
NADKARNI, P. M.; OHNO-MACHADO, L.; CHAPMAN, W. W. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, Oxford University Press (OUP), v. 18, n. 5, p. 544–551, sep 2011.
NAIK, N.; RASKAR, R.; HIDALGO, C. A. Cities are physical too: Using computer vision to measure the quality and impact of urban appearance. *American Economic Review*, American Economic Association, v. 106, n. 5, p. 128–132, may 2016.
NEVO, A.; WHINSTON, M. D. Taking the dogma out of econometrics: Structural modeling and credible inference. *Journal of Economic Perspectives*, American Economic Association, v. 24, n. 2, p. 69–82, May 2010.
OK, E. A.; TSERENJIGMID, G. Deterministic rationality of stochastic choice behavior. maio 2019.
OK, E. A.; TSERENJIGMID, G. Comparative rationality of random choice behaviors. jul. 2021.
OK, E. A.; TSERENJIGMID, G. Indifference, indecisiveness, experimentation, and stochastic choice. *Theoretical Economics*, The Econometric Society, v. 17, n. 2, p. 651–686, 2022.
PARKES, D. C.; WELLMAN, M. P. Economic reasoning and artificial intelligence. *Science*, American Association for the Advancement of Science (AAAS), v. 349, n. 6245, p. 267–272, jul 2015.
PETTIBONE, J. C.; WEDELL, D. H. Examining models of nondominated decoy effects across judgment and choice. *Organizational Behavior and Human Decision Processes*, v. 81, n. 2, p. 300–328, 2000.
PETTIBONE, J. C.; WEDELL, D. H. Testing alternative explanations of phantom decoy effects. *Journal of Behavioral Decision Making*, Wiley, v. 20, n. 3, p. 323–341, 2007.
PEYSAKHOVICH, A.; NAECKER, J. Using methods from machine learning to evaluate behavioral models of choice under risk and ambiguity. *Journal of Economic Behavior &amp Organization*, Elsevier BV, v. 133, p. 373–384, jan 2017.
REISS, P. C.; WOLAK, F. A. Chapter 64 Structural econometric modeling: Rationales and examples from industrial organization. In: *Handbook of Econometrics*. [S.l.]: Elsevier BV, 2007. p. 4277–4415.
RIBEIRO, M. Comparative rationality. 2023.
RIBEIRO, M.; RIELLA, G. Regular preorders and behavioral indifference. *Theory and Decision*, Springer Science and Business Media LLC, v. 82, n. 1, p. 1–12, jun 2016.
RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. [S.l.]: arXiv, 2016.
RIELLA, G. Preference for flexibility and dynamic consistency. *Journal of Economic*

Theory, Elsevier BV, v. 148, n. 6, p. 2467–2482, nov 2013.
ROBBINS, L. An essay on the nature and significance of economic science. *London: Macmillan*, 1932.
ROSENBLATT, F. The perceptron: A perceiving and recognizing automaton. *Cornell Aeronautical Laboratory*, 1957.
RUBINSTEIN, A. Instinctive and cognitive reasoning: A study of response times. *The Economic Journal*, Oxford University Press (OUP), v. 117, n. 523, p. 1243–1259, sep 2007.
RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, Springer Science and Business Media LLC, v. 323, n. 6088, p. 533–536, oct 1986.
RUST, J.; PHELAN, C. How social security and medicare affect retirement behavior in a world of incomplete markets. *Econometrica: Journal of the Econometric Society*, JSTOR, p. 781–831, 1997.
SAMEK, W. et al. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, Institute of Electrical and Electronics Engineers (IEEE), v. 109, n. 3, p. 247–278, mar 2021.
SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, IBM, v. 3, n. 3, p. 210–229, jul 1959.
SCHWARZ, G. Estimating the dimension of a model. *The Annals of Statistics*, Institute of Mathematical Statistics, v. 6, n. 2, mar 1978.
SEN, A. K. *On ethics and economics*. [S.l.]: Blackwell, 1987. 131 p. ISBN 0631164014.
SHAPLEY, L. S. A value for n-person games. *Contributions to the Theory of Games*, v. 28, n. 2, p. 307–317, 1953.
SIBSON, R. SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, Oxford University Press (OUP), v. 16, n. 1, p. 30–34, jan 1973.
SIFRINGER, B.; LURKIN, V.; ALAHI, A. Enhancing discrete choice models with representation learning. *Transportation Research Part B: Methodological*, Elsevier BV, v. 140, p. 236–261, oct 2020.
SILVA, M.; RIELLA, G. Aspiration-based reference dependance. 2020.
SIMON, H. A. A behavioral model of rational choice. *The Quarterly Journal of Economics*, Oxford University Press (OUP), v. 69, n. 1, p. 99, feb 1955.
SMITH, A. *An Inquiry into the Nature and Causes of the Wealth of Nations*. [S.l.]: University Of Chicago Press, 1977. 1152 p. ISBN 9780226763743.
SOUZA, M. J. S. et al. Can artificial intelligence enhance the bitcoin bonanza. *The Journal of Finance and Data Science*, Elsevier BV, v. 5, n. 2, p. 83–98, jun 2019.
SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 56, p. 1929–1958, 2014.
STEINHAUS, H. Sur la division des corps matériels en parties. *Bulletin L'Académie Polonaise des Science*, v. 4, p. 801–804, 1957.
STRATHERN, M. 'improving ratings': audit in the british university system. *European Review*, Cambridge University Press (CUP), v. 5, n. 3, p. 305–321, jul 1997.
SU, C.-L.; JUDD, K. L. Constrained optimization approaches to estimation of structural models. *Econometrica*, The Econometric Society, v. 80, n. 5, p. 2213–2230, 2012.
TEHRANI, A. F.; CHENG, W.; HULLERMEIER, E. Choquistic regression: Generalizing logistic regression using the choquet integral. In: *Proceedings of the 7th conference of*

the European Society for Fuzzy Logic and Technology (EUSFLAT-2011). [S.l.]: Atlantis Press, 2011.

TYSON, C. J. Cognitive constraints, contraction consistency, and the satisficing criterion. *Journal of Economic Theory*, Elsevier BV, v. 138, n. 1, p. 51–70, jan 2008.

VARIAN, H. R. Big data: New tricks for econometrics. *Journal of Economic Perspectives*, American Economic Association, v. 28, n. 2, p. 3–28, may 2014.

WANG, S.; WANG, Q.; ZHAO, J. Deep neural networks for choice analysis: Extracting complete economic information for interpretation. *Transportation Research Part C: Emerging Technologies*, Elsevier BV, v. 118, p. 102701, sep 2020.

WELLMAN, M. P.; RAJAN, U. Ethical issues for autonomous trading agents. *Minds and Machines*, Springer Science and Business Media LLC, v. 27, n. 4, p. 609–624, jan 2017.

YANG, B. et al. Application of ARIMA model in the prediction of the gross domestic product. In: *Proceedings of the 2016 6th International Conference on Mechatronics, Computer and Education Informationization (MCEI 2016)*. [S.l.]: Atlantis Press, 2016.