UNIVERSIDAD
TORCUATO DI TELLA

Master in Management + Analytics

A Machine Learning approach for increasing user engagement in the FinTech Industry

Thesis

Emiliano Lo Sasso

May 2022

Advisor: Nadja Maingard

Abstract

Is it possible to make a user become a regular operator of an application? Make him feel called to use it naturally, as one more task in his daily life?

In this thesis, we seek to respond to this not so trivial concern by using Machine Learning as a support tool for the development of two solutions that allow the user to get more engaged.

As part of a project within a Marketing team of a Fintech company, we seek to help users go from installing the app to the state defined as "Habit". To achieve this, we take advantage of the available data to develop two Artificial Intelligence models based on recommendation systems that seek to find the action within the application that has the greatest chance of being chosen by him.

In the course of this work, some basic concepts (and others not so much) necessary to understand both the business aspects and those related to the more technical aspect will be introduced.

As a final result, we have developed two models whose objective is to suggest the next most favorable action for the user, that is, the one that he would not do by himself but because it was recommended. Always in pursuit of getting the user to reach the state of Habit. The first of them, a model based on Markovian Processes, exploits the concept of the Transition Matrix to determine through it the probability that a person moves from one state (or operation) to another. The second of the solutions, based on machine learning techniques, seeks to find incremental suggestions through an Uplift model that determines those actions that are most likely to generate a positive impact on the user.

With this, we hope to improve the number of users who reach the status of Habit with respect to current initiatives, thus achieving more committed users and of greater value to the company, without neglecting their experience or their interests.

Contents

Table Index

List of Figures

1 Introduction

1.1 Background

For a complete understanding of any project it is important to fully comprehend the nature of the problem it is addressing and for that to be possible it is essential that some level of context is given to the reader so that the later explanation of the solution, results and conclusion are somewhat understandable.

For this reason, in the following sections I will be explaining the different aspects that are core to the project for the reader to have a good notion of what is the problem that I am addressing and the motivation behind this project as well as the context in which it resides, the industry and some key (technical) concepts that are required for a better use of this work.

1.1.1 Fintech Industry: Democratizing financial services

Before diving into the vast and dark deeps of technicalities, it is important to begin this journey with a *soft* approach: **Where do we stand? Where did it all began?**

However so profound the question is an important one. To begin to understand the motivation behind the project, we must first introduce the context in which everything takes place: **Fintech Industries.**

The word itself describes (in a way) what this new emerging companies are: *Financial Technology* Industries. It's a term applied for any technology that's used to augment, streamline, digitize or disrupt traditional financial services[42].

In essence, fintech companies are those that deliver financial services (like a bank would) through the use of technology. They are *technology based* companies. When compared to traditional banks or online banks, the core itself is very similar between the three: they all aim at solving a financial related problem to the user whether it is a payment issue, insurance, loans, credits, etc.

However, compared to the industry's traditional stakeholders, these companies successfully innovate by introducing disruptive business models, taking advantage of the new technologies functionalities and lower costs to offer more efficient products and services.[4]

Originally, the term was used to referred to the back end technology used to operate traditional financial services institutions. Today, it has broadened to incorporate new technological innovations in the financial sector, such as blockchains, cryptocurrencies, robo-advising, and even crowdfunding.

In nowadays digital era, traditional financial institutions are no longer meeting the needs of today's consumers, who are growing used to the times of modern technologies. They prefer to bank with just a click of a mouse or a swipe on their smartphone. As consumers become accustomed to the digital experience provided by tech leaders like Google, Amazon, Facebook, and Apple, they expect that same level of digitally integrated customer experience from their financial services providers[29]. Currently, fintech industries are finding ways to innovate the financial services in order to meet this new consumer's needs.

Between those innovations, you can find everything from being able to check your financial transactions online to applications that allow you to instantly send and receive money.

All of these innovations brought on by the outbreak of Fintech industries imply an important transformation in the industry, one that brings certain benefits for the people

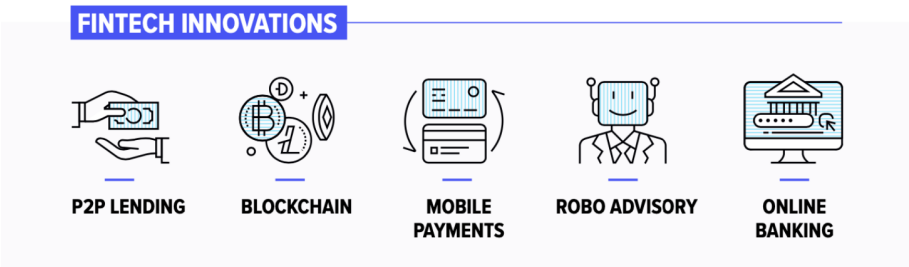


Figure 1. Fintech Innovations[28]

and the working sector when we consider the situation in the region this work takes place: Latin America.

First, the previously mentioned reconfiguration is likely to contribute to bridging the financial gap affecting the region's productive sector. This is especially true for small and medium enterprises (SMEs), which play a critical role in productive development, employment, and economic growth in Latin America. On the one hand, the emergence of new online financial platforms and intermediaries –with lower transactional fees and new techniques and information sources to assess credit risk– will contribute to boost SMEs access to credit. On the other hand, existing payment solutions and digital tools to obtain better business financial performance shall not only promote these transactions digitalization and formalization, but also make the transaction history or digital footprint available to evaluate credit risks, creating new options to solving information asymmetries and positive consequences in terms of opportunities to obtain financial support.[4]

Secondly, the rising of these new stakeholders allows a very big part of the population that still remains excluded or underserved by the traditional financial industry to be part of the system, allowing them to access all the benefits that come from it (credit, insurance, etc). It's worth noting that while financial exclusion, as measured by the possession of a bank account to holding a bank account, is estimated at about 49%, this figure rises significantly when the use of credit, savings or insurance instruments is factored in.[4]

Of course, this benefits not only mean that people will now have a way to pay for their products or services. When we take a look at the main activities involving Fintech we see that this kind of business builds a whole ecosystem around the user where everything is connected and whatever they do can be used as an input for the rest of the services available, reducing the lack of information that the traditional financial services, like the banks, suffer from, and finally granting the users the possibility to access a more complete set of services.

This ecosystem could be summed up into ten general groups (according to the IDB[4] study) where each one contains a set of activities involving Fintech services that try to solve one or more of today financial problems.

Table 1. Fintech Segments

Segment	Businesses
Payment Solutions	• Mobile payments and wallets • International money transfer and remittances • Mobile points of sales • Payment gateways and aggregators
Personal Financial Management	• Savings and Financial efficiency • Comparison sites • Debt management
Asset Management	• Digital wealth management • Robo advisors
Trading and Capital markets	• FX Solutions • Stock Market Solutions
Enterprise technology for financial institutions	• Security and digital ID • KYC solutions • Fraud prevention and risk management • Biometrics • Smart contracts
Digital Banks	
Insurance	
Alternative finance platforms	• Rewards crowdfunding • Donations crowdfunding • Real estate crowdfunding • Equity crowdfunding • Balance sheet business lending • Balance sheet consumer lending • P2P business lending • P2P consumer lending • Factoring invoice lending
Alternative scoring	

It has been settled that the new rising Fintech Industries can help in the development and growth of businesses and, more importantly, they represent an entrance for many people who are currently not part of the financial system. But also, they help to make payments, transactions, investments and personal economy a much more easier task than it is today. Lastly, these industries built an ecosystem that bring forward financial solutions for companies and users alike, which means there is not only one service available but instead a whole catalog of them, each one designed to solve a different problem. This fact raises a question: **How is the user supposed to choose between the different options? How can the companies offer the best service to their clients without spamming them?**

Recommender systems solve this problem by searching through large volume of dynamically generated information (obtained through user interaction with the fintech ecosystem) to provide them with personalized content and services.[17]

1.1.2 Recommendation Systems: Choosing the best action

In the last few decades we have seen big companies like Youtube, Amazon or Netflix grow more and more important in the market and into our lives. In the e-commerce business, one can find almost anything just by looking into Amazon publications. On the other hand, Youtube has outgrown it's original purpose and now it is not only a web service for sharing videos, but a streaming media and an actual way of living too.

All of this might not seem impressive today as we are used to use this type of services,

but it is important to remember that these companies were born a few years ago. With this in mind, and seeing how fast organizations arise and fall in a short period of time, one could ask how are this companies different from the rest? Is there some secret recipe they have?

Part of their success is due to the rise of **recommendation systems**. From e-commerce to online advertisement, recommender systems are today unavoidable in our daily online journeys [32].

Putting it simply, a recommendation system is an algorithm developed for suggesting relevant items to users: movies to watch, texts to read, products to buy, etc.

As simple as it seems, this systems can bring significant value to the companies that use them as they provide an edge that separates them from the rest of the competitors. This is due to the fact that recommendation systems are designed to suggest items that might *interest* users. In other words, the real goal behind this strategy that more companies are starting to apply is to *understand* their clients/users. To understand them means to offer real value propositions that can adjust to their needs, creating a more memorable experience as the clients feel the companies on the other side get them.

Recommender systems can be used to personalize the content of websites for each visitor individually. Other channels such as e-mail newsletters or mobile notifications can be personalized as well. User interactions from multiple channels feed a recommender system, increase the precision of recommendations and improve the personalized experience of users.[19]

By achieving this level of personalization, users spend less time searching for an item and can potentially discover new items of interest. As a result, customers begin a cyclic process in which their levels of loyalty tend to increase as well as the satisfaction with the organizations, leading to an increase in the number of items with which they interact with, causing more consumption and higher profits for the company. Also, because of the newsletters, personalized promoted content and push notifications suggested by the recommendation system, users are encouraged to return, increasing the frequency of visits, reducing churn and increasing their lifetime value.

This is aligned with the main goal of recommender systems, which is the increase of a product's sales (this is expressed in a general way as the sale of a product could refer to the download of an app, buys in a marketplace, etc). After all, these systems are used by companies to increase their profit, as it was previously mentioned. By recommending carefully selected items to users, recommender systems bring relevant items to the attention of users. This increases the sales volume and profits for the company. Although the primary goal of a recommendation system is to increase revenue, this is often achieved in ways that are less obvious than might seem at first sight. In order to achieve the broader business-centric goal of increasing revenue, the common operational and technical goals of recommender systems could be expressed as follows, according to Aggarwal[2].

- **Relevance**: The most obvious *operational* goal of a recommender system is to recommend items that are **relevant** to the user. Users are more likely to consume items they find interesting. Although relevance is the primary operational goal of a recommender system, it is not sufficient in isolation.

- **Novelty**: Another goal of recommender systems is to bring forward new options that the user has never seen before, but that remains truly helpful in terms of satisfying the user's needs. The system will be defective or will lack in originality if

it keeps recommending things that the user has seen in the past (like an old movie) or if it just suggest the most popular films.

- **Surprise**: A related notion is that of *surprise* or *serendipity*. It occurs when a certain recommendation takes the user by surprise, presenting something unexpected, generating the feeling of discovery instead of the same obvious recommendations. It is different from novelty because the recommendations are truly surprising to the user, rather than simply something they did not know about before. It may often be the case that a particular user may only be consuming items of a specific type, although a latent interest in items of other types may exist which the user might themselves find surprising. Unlike novelty, serendipitous methods focus on discovering such recommendations. One positive aspect of this element is that it can help increase sales diversity and it could awake new areas of interest to the users, expanding their possibilities. Of course, this also implies that these algorithms could suggest irrelevant options. Generally, the long term benefits outweighs the short term disadvantages.

- **Increasing recommendation diversity**: An important aspect of a recommendation system is that it has the ability to offer diverse elements to the users. Otherwise, if it only shows the top k-best selections (and they are similar), then there is the risk that the user might end up not liking **any** of the options presented to him. By ensuring a diverse set of recommendations, there is the chance that one of them might be chosen.

Aside from these concrete goals, there are some others that are important from both the perspective of the user and the company. From the user's perspective, recommendations can help improve overall user satisfaction with the platform that is providing the recommendation (website or mobile). For example, a user who repeatedly receives relevant recommendations from Amazon.com will be more satisfied with the experience and is more likely to use the site again. This can improve user loyalty and further increase the sales at the site, just as it was mentioned before. On the other hand, from the company's point of view, the recommendation process can provide insights into the needs of the user and help customize the user experience further. Finally, providing the user an explanation why a particular item is recommended is often useful. For example, in the case of Netflix, recommendations are provided along with previously watched movies.[2]

But are all of these systems the same? Is there any difference between the systems at Netflix and Amazon? The answer is yes, they are different because the recommendations belong to different areas. Table 2 presents some products recommended by various real-world recommender systems.

System	Product Goal
Amazon.com	Books and other products
Netflix	DVDs, Streaming Video
Jester	Jokes
GroupLens	News
MovieLens	Movies
last.fm	Music
Google News	News
Google Search	Advertisements
Facebook	Friends, Advertisements
Pandora	Music
YouTube	Online videos
Tripadvisor	Travel products
IMDb	Movies

Table 2. Example products recommended

Although many of the recommender systems presented in Table 2 are focused on traditional e-commerce applications for various products, including books, movies, videos, travel, and other goods and services, they have expanded beyond the traditional domain of product recommendations. Such is the case of Facebook or Google Search, where it can be seen that the goal is to recommend, for example, advertisements.This implies that different "product goals" require different recommendation systems, each one with it's own strategy and models behind them.

1.1.2.1 Types of Recommender Systems

Underneath these systems we could find complex algorithms running day and night trying to find the best set of features to outperform itself. As will be mentioned in the following section, recommendation systems run on *machine learning* models, and depending on the data they use, these systems can be classified into different methods:

1. *Collaborative filtering methods*: They use the user-items interactions data, such as ratings or buying behavior. Collaborative filtering methods collect and analyze information on user's behaviors, activities or preferences.[16]

2. *Content-based recommender* methods: They use attribute information about the users and items such as textual profiles or relevant keywords. The hypothesis is that if a user was interested in or bought a specific product in the past, they will again be interested to buy or select the same product in the future. In this type of recommender system, keywords are used to describe the items and then a user profile is built to indicate the type of item this user likes.[16]

3. *Knowledge based recommender systems*: As their name suggest, these systems base their recommendations on explicit user requirements. Instead of using historical rating or buying data, external knowledge bases and constraints are used to create the recommendation.

4. *Hybrid systems*: They combine the strength of all of the previously mentioned systems to create more robust recommendations that can be applied to different settings.

1.1.2.2 Recommendation process

Once the product or service to be recommended is defined and the type of system to use is selected, what follows is a recommendation process that is very similar in most cases.

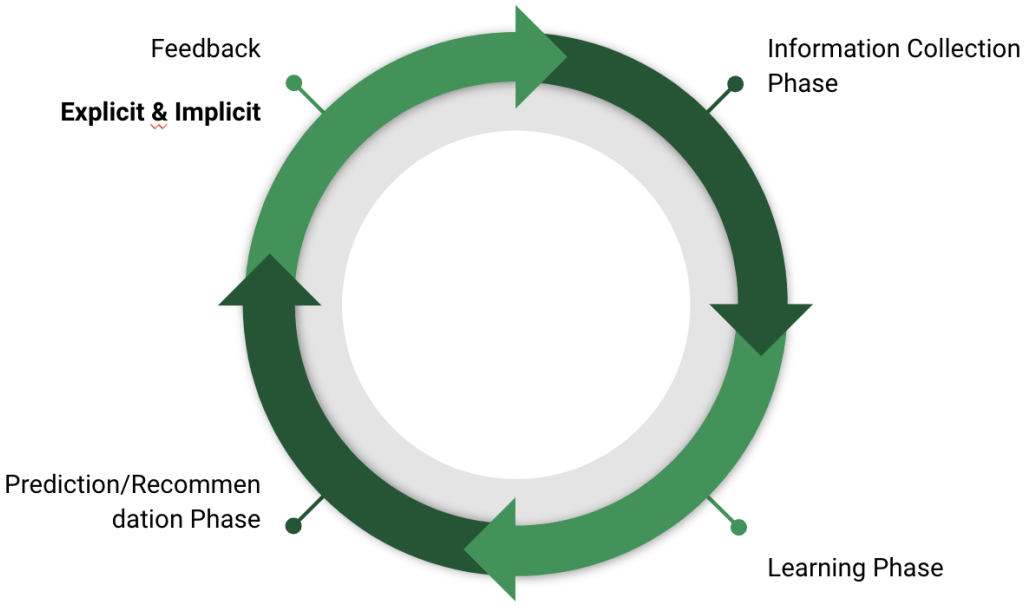


Figure 2. Phases of recommendation process

1. *Information collection phase*: During this phase a *user profile* is created by collecting relevant information about the users. The information could include user's attributes, behaviour or content of the resources the user accesses. This instance is necessary as any recommendation algorithm use this profile for the future suggestions as it helps to compare similar users in the future. The user profile is normally used to retrieve the needed information to build up a model of the user. Thus, a user profile describes a simple user model. The success of any recommendation system depends largely on its ability to represent user's current interests. Accurate models are indispensable for obtaining relevant and accurate recommendations from any prediction techniques.[17]

2. *Explicit Feedback*: For any system to learn, it requires feedback. It could be given directly from the user (explicit) or it could be inferred by analyzing the user behaviour and actions (implicit). This information can help the model to know how good are the recommendations and whether they need to be improved. Explicit feedback require an effort from the users because it requires them to say if the recommendation was a good fit for him or not and why was that. This could be a tedious task for some users and could, at the very least, affect the user experience. For these reasons it is not always possible to collect feedback from the users in a

direct way. On the other hand, the information provides more confidence into the recommendations as it is the user itself who said whether it was good or not.

3. *Implicit Feedback*: The system automatically infers the user's preferences by monitoring the different actions of users such as the history of purchases, navigation history, and time spent on some web pages, links followed by the user, content of e-mail and button clicks among others. Implicit feedback reduces the burden on users by inferring their user's preferences from their behavior with the system. The method though does not require effort from the user, but it is less accurate.[17]

4. *Learning Phase*: A learning algorithm (which will be explained in more detail in the next section) is applied with the objective of filtering and exploiting the user's features that were obtained from the feedback, gathered in information collection phase.

5. *Recommendation Phase*: It recommends or predicts what kind of items the user may prefer. This can be made either directly based on the dataset collected in the information collection phase which could be memory based or model based or through the system's observed activities of the user.[17]

This pipeline adds another question that keeps us digging deeper into these systems. As it is mentioned, all recommender systems go through a *learning phase*, a moment in which an algorithm process all the information in order to learn from it and be able to generate it's own recommendations. But, **what does it mean to have a learning algorithm?**

The next section will introduce the concept of **Machine Learning** and the main techniques behind these learning algorithms, answering the complex question of what does it mean to have a computer learning.

1.1.3 Machine Learning

Nowadays, with the revolution of **Big Data**, some terms like *artificial intelligence, machine learning* and *deep learning* have become very popular and are used as if they where interchangeable but although they are very similar, they do not refer to the same things entirely.

Artificial Intelligence

Firstly there is **Artificial Intelligence** (**AI**). Very popular in the movies and is generally used to say that a machine is working by its own. A more formal definition could be that it is *the effort to automate intellectual tasks normally performed by humans.*

As such, AI is a general field that encompasses machine learning and deep learning, but that also includes many more approaches that don't involve any learning.[9]

For example, the first programs that could play chess were actually a predefined set of harcoded rules the machine followed, but they couldn't actually learn anything new. The machine couldn't improvise, it couldn't follow a different strategy depending on the opponent. Although this approach was very popular for solving well defined logical problems, they failed at figuring out explicit rules for solving more complex problems like image classification, speech recognition and language translation. This is why a new approached arise: **Machine Learning**.[9]

Machine Learning

Up to this point, all this programs are capable of doing what they are hard coded to do. In other words, it is like they are following a recipe: one step after the other until they complete the task at hand. But what if computers had the ability to learn how to solve real problems on its own?

This question implied a new programming paradigm. Before, programmers designed rules that were meant to be followed. Data was then passed through this rules in order to obtain a certain output. Now, the new paradigm propose that, with machine learning, humans input data as well as the answers expected from the data, and out come the rules. These rules can then be applied to new data to produce original answers.[9]

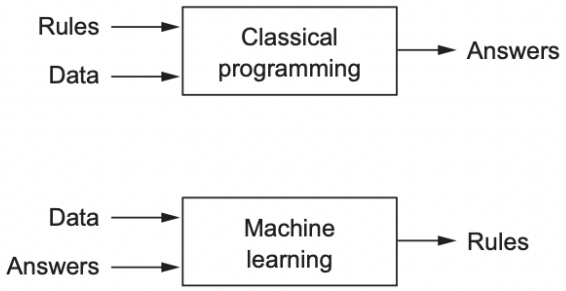


Figure 3. A new programming paradigm[8]

In conclusion, machine learning systems are *trained* to solve a certain task rather than *programmed*. There is no need to previously understand all the possible outcomes and note down every rule. Instead, machine learning pass down the data and the answer expected to the system and "learns" the necessary rules for it to solve new instances of the problem.

These systems are not separated from the field of AI but in fact they are part of it. It is a successful sub-field. So these systems are still focused on automating tasks that are normally performed by humans. But, the approach taken is to train computers for learning the intrinsic rules behind a set of data so that in the future, when new data arrives, it can automatically perform the task that it was meant to do, without the need of human supervision.

To formalize the concept, Machine Learning can be defined as *"the field of study that gives computers the ability to learn without being explicitly programmed"*.[11] This "learning" is, in fact, the process executed by the model through which it transforms its input data into meaningful outputs, by exposing it to known examples of inputs and outputs. Therefore, the central problem in machine learning is to learn meaningful representations of the input data (within a predefined space of possibilities, using guidance from a feedback signal) that can get it closer to the expected output.

The reason why these systems are so important is because they are very useful for solving a specific task that can't be previously programmed like image classification, speech generation, churn prediction, etc.

Businesses incorporate ML into their core processes for a variety of strategic reasons. It can deliver benefits such as the ability to discover patterns and correlations, improve customer segmentation and targeting, and ultimately increase a business revenue, growth and market position.

Deep Learning

But if machine learning already is a way for the computers to "learn" without explicitly being programmed, then **what is deep learning and what are the main differences between them?**

Just as machine learning was a sub-field of AI, deep learning can be considered as a specific sub-field of the former one. Just as promised, all these concepts are related to each other - they are all related to the concept of artificial intelligence - but they differ in the techniques and methods applied, as well as in the task they can perform.

Deep learning algorithms focus on the problem of learning meaningful representations of the data, with an emphasis on learning them through successive *layers*, each one more meaningful than the previous one. This idea of successive layers of representations is what the *deep* in deep learning stands for. In fact, the more of these layers are placed, the deeper the model. Nowadays, they can go from a few to even hundreds of successive layers of representations, depending on the task they are designed to learn from, and these are all learned automatically without any human intervention, solely through exposure to training data.

So far, it has been established that automation is possible through artificial intelligence. The idea of a computer automatically doing certain tasks makes perfect sense. We have also been able to dig deeper and found out that a good way to learn automatically can be achieved through machine learning algorithms. Putting it all together, we can safely assume that if we have a problem and we can translate it, somehow, into data, there is a chance that an algorithm can learn from it by creating a "good enough" representation, which it can later be used to solve the original task, this is big step forward.

Like recommender systems, machine learning algorithms can be classified into different groups depending on the data and the way they learn, according to Géron[11]:

- Whether or not they are trained with human supervision (**supervised, unsupervised, semisupervised**, and **Reinforcement Learning**)
- Whether or not they can learn incrementally on the fly (**online** versus **batch learning**)
- Whether they work by simply comparing new data points to known data points, or instead detect patterns in the training data and build a predictive model, much like scientists do (**instance-based** versus **model-based learning**)

These different groups are composed of different kind of models and algorithms, depending on the application.

Up to this point, it is safe to say that we have covered the most important elements for the reader to understand the main goal this work pursues (which will be introduced later on). However, before moving on, it is crucial to place this work in a certain context, as it is not the first project about recommender systems nor machine learning that has been made. In the following section I will present the different work and researches that have been made in the subject with the sole goal of familiarizing with the business cases that these systems have in the industry. The research will also put this work into perspective as it will help define its scope and what will be covered by it and what will not.

In the following sections I will summarise the main differences between them and the models that can be found in each one.

1.1.3.1 Supervised & Unsupervised Learning

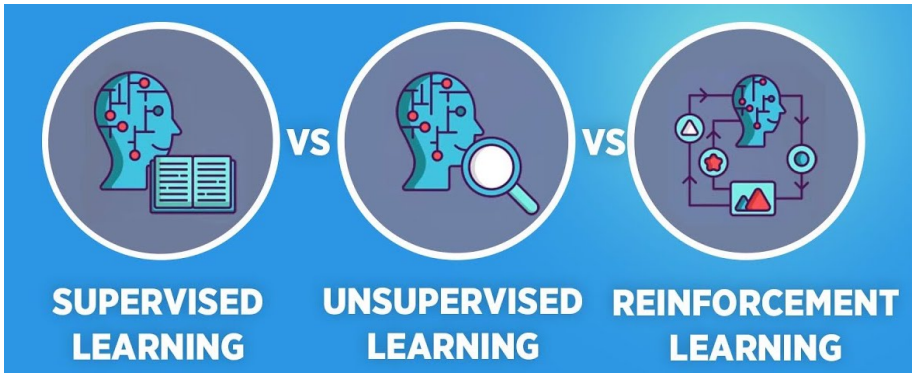


Figure 4. Different learning types[41]

Machine Learning systems can be classified according to the amount and type of supervision they get during training. The concept of "supervision" refers to the idea that while training, the algorithm knows how well it is performing by comparing the output it generates against the real output. There are four major categories: supervised learning, unsupervised learning, semisupervised learning, and Reinforcement Learning[11]. Each of these groups have a different level of supervision that sets them apart.

Supervised Learning

These systems are characteristic for having an input set of data that is comprised of:

- *Feature set*: Normally called a *dataset*, this data contains detailed information (features) related to the task that is trying to be solved with the algorithm. For example, if the goal is to estimate the churn probability of a client, then the features could contain personal data of them, transaction movements, geo-referenced data, etc. In essence, the feature set contains information that will be analyzed in hopes of finding a significant relation with a dependable variable or output that we are trying to predict.

- *Target set*: This data is the concept of supervision that was previously mentioned. Basically, it is the *desired* output that these models try to replicate through learning.

In other words, in supervised learning, the training data you feed to the algorithm includes the desired solutions, called **labels**[11].

A typical supervised learning task is **classification**. A good example of this are image classification programs. Nowadays they are very good introductory projects, especially in Deep Learning courses. They consists of a dataset with many labeled pictures (it could be animals, objects, fruits, faces, etc.). The fact that they are labeled means that we already know **what** is in that picture. The goal of these models is to correctly classify each picture.

Another typical task is to predict a target numeric value, such as the price of a car, given a set of features (mileage, age, brand, etc.). This sort of task is called regression. To train the system, many examples of cars are needed to be provided, including both their predictors and their labels (i.e., their prices)[11].

Some of the most common supervised learning algorithms are:

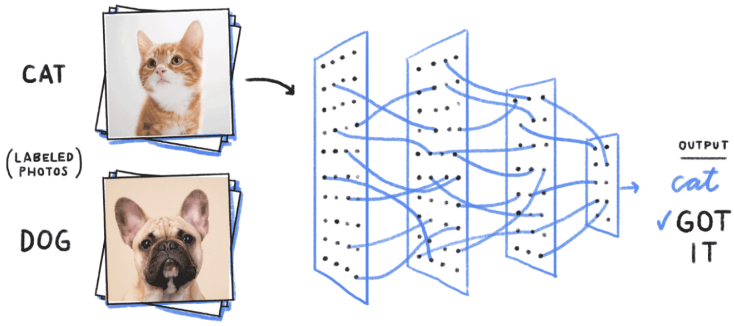


Figure 5. Classification models[36]

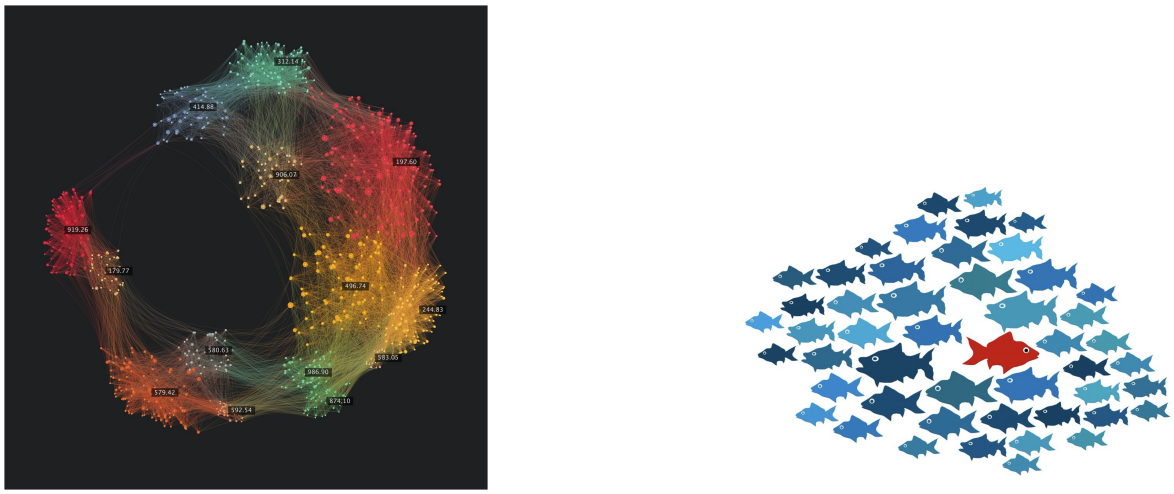
- **k-Nearest Neighbours**
- **Linear Regression**
- **Logistic Regression**
- **Support Vector Machines (SVMs)**
- **Decision Trees and Random Forests**
- **XGBoost**
- **Neural Networks**

Unsupervised Learning

As suggested by the name these systems do not contain any kind of labels, therefore, the algorithm tries to learn by itself. These models are mostly used for more analytical tasks where the end goal is not always so clear in the sense that anything that comes as output will ultimately be a "discovery". When using an unsupervised model we are trying to understand the data, not predict a specific outcome.

Some of the most important models of this category include:

- **Clustering algorithms**
  - K-means
  - DBSCAN
  - Hierarchical Cluster Analysis
- **Anomaly detection and novelty detection**
  - One-class SVM
  - Isolation forest
- **Visualization and dimensionality reduction**
  - Principal Component Analysis (PCA)
  - Kernel PCA



(a) Clustering[39] **(b)** Anomaly detection[3]

Figure 6. Unsupervised learning examples

  - Locally-Linear Embedding
  - t-distributed Stochastic Neighbor Embedding

- **Association rule learning**
  - Apriori
  - Eclat

As it was mentioned, most common tasks associated with unsupervised learning are related with obtaining a good understanding of the data given the fact that it comes without any kind of label or output, therefore there is nothing to learn but to discover. Clustering, anomaly detection or dimensionality reduction are all trying to gain a certain knowledge about the data -whether it is to group it, detect uncommon behaviour, or reduce redundant information- there is no "good" output, they are trying to "understand" the data.

Semisupervised Learning

These systems are a middle ground between both supervised and unsupervised. The algorithms can deal with partially labeled training data (in most cases, there are many unlabeled data and just a few labeled data).

Reinforcement Learning

Reinforcement Learning is very different from the rest of the learning techniques. The learning system, called an **agent** in this context, can observe the environment, select and perform actions, and get **rewards** in return (or penalties in the form of negative rewards). It must then learn by itself what is the best strategy, called a **policy**, to get the most reward over time. A policy defines what action the agent should choose when it is in a given situation.

In a way, reinforced learning algorithms learn not by looking at a label, but they do it in a similar way that a baby would learn to identify objects. It observes, based on that observation it takes an action (in the case of a baby it could be that he grabs a banana

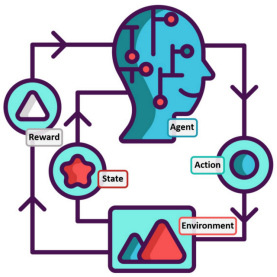


Figure 7. Reinforcement learning[22]

instead of an apple) and waits for a reward that tells him if the action was the right one or not. With time, the agent learns to choose the actions that can give him the biggest reward. In essence, it learns through trial and error, just as a baby learns that a banana is different than an apple.

1.1.3.2 Batch & Online Learning

Another criterion used to classify Machine Learning systems is whether or not the system can learn incrementally from a stream of incoming data.

Batch Learning

In batch learning, the system is incapable of learning incrementally: it must be trained using all the available data. This will generally take a lot of time and computing resources, so it is typically done offline. First the system is trained, and then it is launched into production and runs without learning anymore; it just applies what it has learned. This is called offline learning[11]. For it to learn from new inputs it is necessary to re-train the dataset with all the data -old and new- and replace the old model with the new, more updated one.

The main problem with this kind of learning is evident, if the dataset starts to grow significantly, the process time required starts to rise. By itself this is not a serious problem if the time is not a valuable asset or if it is not necessary to update the models in a frequent manner. However, if the dataset changes very quickly (forcing the model to update too) then it may even be impossible to use a batch learning algorithm.

A better option in all these cases is to use algorithms that are capable of learning incrementally.

Online Learning (Incremental learning)

The system train incrementally by feeding it data instances sequentially, either individually or by small groups called mini-batches. Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives.[11]

In contrast with batch learning, these algorithms are very useful for systems that receive data in a continuous flow and that need to adapt quickly to those changes. They are very good if the processing resources are limited, this is because once an online learning system has learned about new data instances, it does not need them anymore, so they can be discarded[11]

A big challenge with online learning is that if bad data is fed to the system, the system's performance will gradually decline. This is especially bad if the system is live because the users would immediately notice it. To reduce this risk, the system should be

closely monitored and, in case there is a decline in performance, switch learning off (and possibly revert to a previously working state). It is also important to monitor the input data to check for abnormal data.

1.1.3.3 Instance-Based & Model-Based Learning

Machine learning systems can be classified based on how they generalize. This is an important concept in the field because the main goal of these models is to learn in a way that will allow them to solve the input data on their own. That means, being able to generate an output for input data that it has never seen before (otherwise, the systems wouldn't be very useful if it could only solve for known data)

There are two approaches to generalization:

Instance based learning

The most simple way of learning that exists would be to learn by hard (one could argue that it is not actual learning but memorizing). The idea behind these systems is that it learns the input data examples by **hard** and then generalize it by comparing new data to the one it memorized using a **similarity measure**. For example, in Figure 8 a new instance is introduced into the data and, because the majority of the classified training instances belong to the triangle class, then this new one will be classified as one too.

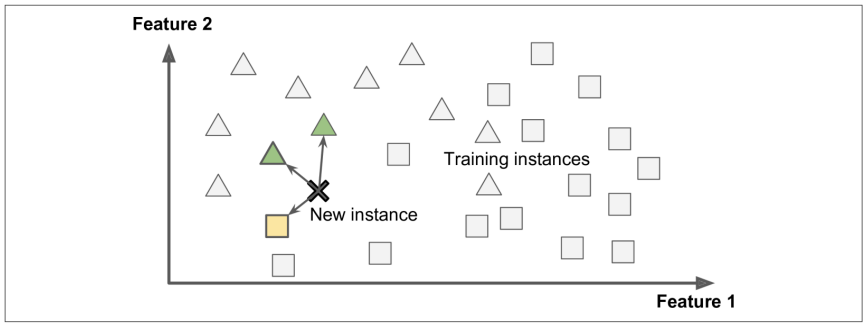


Figure 8. Instance based learning[11]

Model based learning

In these systems, the generalization is achieved by building a model of the data, and later use that model to make predictions.

The goal is to construct a *good* representation of the data, not so much to memorize it exactly. This allows for the model to generalize in such a way that even if new data comes in, it will still perform nicely (if the model is good, that is, if the generalization is good enough).

The model selection should be done after a detailed study of the data that is being modeled. In some cases, a linear model fits perfectly, in some others, a more abstract not linear representation is needed. This depends entirely on the data available and, in many cases, there is more than one suitable option.

Before using the model, it is necessary to define its parameters, after all, a model is nothing more than a mathematical expression that tries to **model** real world scenarios. However, **how can we know which values will make the model perform best?**

To answer this question, a performance measure is specified. It could either be a *utility function* (or fitness function) that measures how good the model is, or a *cost function* that measures how bad it is. For linear regression problems, for example, people typically use a cost function that measures the distance between the linear model's predictions and the training examples; the objective is to minimize this distance.[11]

After defining every important aspect -the model and the performance measure to be used- the next step is to train this model. In this case, the training process consists of learning the right parameters that optimize the performance measure, either by maximize or minimize it.

The result of the model-based systems is a good representation of the data that can make predictions (whether it is a classification or a regression) on new, unknown data inputs. If all went well, the model will make good predictions. If not, the process should be reviewed in order to improve whatever that is causing trouble: adding more attributes, getting more or better quality training data, or perhaps selecting a more powerful model.

1.1.4 Related Work

The idea of applying self taught algorithms to make businesses like decisions is not new, as it was mentioned before -we could go back to the 90's when the first learning programs were being developed[23].

For this section, I will present some examples of machine learning algorithms that have been used in the industry (and still are). These examples will be focused mostly in recommender systems (as it is the core idea behind this work). However not all of them might be related to fintech industries, but the idea behind them still remains the same: **let a computer study, learn and ultimately suggest a user what to choose from a pool of options.**

In the entertainment and streaming industry, companies like Netflix and Youtube are applying machine learning into the core of their businesses. Netflix uses their user's viewing and search history in order to recommend content that they might like. They have multiple algorithms working together to define the so called *Netflix experience*. Their **personalized video ranker** or **PVR**, which orders the entire catalog of videos (or subsets selected by genre or other filtering) for each member profile in a personalized way.[12]. They also use machine learning to select their user's **Top - N** titles (the ones presented to us at the landing page) in a much more personal manner, as it suggest its users what it might like them the most, however the genre. Netflix applies machine learning into almost every aspect of their product: the **Trending now** recommendations, comprise of both seasonality trendings as well as one shot occasions. Their **Continue watching** row, although it might seem random at first, has a recommender system running behind it that sorts the unfinished content according to the likelihood that the user will resume it. Another row that benefits from machine learning is the **Because you watched** row. Under the hood, those recommendations are elaborated based on similarity. So, given a certain content that a user watched previously, the algorithm find similar videos that he might enjoy too. Finally, not only the content is suggested through algorithms, but also the layout of the platform itself. Which rows to show to the user, how many dedicated to trending, or similar videos, etc. All the layout change for each user, even during the day, the same user might see two different layouts depending on the mood, the day, and many other factors that Netflix knew how to exploit in order to give a true personal experience. By applying machine learning into their business, Netflix has achieved meaningful increases in overall engagement with the product (e.g., streaming

hours) and lower subscription cancellations and churn rates, saving the company almost 1B$ a year[12].

Amazon was one of the pioneers in the use of machine learning in their processes. In a study conducted a few years ago by Linden et al.(2003) (almost at the very beginning of Amazon.com) their research team developed an innovative way to tackle the recommendation problem. Through a *item-to-item collaborative filtering* they manage to create a recommender that was both precise and scalable. By implementing this new system, they were able to tackle a very common problem that persists nowadays, that is, doing online computations with very low latency but keeping a good level of recommendations.

Similarly, a few years later, a machine learning model implemented through reinforced learning was developed and put into production for the **Amazon music conversational recommender**. In the experiment, researchers tried to make Alexa (Amazon's virtual assistant) suggest better alternatives to the users. Since 2018, Amazon Music customers in the US who were not sure what to choose were able to converse with the voice assistant, allowing it to fill the gaps in the information in order to help the customers arrive at their right choice. The Amazon Music Conversations team developed a next-generation of conversation-based music recommender, one that harnessed ML to bring the Alexa music recommender closer to being a genuine, responsive conversation, as mentioned by Sean O'Neill (2022). The team ran two experiments to improve conversational efficiency (that is, asking better questions) and to incorporate user's history (Tao Ye et al., 2021). As mentioned, through an offline reinforced algorithm they were able to "teach" Alexa to ask for questions that would increase the chances of finding the right choice for the users. As a result of this model, the team achieved an increase of 12% in customer's outcomes and they reduced by 13% the number of conversational turns.

Regarding the music industry, in a study conducted by Francesco Sanna Passino et al. (2021) from Spotify, they came up with a new approach for a recommender system that took into account the constantly evolving preferences of the users. As they put it, most recommender systems try to capture simultaneous preferences: "Users who like A tend to like B as well". Although this approach proved useful at suggesting similar items, it might conceal the users into a specific group, excluding them from the possibility of genuine discovery, causing them to feel like they are constantly presented with the same options. The authors came up with a solution, a **Preference Transition Model (PTM)**, a dynamic model of preferences that predicts how users move in the space of items. This model can anticipate changes in users' preferences, and accompanies them through their journey. Furthermore, it can gradually bring users towards unexplored regions of item space, thus increasing diversity in a meaningful way. The results were promising as the model outperformed other recommender methods in a variety of datasets. Furthermore, the model output -a matrix of transition probabilities between item classes- is very interpretable. This helps researchers to understand how their users behave and how their preferences change over time.

With these few examples, I have established the importance of machine learning in modern industries. It is possible to train an algorithm to extract important correlations among the data and use that learning to automate specific task that we, as human, would never be able to do. I believe that, through a machine learning model, it is possible to solve the problem of recommending the best products of a fintech industry that offers a wide variety of services, as that is the goal of this work.

1.2 Justification

Previous examples provide proof that recommender systems are not science fiction nor impossible to use, instead, we are surrounded by them. Under the assumption that these systems are applicable to any situation in which the user has many services to choose from (assuming the necessary data is available), my proposal is to develop one of these systems for a fintech company (which will remain anonymous) but with an end goal that differ (however slightly) from those previously seen in the examples.

The company involved in the project focus most of its business efforts into its mobile applications, which, for simplicity, I will say there are two: an e-commerce and a fintech. This is because the company was originally born as an e-commerce business, but later expanded into the world of finance as a natural transformation, in order to create an ecosystem.

For most applications, user's life-cycle consist of different stages: discovery, installation, engagement, revenue and referral[5]. These stages are not written in stone, they change and there might be more or less depending on the company and their business. However, it is undeniable that for every application to work, there are two necessary events that must occur: The user needs to install the app and he needs to **use** it, frequently, he needs to be engaged.

This is where my project comes in. At the company, there are teams working in every stage of the life-cycle, especially in the installation part. However, almost nothing is being done in terms of engagement, not directly. Different business units focus on specific services that the app can provide, but not one of them is facing this issue in a global way, with the objective of achieving user engagement.

I propose to develop a solution to take on this challenge: Produce a differential increment in the number of users that become engaged with the app. I will be taking advantage of my position as part of the Marketing Modelling team inside the company. This position grants me access to the data necessary to propose a solution and to different channels through which we can reach those users.

1.3 Objective

The objective of this project is to develop an **artificially intelligent** model that will suggest services to the users in order to get them to an engaged stage of their life-cycle. Note that I did not say a **machine learning** model. That is because this problem will be approached from two perspectives.The first one is a simpler rule based model approach, where the suggestions will be made through a transition matrix built from user's behaviour. The other one will be a machine learning approach, where an **Uplift** model will be developed to determine whether suggesting certain services to the users will positively influence them or not. It is important to highlight that the objective is not the development of the model itself, but to obtain a significant increment in the number of users that get to an engaged stage of their life-cycle.

The first solution was chosen because of its simplicity and because it mimicked users behaviour. The other model, by its own definition, looks for incremental changes by selecting actions that can generate impact. Because the objective is to get users that, **if left alone**, will never use the app to a point where they use it frequently, the uplift model suited perfectly for our needs.

The goal of this work includes not only the development of the aforementioned models, but also their testing. Given the possibility to create marketing campaigns to actively

reach users, I am also in a position to continuously control the campaigns development. For this reason, the testing and final conclusions -whether the solutions worked or not- are also a part of the objective.

2 Data

The data used for the development of this project was obtained from a company in Argentina that operates online marketplaces dedicated to e-commerce. Operating under five main business units, this company is one the largest in Argentina.

For the purposes of this work, the data was from their fintech unit. This was possible as I am part of such company, allowing me to access the necessary data for the project. For privacy reasons, some feature names might change. Also, any sensible information regarding the users (phone number, names, etc) will remain excluded from this work.

First of all, given the fact that at the time there is no other project in the team (that is, as far as I know) that focus on the problem of increasing the number of engaged users, there are no useful datasets that can be used, meaning that they were created from the data available in the databases. These databases included:

- Movements database

  Includes information about user's movements in the application (referring to an interaction with the application functionalities). Every row represents an action that a user made at a certain timestamp in the history. Each user is identified by a unique *user id*. The different possible actions are also identified with a unique name that belongs to a specific value proposition in the application. This information is key to determine whether a user has interacted with the app or not. The possible actions are extensive, and not all of them are of interest for the purposes of this project (according to the business experts in the company), however, they are related with the following segments of the app:

Segment	Description
Account Funding	Actions related to the digital account
Credit	Actions related to the credit services
Wallet	Actions related to the digital wallet
Cards	Actions related to the cards (credit/debit)
Online	Actions related to online payments

Table 3. Segments being tracked

  The specific actions will be mentioned later in the Methodology section as not all of them are considered for the project and their selection was based on business decisions.

- Notifications database

  Includes information about **notifications campaigns** that were sent to each user. The application has many channels through which the users can be notified. For this project, however, we will only focus on the so called **notifications** channel. It consists of the messages sent to the users through the application itself. All

notifications belong to a certain **campaign** and this database contains registers for each one of them, for every user. Here, every register corresponds to a campaign sent to a user. Also, the notifications have different stages and each of them would be a new register in the databases, these are presented in **Table 4**. For example, if a user A received a notification from campaign B and he opened it, then there will be three rows for that user (for that campaign, in that datetime): One for the campaign stage **sent**, another for the **shown** status and finally the **open** event.As before, the users are identified with a unique **user id**.

Status	Description
arrived	The campaign arrived at the user's device, but not necessarily viewed
blacklist	The user asosiated to the campaign is excluded from it
control	The user belongs to the campaign's control group
discarded	The user discarded the notification
holdad	
holdre	
open	The user opened the notification
sent	The notification was sent, but not necessarily arrived or viewed
shown	The notification was shown to the user

Table 4. Push message stages

It is important to mention that a notification doesn't necessarily go through all of the mentioned stages. Either because of a tracking failure in the infrastructure or simply because a user had no signal at the moment, some stages might not occur. The only stage that is **always** suppose to happen is the **sent** event because it doesn't depends on the user. In essence, if a campaign was sent, we should be able to see the message exit, past that point, anything could happen. Another important thing to mention is that some stages are mutually exclusive and could not occur at the same time. For example, if a user **discarded** a notification, there will not be an **open** event, that would be impossible.

- Marketplace transactional database

  This database contains information about users transaction **in the ecommerce ecosystem.** Here we find details about the date, type and segment of the transaction, as well as the amount expended and the type of payment. For every transaction there is a register and by grouping them it is also possible to obtain the number of orders for every user id. The following table presents the features considered for later use in the creation of the datasets.

Feature	Description
Type of payment	Method selected by the user for pay in the marketplace
Category bought	Type of product bought in the marketplace according to platform categories
Amount spent (USD/Local currency)	How much did the user spend in the product
Date of the operation	

Table 5. Marketplace's transactional features considered

- Install database

  This table contains the installation history for every user, whether it is the marketplace or the fintech application, this database saves every installation date, as well as a classification to determine what kind of install it was (organic or if there was some kind of intervention).

- Fintech navigational database

  It contains information about the user's behaviour in the payment app. Everything from a view to a click gets registered, for every user id, every day. With this information, it is possible to know what was the user looking at, what was he interested in, what value propositions was he scrolling, etc. Apart from that, this database also registers the time of navigation and type of connection (mobile, desktop or WiFi).

- Marketplace navigational database

  Just like the navigational data from the payment platform, the ecommerce navigational database contains all the information related to the user's activities in the marketplace application. Among the features of interest are the **views** or visits in specific categories that will later be mentioned and the **intentions to buy** in those. Like the rest of the databases, every register is associated with a specific date and user.

The previous databases are the foundation upon which the rest of the datasets (which are going to be used for the later models) will be created. As mentioned before, the objective is to create two solutions or "models" and, like with every artificial intelligence approach, they need certain inputs, in other words, data. For this purpose we developed some **datasets** that were used for both analyze the current situation or **BAU (business as usual)** and develop the two final solutions.

2.1 Information schema

The information used as input for the different models is divided into a series of datasets built from the databases mentioned in the Data section. Some of them, as will be mentioned later in the section of Methodology, exist only to feed a certain part of a model or because they were created as part of an analysis.

In order to understand how the following tables were created, an important aspect of this project must be clarified: **what do we mean by engagement?**

It is a fair question (and a very good one) because, when it comes to defining when a user has installed an app or when has he used it for the first time, it is pretty straightforward, not too much discussion there. However, when does someone effectively engage with an app? In other words, **when does the use of an application becomes a habit?**

According to the Cambridge Dictionary, a habit is ***"something that someone do often and regularly, sometimes without knowing that he is doing it"***. Another possible definition is that it is ***"a particular act or way of acting that someone tend to do regularly"***.

We are getting close, but this definitions do not say anything new. However there is a key element when talking about a habit and it is the regularity. So we know that for something to become a habit, it must be done regularly.

Upon this premise, we would say that a user of the fintech application is engaged when he uses it in a regular way. For this reason, the business experts at the company came up with a definition for us to work with. From different studies and analyses that are out of the scope of this work, they found out that when a user **operates** in the app in **five** different days in a timeframe of **thirty** days since his first operation, then his probabilities of not churning and keep using the app increase significantly from those who fail to operate that number of days. They found out that the users that actually managed to surpass that five operations days barrier, actually ended up spending more and engaging in a much deeper level with the application.

With this definition in mind, from now on, when talking about engagement in the rest of the work it will be referring to a person who has done (or not) those five operations in the specified timeframe.

Next, we will proceed to describe the datasets that were created for the training and understanding of the models.

2.1.1 Available Operations

The whole concept of engagement (or in this case, *habit*) revolves around **using** the app. That means, engaging with the possible operations available for the users. For that reason, before explaining in detail the datasets that allowed us to build our solution, it is important to list the multiple services offered through the app. After this, every time an **operation** is mentioned, it will be referring to one or more of the following activities:

Pillar	First Category	Operation	Description
Money Input	**Money Funding**	**-**	**Users that input money into their app account from another (external) account that belongs to them**
Money Input	Money Funding	TED	Money input through a TED account
Money Input	Money Funding	PIX	Money input through PIX
Money Input	Money Funding	PEC	Money transfer between companies
Money Input	Money Funding	DEBCARD	Money transfer from the user's personal account into the app account through his debit card
Money Input	Money Funding	Portability	Money input as part of the users salary
Money Input	**Money Receiving**	**-**	**Users that input money into their app account from another (external) account that does not belong to them**
Money Input	Money Receiving	TED	Money input through a third person's TED account
Money Input	Money Receiving	PIX	Money input through a third person's PIX account
Money Input	Money Receiving	Ticket / Boleto	Money input through a Boleto that another person emmited
Money Input	Money Receiving	P2P	Money input between accounts of the app
Money Input	Money Receiving	Refund	Money input as part of a refund requested by the user in the ecommerce application
Money Input	**Money Borrowed**	**-**	**Users that input money into their app account by lending money from it**
Money Input	Money Borrowed	PER	Users that input money by requesting a personal loan
Money Input	Money Borrowed	CSR	Users that used credit in order to pay
Payments	**Physical Purchases**	**-**	**Users that buy through the app in a retail location**
Payments	Physical Purchases	QR MP	User pays through a QR code
Payments	Physical Purchases	QR PIX	User pays with the app through PIX account to other institutions by reading a QR Code
Payments	Physical Purchases	ADQ (ACQR)	User pays with a QR code in a Getnet or Cielo app
Payments	Physical Purchases	TC (CC)	User pays with the company's credit card
Payments	Physical Purchases	TD (DC)	User pays with the company's debit card
Payments	**Online Purchases**	**-**	**Users that buy online through the app**
Payments	Online Purchases	Link	The user pays with the app by being redirected to it through a pay link in another web page
Payments	Online Purchases	CHO	The user pays with the app because the site uses the application as a payment method
Payments	Online Purchases	AM ML	The user pays with the money available in their app account
Payments	Online Purchases	CRD ML	The user pays with the credit given to him by the app
Payments	Online Purchases	V. Debit Card / TVD	The user pays with the company's virtual debit card
Payments	Online Purchases	V. Credit Card / TVC	The user pays with the company's virtual credit card
Payments	**In App Purchases**	**-**	**Users that use the app services**
Payments	In App Purchases	Utilities	User pays services through the app (gas, electricity, etc)
Payments	In App Purchases	Recharge	User recharges either his cellphone or his antenna tv through the app
Payments	In App Purchases	Subscriptions	User pays his subscriptions to services like Disney Plus, HBO, etc.
Payments	In App Purchases	Donations	User does donations through the app to a specific NGO
Payments	In App Purchases	Automatic Debit	User pays his bills where the amount is debited automatically
Payments	In App Purchases	Transport	User pays for products of urban mobility for public and individual transport (bilhetagem digital and UltraPasse respectively)
Payments	In App Purchases	Digital Goods	User purchases pre-paid credit in digital services platforms like Google Play, PlayStation, Uber, etc.
Payments	In App Purchases	Delivery	User purchases products in a delivery platform
Payments	**Other In App Purchases**	**-**	**Users that use the app services**
Payments	Other In App Purchases	Insurtech	User purchases one of the company's ensurance
Payments	Other In App Purchases	Critpo	User purchases cripto currency
Payments	**Different Accounts**	**-**	**Users that output money from their app account to an account of different ownership**
Payments	Different Accounts	TED	Users with withdrawal of money in their app account through a TED of different holders.
Payments	Different Accounts	PIX	Users with withdrawal of money in the app account through a PIX of different holders
Payments	Different Accounts	P2P	Money transfer between app accounts

Table 6. App's available operations

The operations are presented in a business like manner, but when talking about how they were actually registered, we have the following set of operations.

Feature	Description
Account Funding	Movements related with the user's account money (in or out)
Portability	Movement related with the input of money as part of a salary
Marketplace operations	Operations registered in the marketplace
Debit Card	Movements done with the debit card
Card - Prepaid	Movements done with a prepaid card
Credit Card	Movements done with a credit card
Credits	Credit loan or payment
Cripto	Criptocurrency transaction
Insurtech 1	Movement related with the company's insurance
Insurtech 2	Movement related with the company's insurance
Insurtech 3	Movement related with the company's insurance
Online Payment 1	Movements done outside the ecosystem but using the fintech platform as a pay method
Online Payment 2	Movements done outside the ecosystem but using the fintech platform as a pay method
Online Payment 3	Movements done outside the ecosystem but using the fintech platform as a pay method
Antenna TV	Movement related with paying the antenna TV service
Recharge	Movement related with paying the cellphone recharge
Delivery	Movement related with paying a delivery service
Digital Goods	Movement related with purchases of pre-paid credit in digital services platforms
Donations	Movement related with doing donations through the app to a specific NGO
Instore	Movement related with paying with a QR code from the app
Recieve Money	Receive money from another user
Send Money 1	Send money to another user
Send Money 2	Send money to another user
Send Money 3	Send money to another user
Others	-
Transport	Movement related with paying for products of urban mobility for public and individual transport
Utilities	Movement related with paying public services

Table 7. App's available operations - Database survey

2.1.2 Habit Journey table

This dataset contains information about those users who got to the aforementioned **habit** state. It tells, for each user, what movement did he do and on what date. This dataset is used for both model development (based on these data, transition matrix were created) and for user analysis.

Feature	Data Type	**Description**
Country	STR	Country of the customer
Customer ID	BIGINT	Customer unique identifier
First Payment	STR	Customer's first operation
First Payment Date	DATE	Customer's first operation date
Second Payment	STR	Customer's second operation
Second Payment Date	DATE	Customer's second operation date
Third Payment	STR	Customer's third operation
Third Payment Date	DATE	Customer's third operation date
Fourth Payment	STR	Customer's fourth operation
Fourth Payment Date	DATE	Customer's fourth operation date
Fifth Payment	STR	Customer's fifth operation
Fifth Payment Date	DATE	Customer's fifth operation date

Table 8. Habit journey table schema

As table 8 shows, every user id is unique in this dataset, meaning that all the information associated with a person is saved in one register. It is important to mention that users do more than these five payment actions, however, this table only keeps track of

those because that is the information required to train or study any model. If it was to include any more movements, it would produce a data leakage problem.

2.1.3 Prospect users table

This table contains the actionable universe, that is, the users that we can reach and affect through the models. It keeps track of the users that have already done their first payment and still have not reach their thirty days time limit nor have completed the five payment days. Because of this, this dataset only retains the last information about each user. That means that for every user id, it will have it's last payment, in what category it was done and the days that have passed since then (this dataset contains unique registers for every user, meaning there are no duplicates in the user id feature). On the other hand, because users are reached through a **push channel** and we are interested in determining when it is better to send a certain campaign, this dataset registers the last campaign that was sent to the user and what category it fell into.

Feature	Data Type	**Description**
Country	STR	Country of the customer
Customer ID	BIGINT	Customer unique identifier
Category of last payment	STR	Last payment category
Number of the last payment	INT	Which operation day was the last payment
Days since last payment	INT	Days passed since last operation
Days since last notification	INT	Days passed since last notification
Category of last notification	STR	Category of the last notification sent

Table 9. Prospect users table

As it always keep the latest status for each user, this dataset is the one being used as input for new predictions every day (the final dataset will be later introduced, as this one is incomplete because it is missing the already mentioned transactional and navigational features).

2.1.4 Master payments table

Similar to the Habit Journey table, this dataset contains information about user's operations. As expected, it could only save up to five different operation days. However the two of them differ in many aspects. The master payments dataset does not have unique registers but multiple entries for the same user, depending on the number of operation days he has up to that point. This means that if a person has operated in three different days, then that user will show in the table three times.

Each line contains information about **what** operation day is being saved at that moment (first, second, etc.) as well as the category and the date it was made. The usual country and user id features are also present in this dataset.

There is also a **flag** feature that determines whether the user is a **seller** or not. Because of the nature of the company (fintech and ecommerce) some users are classified as **individuals** and some others as **sellers**. The main difference between them is that a seller is a person that showed a tendency to sell (whatever kind of product or service, in any of the marketplace or the fintech app) in the late time. The definition is not absolute, and it can change over time. For example, a person that is moving might need

to sell furniture and decorations in a short period of time. By doing so, that person might become a seller for the system. However, after a while, if the user has not shown any kind of activity that is seller related, then it will go back to being a regular individual.

Of course, the previous example does not represent the vast majority of the sellers. In fact, those type of users are not really that important for us. The idea behind the flag is to identify the active ones, that is, shops, e-shops, supermarkets, etc. Those kind of sellers are constantly active, constantly selling, therefore we wish to identify them because they are excluded from the training part of the process as will be explained later in the Methodology section.

Finally, another difference between this dataset and the Habit Journey table is that the first one keeps track of **all** users and their operations, including those that did not reach the **habit** state.

Feature	Data Type	**Description**
Customer ID	BIGINT	Customer unique identifier
Country	STR	Country of the customer
Date of payment	DATE	Date of the operation
Number of payment	INT	Number of operation
Category of the payment	STR	Payment category
Flag Seller	BOOL	Indicator if the user is a seller or an individual

Table 10. Master payments table

2.1.5 Target set table

As mentioned in the Machine Learning section, for any **supervised** model to work, it is important to provide it with the expected output for it to learn from. This table contains the expected output for the **Uplift** model to do that. As will be explained in the next section, this model needs information related to the treatments applied to each user and how they responded to them. This translates in the following features.

Feature	Data Type	**Description**
Customer ID	BIGINT	Customer unique identifier
Country	STR	Country of the customer
Number of payment	INT	Number of operation
Treatment	STR	What treatment (notification) was applied to the user
Output	BOOL	Indicates how the user responded to the treatment (operated in the category sent or not)

Table 11. Target set table

For this kind of models to work, it is necessary to provide the following set of inputs:

- Set of features; **X**
- Applied treatment; **t**
- Resulting output; **y**

The features will be explained in the following subsection of data. When talking about a "treatment", it is normally referred to a certain action that was applied to a population: it could be a discount, a targeted campaign, a new drug, etc. As it can be seen, this model tackles the problem of causality: **is this action going to produce a different response in the user?**

And it is not bound to just one treatment,there could be many of them. In this particular problem, the **t** refers to the all the notifications associated with the possible operations that are present in the app. This is because we are only interested in proposing the options that will genuinely produce an impact on the user.

Finally, the output indicates if the user responded positively or negatively to the treatment, that is, whether he operated in the category he was proposed or not. The definition is very restrictive because it is possible that the user payed in a category different from the one presented to him. In that case, however, we consider that the output is **zero**, meaning that the treatment did not work.

This table was constructed based on the Master payments table, meaning it gathers all historical data about the users that walked through the habit process. For those users, the dataset collects information about the notifications they received in the past. In order to keep coherent with the goal of the model, it only picks up notifications associated with the fintech application (because that is the place where we want the users to look at) and only considers notifications with **shown** status (those the user effectively received in his cellphone). As a disclaimer, we could only consider one status because otherwise the notifications would be duplicated, on the other hand, we did not consider **sent** nor **open** statuses because, for the first one, it does not guarantee that the device successfully received the message. For the case of the **open** status, we decided not to look at those because it would have significantly reduced the number of notifications we could have had as not all the users open the notifications they received. Besides, an open does not guarantee a success, in some cases, the user cannot open it because he is unable to, but could have still seen it (the messages are shown in a pop-up manner), making the **open** status unnecessary.

Following with the construction of the Target set table, having collected all the notifications that were sent in the history, it focus only on the ones that happened in the user's next 30 days since he first operated in the application. As mentioned before, each notification occurs at a specific moment in the user journey to the habit state, and this table illustrates that through the **Number of payment** feature in Table 11, meaning that we are joining the notifications not only by the user but also by the moment when they were sent. This is because it is important to capture the idea of a sequence, of a series of steps. Each step would bring the user closer to the habit state, therefore, it is expected to notice a difference between the treatments (notifications) applied in the early days of the user and the ones at the end of the journey.

Finally, the result to the treatment applied (the **output**) is obtained by looking at the category of the notification sent (see Available Operations) and checking if, in a specific time window called the **attribution window** the user actually operated in it. This information is obtained from the **Movements table**. Of course, the user might operate, but not in the desired category, which is considered as a penalization (output zero), because it was an incorrect message.

The **attribution window** is set at two days (since the notification was shown) for every operation available. That means that, when checking if the output was positive or not, we consider the operations the user did in the following two days of receiving a

notification. Such timeframe was set as a business decision, given the short period of time the user has to reach the habit state.

2.1.6 Feature set table

Based upon the Prospect users table, the **Feature Set table** is a collection of features that were especially gathered as predictor variables for the **Uplift model**, which was already mentioned before. This table, whose features are detailed in table 12, is comprised of **Transactional, Navigational and Notification data** about the user. The information is collected starting from the users that have yet to get to the habit state or are outside of the 30 days time window, in other words, the prospect users. Because the last table is constructed with a single register by user, the feature set is built in the same way. The reader should be reminded that the Prospect users table contains the latest information about the user, so it is reasonable to expect that the features contained in the feature set is also constantly updated to be as near real time as possible.

In terms of aggregation, some features contain the latest data available, without any kind of grouping. Some others, like most of the navigational ones for example, are summed for the latest 7 days. This is because the navigation, views and searches contain behavioural information that allow us to infer the user's preferences. In this subject, it is better to have the latest movements.

In other cases, like some transactional features, we were interested in grouping the data for the last 30 days. This was based mostly on business and infrastructural decisions as we noticed that a week or two would not gather enough data for a model to train.

Feature	Descripcion
CUS_CUST_ID	User unique identifier
SIT_SITE_ID	User's country id
LAST_PAYMENT	Number of the last operation made (from 1 to 4)
DAYS_SINCE_LAST_PAYMENT	Days passed since the last operation
DAYS_SINCE_LAST_PUSH	Days passed since the las notification recieved
LAST_CAMPAIGN_FLOW	Category of the last notification recieved
LAST_PAYMENT_FLOW	Category of the last operation made
N_TRANSACTION_LAST_PAYMENT	Number of transactions done in the last operation day
MAYOR_EXPENSE_DOL	Mayor expense done since user's first payment (USD)
SUM_EXPENSE_DOL	Sumarization of all expenses done since user's first payment (USD)
MAYOR_EXPENSE_FLOW	Category of the mayor expense done since user's first payment
DAYS_SINCE_FIRST_PAYMENT	Days passed since the user's first payment
SPEND_ACCFUND	Total of money spent in the category of Account Funding
SPEND_ACCMONEY	Total of money spent in the category of Account Money
SPEND_CARDS	Total of money spent in the category of Cards
SPEND_PREPAID	Total of money spent in the category of Prepaid Cards
SPEND_CRED_CARD	Total of money spent in the category of Credit Cards
SPEND_INSURTECH	Total of money spent in the category of Insurtech
SPEND_ONL_PAYM	Total of money spent in the category of Online Payments
SPEND_ANT_RECH	Total of money spent in the category of Antena Recharge
SPEND_DIG_GOOD	Total of money spent in the category of Digital Goods
SPEND_INSTORE	Total of money spent in the category of QR
SPEND_MON_TRANSF	Total of money spent in the category of Money Transference
SPEND_TRANSPORT	Total of money spent in the category of Transportation
SPEND_UTILS	Total of money spent in the category of Utilities
SPEND_CELLPHONE	Total of money spent in the category of Cellphone Recharge
N_PAYMENT_VOUCHER_CARD	Number of payments done with a voucher in the marketplace in the last 7 days
N_PAYMENT_DIGITAL_WALLET	Number of payments done with the digital wallet in the marketplace in the last 7 days

Feature	Descripcion
N_PAYMENT_CREDIT_CARD	Number of payments done with credit card in the marketplace in the last 7 days
N_PAYMENT_ACQ_MONEY	Number of payments done with money available in the account in the marketplace in the last 7 days
N_PAYMENT_DEBIT_CARD	Number of payments done with debit card in the marketplace in the last 7 days
N_PAYMENT_TICKET	Number of payments done with a ticket in the marketplace in the last 7 days
N_PAYMENT_BANK_TRANSFER	Number of payments done with transference in the marketplace in the last 7 days
N_PAYMENT_DIGITAL_CURRENCY	Number of payments done with digital currency in the marketplace in the last 7 days
Q_ORDERS	Number of products bought in the marketplace in the last 7 days
SUM_GMV_USD_CELULARES_E_TELEFONES	Total gross merchandise value in the last 7 days for cellphone and telephones categories
SUM_GMV_USD_LIVROS	Total gross merchandise value in the last 7 days for book category
SUM_GMV_USD_AGRO_INDUSTRIA_E_COMERCIO	Total gross merchandise value in the last 7 days for agro category
SUM_GMV_USD_SAUDE	Total gross merchandise value in the last 7 days for health category
SUM_GMV_USD_FILMES_E_SERIADOS	Total gross merchandise value in the last 7 days for movies and shows categories
SUM_GMV_USD_ANTIGUIDADES	Total gross merchandise value in the last 7 days for antiques category
SUM_GMV_USD_XEROX	Total gross merchandise value in the last 7 days for printers category
SUM_GMV_USD_MAIS_CATEGORIAS	Total gross merchandise value in the last 7 days for other categories
SUM_GMV_USD_CALCADOS_ROUPAS_E_BOLSAS	Total gross merchandise value in the last 7 days for clothing category
SUM_GMV_USD_INGRESSOS	Total gross merchandise value in the last 7 days for shows category
SUM_GMV_USD_ALIMENTOS_E_BEBIDAS	Total gross merchandise value in the last 7 days for food and drinks categories
SUM_GMV_USD_ARTE_E_ARTESANATO	Total gross merchandise value in the last 7 days for art category
SUM_GMV_USD_ELETRODOMESTICOS	Total gross merchandise value in the last 7 days for technology category
SUM_GMV_USD_MUSICA	Total gross merchandise value in the last 7 days for music category
SUM_GMV_USD_CAMERAS_E_ACESSORIOS	Total gross merchandise value in the last 7 days for cameras and accesories categories
SUM_GMV_USD_INFORMATICA	Total gross merchandise value in the last 7 days for informatic category
SUM_GMV_USD_BELEZA_E_CUIDADO_PESSOAL	Total gross merchandise value in the last 7 days for personal care and beauty categories
SUM_GMV_USD_ANIMAIS	Total gross merchandise value in the last 7 days for pet category
SUM_GMV_USD_INSTRUMENTOS_MUSICAIS	Total gross merchandise value in the last 7 days for musical instruments category
SUM_GMV_USD_ELETRONICOS_AUDIO_E_VIDEO	Total gross merchandise value in the last 7 days for audio and video electronics category
SUM_GMV_USD_BEBES	Total gross merchandise value in the last 7 days for babies category
SUM_GMV_USD_ACESSORIOS_PARA_VEICULOS	Total gross merchandise value in the last 7 days for vehicle accesories category
SUM_GMV_USD_BRINQUEDOS_E_HOBBIES	Total gross merchandise value in the last 7 days for toys and hobbies category
SUM_GMV_USD_FERRAMENTAS_E_CONSTRUCAO	Total gross merchandise value in the last 7 days for construction tools category
SUM_GMV_USD_LIVROS_REVISTAS_E_HQ	Total gross merchandise value in the last 7 days for magazines category
SUM_GMV_USD_GAMES	Total gross merchandise value in the last 7 days for gaming category
SUM_GMV_USD_JOIAS_E_RELOGIOS	Total gross merchandise value in the last 7 days for jewelery category
SUM_GMV_USD_ESPORTES_E_FITNESS	Total gross merchandise value in the last 7 days for sports and fitness category
SUM_GMV_USD_CASA_MOVEIS_E_DECORACAO	Total gross merchandise value in the last 7 days for house decorations category
SUM_GMV_USD_COLECOES_E_COMICS	Total gross merchandise value in the last 7 days for comic and collections category
COUNT_CELULARES_E_TELEFONES	Number of transactions done in the last 7 days in cellphone and telephone categories
COUNT_LIVROS	Number of transactions done in the last 7 days in books category

Feature	Descripcion
COUNT_AGRO_INDUSTRIA_E_COMERCIO	Number of transactions done in the last 7 days in agro category
COUNT_SAUDE	Number of transactions done in the last 7 days in health category
COUNT_FILMES_E_SERIADOS	Number of transactions done in the last 7 days in movies and shows categories
COUNT_ANTIGUIDADES	Number of transactions done in the last 7 days in antiques category
COUNT_XEROX	Number of transactions done in the last 7 days in printers category
COUNT_MAIS_CATEGORIAS	Number of transactions done in the last 7 days in other categories
COUNT_CALCADOS_ROUPAS_E_BOLSAS	Number of transactions done in the last 7 days in clothing category
COUNT_INGRESSOS	Number of transactions done in the last 7 days in shows category
COUNT_ALIMENTOS_E_BEBIDAS	Number of transactions done in the last 7 days in foods and drinks categories
COUNT_ARTE_E_ARTESANATO	Number of transactions done in the last 7 days in art category
COUNT_ELETRODOMESTICOS	Number of transactions done in the last 7 days in technology category
COUNT_MUSICA	Number of transactions done in the last 7 days in music category
COUNT_CAMERAS_E_ACESSORIOS	Number of transactions done in the last 7 days in cameras and accesories categories
COUNT_INFORMATICA	Number of transactions done in the last 7 days in informatic category
COUNT_BELEZA_E_CUIDADO_PESSOAL	Number of transactions done in the last 7 days in personal care and beauty categories
COUNT_ANIMAIS	Number of transactions done in the last 7 days in pet category
COUNT_INSTRUMENTOS_MUSICAIS	Number of transactions done in the last 7 days in musical instruments category
COUNT_ELETRONICOS_AUDIO_E_VIDEO	Number of transactions done in the last 7 days in audio and video electronics category
COUNT_BEBES	Number of transactions done in the last 7 days in babies category
COUNT_ACESSORIOS_PARA_VEICULOS	Number of transactions done in the last 7 days in vehicle accesories category
COUNT_BRINQUEDOS_E_HOBBIES	Number of transactions done in the last 7 days in toys and hobbies categories
COUNT_FERRAMENTAS_E_CONSTRUCAO	Number of transactions done in the last 7 days in construction tools category
COUNT_LIVROS_REVISTAS_E_HQ	Number of transactions done in the last 7 days in magazines category
COUNT_GAMES	Number of transactions done in the last 7 days in gaming category
COUNT_JOIAS_E_RELOGIOS	Number of transactions done in the last 7 days in jewelery category
COUNT_ESPORTES_E_FITNESS	Number of transactions done in the last 7 days in sports and fitness category
COUNT_CASA_MOVEIS_E_DECORACAO	Number of transactions done in the last 7 days in house decorations category
COUNT_COLECOES_E_COMICS	Number of transactions done in the last 7 days in comic and collections category
SUM_GMV_USD	Total gross mercahndise value in the last 7 days
AVG_GMV_USD	Average gross merchandise value in the last 7 days
LAST_INSTALL_MARKETPLACE	Whether the last registered installation of the ecommerce app was Organic, caused by a notification or by other influence
LAST_INSTALL_FINTECH	Whether the last registered installation of the fintech app was Organic, caused by a notification or by other influence
NUMBER_INSTALLS_MARKETPLACE	Total number of installations for the ecommerce app
NUMBER_INSTALLS_FINTECH	Total number of installations for the fintech app
NAVEG_MONEY_OUT	Flag that indicates if the user navigated in the output money category in the last 7 days
NAVEG_MONEY_CASH_OUT	Flag that indicates if the user navigated in the cash out category in the last 7 days
NAVEG_MONEYOUTVSMONEYOUTCASH	Ratio of navegations in the last 7 days for the money out and cash out categories
NAVEG_UTC_0_6	Share of hours navigated between 0 and 6 UTC time in the day for the last 7 days
NAVEG_UTC_7_12_UTC	Share of hours navigated between 7 and 12 UTC time in the day for the last 7 days
NAVEG_UTC_13_19_UTC	Share of hours navigated between 13 and 19 UTC time in the day for the last 7 days

Feature	Descripcion
NAVEG_UTC_20_23	Share of hours navigated between 20 and 23 UTC time in the day for the last 7 days
NAVEG_VIEW_GEN_CELLPHONE	Share of views in cellphone main menu for the last 7 days
NAVEG_VIEW_GEN_TRANSPORT	Share of views in transport main menu for the last 7 days
NAVEG_VIEW_SERV_CATALOG	Share of views of the service industries catalog for the last 7 days
NAVEG_VIEW_VIEW_GEN_ENV_DIN	Share of views of the sending money category for the last 7 days
NAVEG_VIEW_GEN_QR	Share of views of the scanning QR category for the last 7 days
NAVEG_VIEW_GEN_WEALTH	Share of views of the banking category main menu for the last 7 days
NAVEG_VIEW_QR_CATALOG	Share of views of the QR catalog for the last 7 days
NAVEG_VIEW_QR_MAPA	Share of views of the QR map for the last 7 days
NAVEG_VIEW_WEALTH_TRANSF	Share of views of the money transfer option for the last 7 days
NAVEG_VIEW_WEALTH_MON_IN	Share of views for the input money option for the last 7 days
NAVEG_VIEW_WEALTH_MON_OUT	Share of views for the output money option for the last 7 days
NAVEG_VIEW_WEALTH_CREDIT	Share of views for the credits options for the last 7 days
NAVEG_VIEW_TRAN_ADD_NUMB	Share of views for the cellphone recharge option for the last 7 days
NAVEG_Q_WIFI_CONECTION	Share of connections done through WiFi for the last 7 days
NAVEG_Q_LTE_CONECTION	Share of connections done through LTE standar (Cellphone data) for the last 7 days
NAVEG_Q_DESKTOP_CONECTION	Share of connections done through a desktop connection for the last 7 days
NAVEG_EVENT_SHORT_BENEFITS_DISCOUNTS	Number of times the user explored the benefits section for the last 7 days
NAVEG_ELAPSED_DAYS_DISCOUNT	Minimum number of days passed since a user explored the benefits section in the last 7 days
ACCOUNT_MONEY_AMOUNT	Current money available in the user's account (not counting cripto currency)
FLAG_SELLER	Flag to identify the user as a seller
FLAG_CREDIT	Flag to identify if the user has asked for a credit
DAYS_SINCE_NAVEG_MARKETPLACE	Number of days passed since the user last navigation in the marketplace
NAVEG_VIEW_PRODUCTS_CELULARES_E_TELEFONES	Number of times the user viewed products in the cellphone and telephone category in the 30 days
NAVEG_VIEW_PRODUCTS_LIVROS	Number of times the user viewed products in the book category in the last 30 days
NAVEG_VIEW_PRODUCTS_AGRO_INDUSTRIA_E_COMERCIO	Number of times the user viewed products in the agro category in the last 30 days
NAVEG_VIEW_PRODUCTS_SAUDE	Number of times the user viewed products in the health category in the last 30 days
NAVEG_VIEW_PRODUCTS_FILMES_E_SERIADOS	Number of times the user viewed products in the movies and shows category in the last 30 days
NAVEG_VIEW_PRODUCTS_ANTIGUIDADES	Number of times the user viewed products in the antiques category in the last 30 days
NAVEG_VIEW_PRODUCTS_XEROX	Number of times the user viewed products in the printers category in the last 30 days
NAVEG_VIEW_PRODUCTS_MAIS_CATEGORIAS	Number of times the user viewed products in the other categories classification in the last 30 days
NAVEG_VIEW_PRODUCTS_CALCADOS_ROUPAS_E_BOLSAS	Number of times the user viewed products in the clothing category in the last 30 days
NAVEG_VIEW_PRODUCTS_INGRESSOS	Number of times the user viewed products in the shows category in the last 30 days
NAVEG_VIEW_PRODUCTS_ALIMENTOS_E_BEBIDAS	Number of times the user viewed products in the food and drinks category in the last 30 days
NAVEG_VIEW_PRODUCTS_ARTE_E_ARTESANATO	Number of times the user viewed products in the art category in the last 30 days
NAVEG_VIEW_PRODUCTS_ELETRODOMESTICOS	Number of times the user viewed products in the technology category in the last 30 days
NAVEG_VIEW_PRODUCTS_MUSICA	Number of times the user viewed products in the music category in the last 30 days
NAVEG_VIEW_PRODUCTS_CAMERAS_E_ACESSORIOS	Number of times the user viewed products in the cameras and accesories category in the last 30 days
NAVEG_VIEW_PRODUCTS_INFORMATICA	Number of times the user viewed products in the informatics category in the last 30 days
NAVEG_VIEW_PRODUCTS_BELEZA_E_CUIDADO_PESSOAL	Number of times the user viewed products in the beauty and personal care category in the last 30 days
NAVEG_VIEW_PRODUCTS_ANIMAIS	Number of times the user viewed products in the pet category in the last 30 days
NAVEG_VIEW_PRODUCTS_INSTRUMENTOS_MUSICAIS	Number of times the user viewed products in the musical instruments in the last 30 days
NAVEG_VIEW_PRODUCTS_ELETRONICOS_AUDIO_E_VIDEO	Number of times the user viewed products in the audio and video electronics category in the last 30 days
NAVEG_VIEW_PRODUCTS_BEBES	Number of times the user viewed products in the babies category in the last 30 days
NAVEG_VIEW_PRODUCTS_ACESSORIOS_PARA_VEICULOS	Number of times the user viewed products in the vehicle accesories category in the last 30 days
NAVEG_VIEW_PRODUCTS_BRINQUEDOS_E_HOBBIES	Number of times the user viewed products in the toys and hobbies category in the last 30 days
NAVEG_VIEW_PRODUCTS_FERRAMENTAS_E_CONSTRUCAO	Number of times the user viewed products in the construction tools category in the last 30 days
NAVEG_VIEW_PRODUCTS_LIVROS_REVISTAS_E_HQ	Number of times the user viewed products in the magazines category in the last 30 days
NAVEG_VIEW_PRODUCTS_GAMES	Number of times the user viewed products in the gaming category in the last 30 days

Feature	Descripcion
NAVEG_VIEW_PRODUCTS_JOIAS_E_RELOGIOS	Number of times the user viewed products in the jewelery category in the last 30 days
NAVEG_VIEW_PRODUCTS_ESPORTES_E_FITNESS	Number of times the user viewed products in the sports and fitness category in the last 30 days
NAVEG_VIEW_PRODUCTS_CASA_MOVEIS_E_DECORACAO	Number of times the user viewed products in the house decorations category in the last 30 days
NAVEG_VIEW_PRODUCTS_COLECOES_E_COMICS	Number of times the user viewed products in the comic and collections category in the last 30 days
NAVEG_BUY_INTENTIONS_CELULARES_E_TELEFONES	Number of times the user intended to buy a product in the cellphone and telephone category in the last 30 days
NAVEG_BUY_INTENTIONS_LIVROS	Number of times the user intended to buy a product in the book category in the last 30 days
NAVEG_BUY_INTENTIONS_AGRO_INDUSTRIA_E_COMERCIO	Number of times the user intended to buy a product in the agro category in the last 30 days
NAVEG_BUY_INTENTIONS_SAUDE	Number of times the user intended to buy a product in the health category in the last 30 days
NAVEG_BUY_INTENTIONS_FILMES_E_SERIADOS	Number of times the user intended to buy a product in the movies and shows category in the last 30 days
NAVEG_BUY_INTENTIONS_ANTIGUIDADES	Number of times the user intended to buy a product in the antiques category in the last 30 days
NAVEG_BUY_INTENTIONS_XEROX	Number of times the user intended to buy a product in the printers category in the last 30 days
NAVEG_BUY_INTENTIONS_MAIS_CATEGORIAS	Number of times the user intended to buy a product in the other categories classification in the last 30 days
NAVEG_BUY_INTENTIONS_CALCADOS_ROUPAS_E_BOLSAS	Number of times the user intended to buy a product in the clothing category in the last 30 days
NAVEG_BUY_INTENTIONS_INGRESSOS	Number of times the user intended to buy a product in the shows category in the last 30 days
NAVEG_BUY_INTENTIONS_ALIMENTOS_E_BEBIDAS	Number of times the user intended to buy a product in the food and drinks category in the last 30 days
NAVEG_BUY_INTENTIONS_ARTE_E_ARTESANATO	Number of times the user intended to buy a product in the art category in the last 30 days
NAVEG_BUY_INTENTIONS_ELETRODOMESTICOS	Number of times the user intended to buy a product in the technology category in the last 30 days
NAVEG_BUY_INTENTIONS_MUSICA	Number of times the user intended to buy a product in the music category in the last 30 days
NAVEG_BUY_INTENTIONS_CAMERAS_E_ACESSORIOS	Number of times the user intended to buy a product in the cameras and accesories category in the last 30 days
NAVEG_BUY_INTENTIONS_INFORMATICA	Number of times the user intended to buy a product in the informatics category in the last 30 days
NAVEG_BUY_INTENTIONS_BELEZA_E_CUIDADO_PESSOAL	Number of times the user intended to buy a product in the beauty and personal care category in the last 30 days
NAVEG_BUY_INTENTIONS_ANIMAIS	Number of times the user intended to buy a product in the pet category in the last 30 days
NAVEG_BUY_INTENTIONS_INSTRUMENTOS_MUSICAIS	Number of times the user intended to buy a product in the musical instruments in the last 30 days
NAVEG_BUY_INTENTIONS_ELETRONICOS_AUDIO_E_VIDEO	Number of times the user intended to buy a product in the audio and video electronics category in the last 30 days
NAVEG_BUY_INTENTIONS_BEBES	Number of times the user intended to buy a product in the babies category in the last 30 days
NAVEG_BUY_INTENTIONS_ACESSORIOS_PARA_VEICULOS	Number of times the user intended to buy a product in the vehicle accesories category in the last 30 days
NAVEG_BUY_INTENTIONS_BRINQUEDOS_E_HOBBIES	Number of times the user intended to buy a product in the toys and hobbies category in the last 30 days
NAVEG_BUY_INTENTIONS_FERRAMENTAS_E_CONSTRUCAO	Number of times the user intended to buy a product in the construction tools category in the last 30 days
NAVEG_BUY_INTENTIONS_LIVROS_REVISTAS_E_HQ	Number of times the user intended to buy a product in the magazines category in the last 30 days
NAVEG_BUY_INTENTIONS_GAMES	Number of times the user intended to buy a product in the gaming category in the last 30 days
NAVEG_BUY_INTENTIONS_JOIAS_E_RELOGIOS	Number of times the user intended to buy a product in the jewelery category in the last 30 days
NAVEG_BUY_INTENTIONS_ESPORTES_E_FITNESS	Number of times the user intended to buy a product in the sports and fitness category in the last 30 days
NAVEG_BUY_INTENTIONS_CASA_MOVEIS_E_DECORACAO	Number of times the user intended to buy a product in the house decorations category in the last 30 days
NAVEG_BUY_INTENTIONS_COLECOES_E_COMICS	Number of times the user intended to buy a product in the comic and collections category in the last 30 days
NAVEG_Q_QUESTIONS_CATALOG	Number of times the user viewed the questions catalog
MARKETPLACE_NAVEG_WKEND_SHARE	Share of days the user navigated in the weekends in the last 30 days
DAYS_SINCE_LAST_OPEN_NOTIF	Days passed since the last time the user opened a notification
MOST_OPEN_NOTIF_FLOW	Most opened notification category

Table 12. Feature set table

2.2 Data analysis

Data analysis and exploration is an essential aspect of every machine learning project like this one. It helps both to check the integrity of the data and to understand the problem at hand.

For the most part, these analyses were done with the whole history of data, all the way back to 2010 in some cases. However, because the company, the application and the users have changed over the years, some parts were studied with the most recent data (a timeframe of a year and six month from the current date)

Another important thing to mention is that although the application is present in

many countries, the analysis and development of the solutions were done only for one. For this we chose the one with the biggest volume of data, coincidentally this country works as a testing field before launching anything new into production at a massive scale.

2.2.1 Frame the problem: Current status

Before jumping into colorful charts, it is necessary to understand what is the situation at the moment. **What is the rate of users that *actually* manage to get to the habit state?**

2.2.1.1 Habit rate

As it was mentioned in the Justification section, the user's lifecycle begins with him **installing** the application, followed by his first interaction with it or **activation**. It is the moment when a user first realizes the value of the product. Some call it an "aha moment" or "the eureka effect." From a user's perspective, it's when they figure out how to benefit from the app and from the business's perspective, it's when the first conversion happens [1]. In this case, we consider an interaction as an **operation**. After that, and as we already know, the clock starts ticking and by the end of thirty days, he might reach to the desired habit state or not.

In the industry, it is pretty normal for the numbers to reduce as we move along the life-cylce. That is, the number of users who activate will be more than the number of users who actually manage to get to the habit state, and, at the same time, those numbers will be higher than the volume of users who get to be engaged, and so on. That is because the further the user moves through the life-cylce, the better he is for the industry, and of course, the ideal user is pretty rare.

As a reference, here are the activation rates by industry:

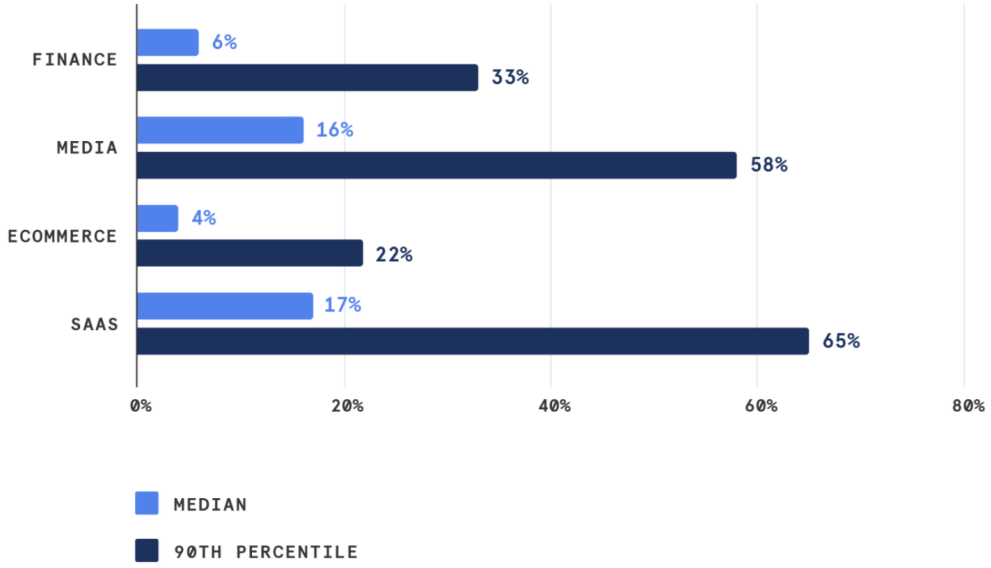


Figure 9. Activation Rate by Industry [1]

As it can be seen, Finance/Fintech and Ecommerce users require more time or convincing, judging by their low activation rate.

The reason why this is mentioned is because, for privacy policy, the installation and activation status cannot be exposed, only the habit rate, which, in the end, is the metric we care about when it comes to context comprehension.

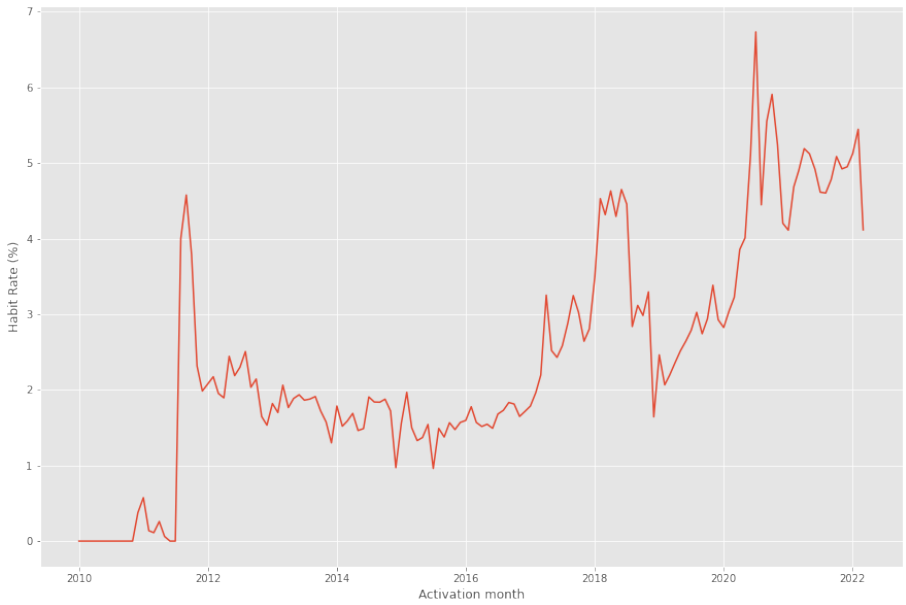


Figure 10. Habit Rate by activation month

The timeline in Figure 10 represents the **habit rate** for the different cohorts of activation. That means that, for a certain month, for example, January 2022, from all the users who **activated** (made their first operation in the application) in that month, the proportion that got to the habit state is represented in the y-axis.

The first thing to notice is that since the year 2020 there is a tendency of continuous growth. This phenomenon can be attributed to the global pandemic which forced everyone to be inside and depend much more on their phones for doing their payments and all the rest of their operations.

As it can be seen, the rates are not static but rather change over time, even from month to month. However, when looking at the big picture, it is clear that the rates are rising. When looking up close - the last year, with daily rates- we can see that the trend continues (Figure 11). This timeframe can be used to obtain the average habit rate, with a 90% confidence interval.

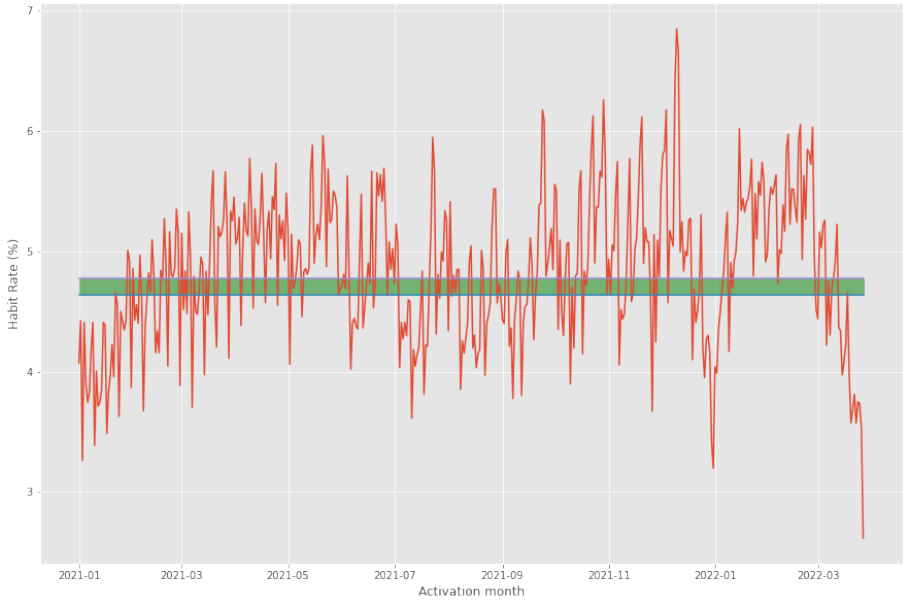


Figure 11. Average habit rate

Estimator	Result
Average Habit Rate	4.7%
Interval	(4.63%, 4.78%)

Table 13. Average habit rate confidence interval

The plot presented above shows the same information as Figure 10, but this one presents the daily results for the last year (red line) and the confidence interval for the average habit rate (green area). These last results are presented in Table 13. From these, we can conclude that the percentage of users who manage to do five operations in the first thirty days after the activation is 4.7%, setting a benchmark to which we can compare when developing our solutions.

2.2.1.2 Habit Speed - How long does it take to get there?

Another important piece of information to study is the "speed" at which the users are arriving at the aforementioned habit state. In other words, **how many days does it take to a user to do five operations?** Of course, given the nature of the event, this value will be between 4 and 29 days, no more, no less. The idea is to identify a pattern that might indicate whether most of the users focus their first operations in the primary days of activating the app or if they are late bloomers and wait until a few weeks after. This would help with the communication strategy: **when should we send a notification?**

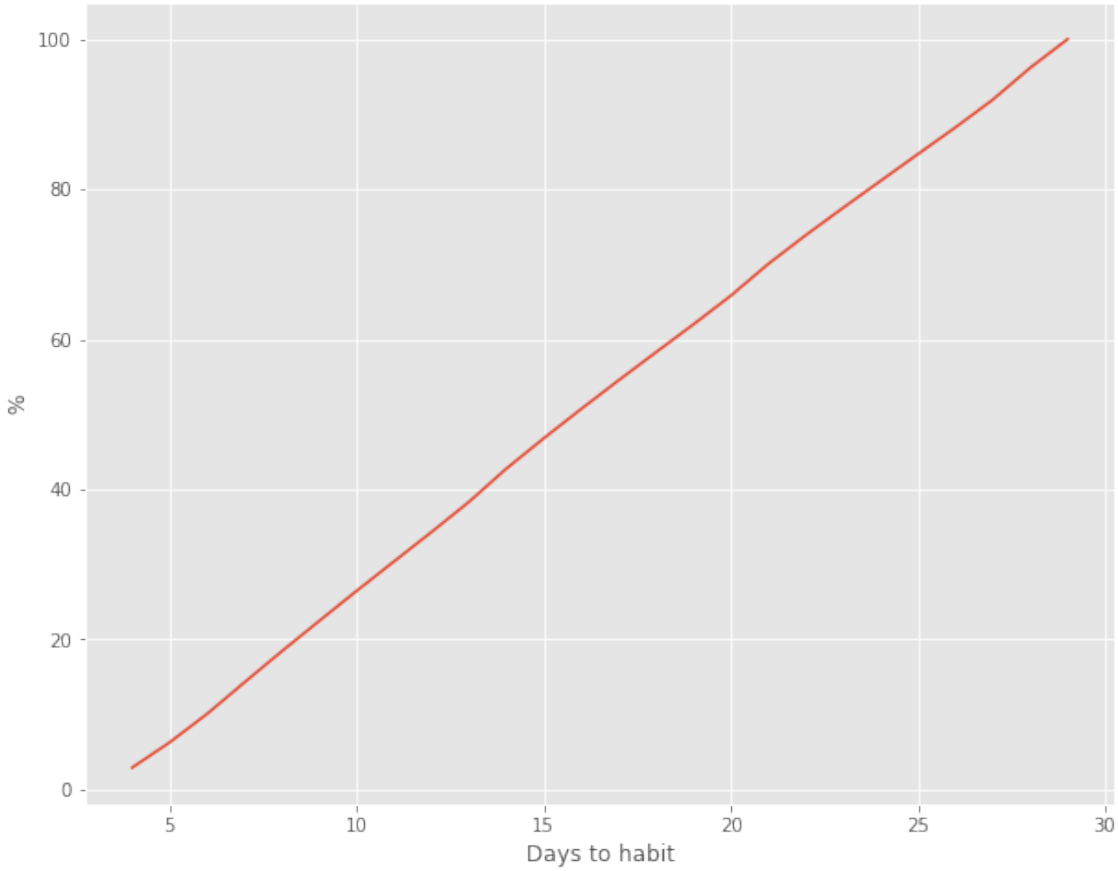


Figure 12. Days to habit distribution

Estimator	Result
Average days to habit	16.518%
Interval	(16.508%, 16.528%)

Table 14. Average days to habit

From Figure 12 we see the number of days passed since the first and fifth payment (x-axis) and the proportion of the population that got to the habit state in that number of days (y-axis). The relation is fairly lineal, indicating that it is evenly distributed. What it means is that there is no concentration in any part of the timeframe of thirty days, all the users have the same chance to get to the habit state in 4, 5, 15 or 29 days.

Knowing that distribution, it is not surprising that the average days it takes a user to get to the five operations is 16 (with a 99% of confidence), which would indicate a frequency of one payment every 3 days.

2.2.1.3 Time between operations

It has already been established that *time* is of the essence for this problem. There is a limited time in which we can influence our users. This is why it is so important to suggest good options that will most likely have a positive impact.

In the previous section we measured the average number of days it takes a user to do all five operations required to reach the habit state. Now, we would like to go deeper

and understand **how many days pass from one operation to the other**. This is interesting as it can be used for determining when do we have the best chance of reaching the user.

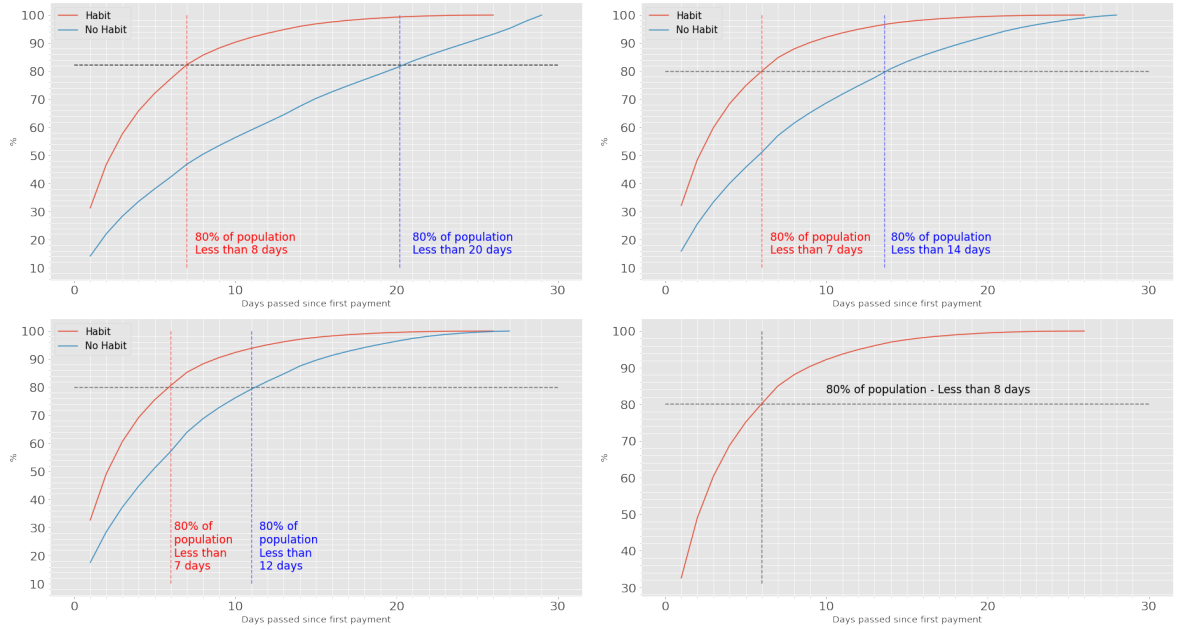


Figure 13. Time passed from one operation to the next

Habit	Estimator	From 1º to 2º operation day	From 2º to 3º operation day	From 3º to 4º operation day
No	**Average days**	10.8230	8.2038	6.9543
	Interval	(10.8196, 10.8264)	(8.1992, 8.2085)	(6.9473, 6.9613)
Yes	**Average days**	4.3739	4.0733	4.0121
	Interval	(4.3701, 4.3776)	(4.0699, 4.0768)	(4.0087, 4.0156)

Table 15. Average days between payments - Habit vs Non Habit Users

Figure 13 tries to answer the question placed above, **how much time passes since a user does an operation until he does another one?**. The graphs represent how is the population distributed when looking at the number of days that passed since two consecutive operations were done. A plot was done for every "step" in the user's journey to become a habit user: top left illustrates the passing from the 1º to the 2º operation day, top right from 2º to the 3º, bottom left from 3º to 4º, and finally, bottom right represents the passing from the 4º to the 5º operation day.

The study was carried out separately for users who reached the state of habit (red line) and those who did not (blue line). The last plot has no blue line because it is the last operation day, if there are any users at this point, they reached the desired state.

Just by looking at Figure 13, it is clear that both groups behave very differently. For starters, it is safe to say that the people who get to the habit state move **faster** than those who do not. As pointed in all plots, around 80% of the population that reached the habit state did two consecutive operations or movements in less than 8 or 7 days. That is, in just a week, 80% of the users used the app twice. On the other hand, the time between movements for the people who did not reach the habit state is longer, and it varies depending on how many operations they did before. The more they do, the faster they will move to the next one.

Table 15 summarises the conclusions by presenting the average number of days between operation days. As mentioned before, those users who achieved engagement take, in average, 4 days to do two consecutive movements, whereas the users who did not make it start off slowly (almost 11 days) but as they make more movements, the average time gets smaller.

2.2.1.4 Time Matrices

From the previous analysis we have seen that both groups -habit and non habit users- behave differently when it comes to their timing. In other words, both type of users take different times to do every sequence of movements.

Following that line of though, it would be interesting to see how does this time difference occurs for each available operation in the application. Below we present the **Time Matrices** which estimate the average number of days passed for every succession of operations.

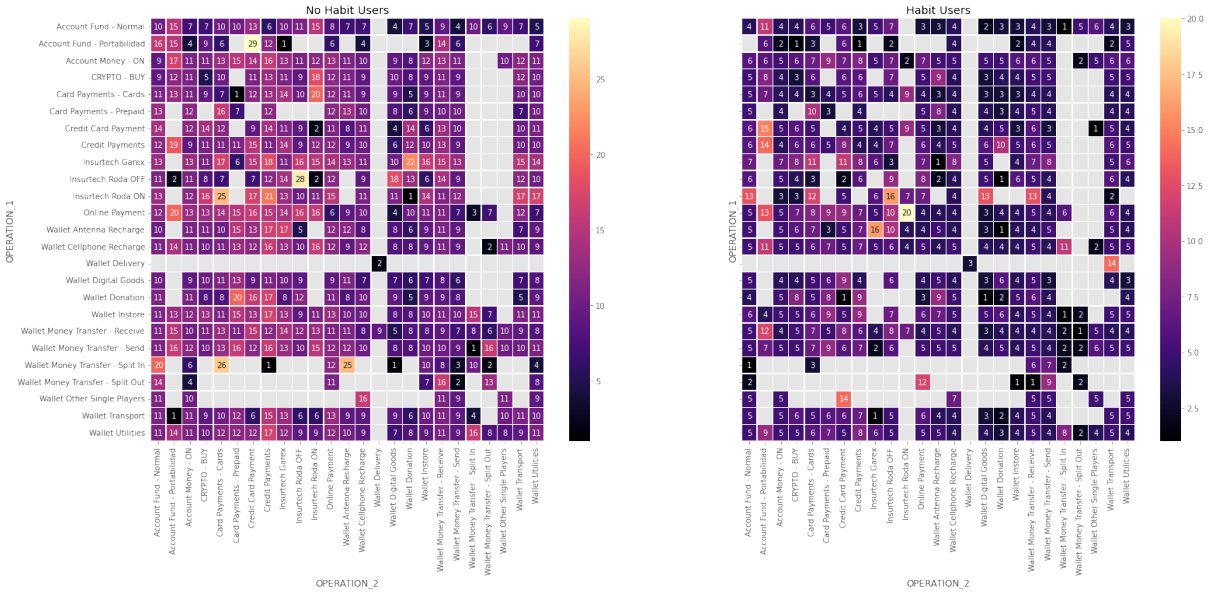


Figure 14. Time matrix - From 1ª to 2ª operation

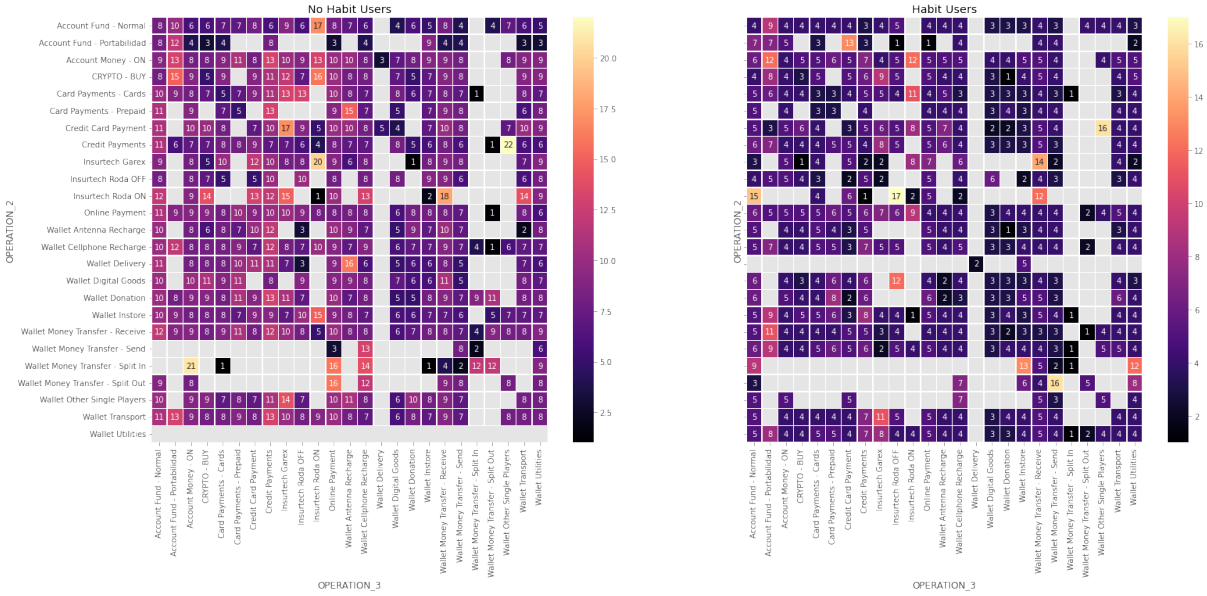


Figure 15. Time matrix - From 2ª to 3ª operation

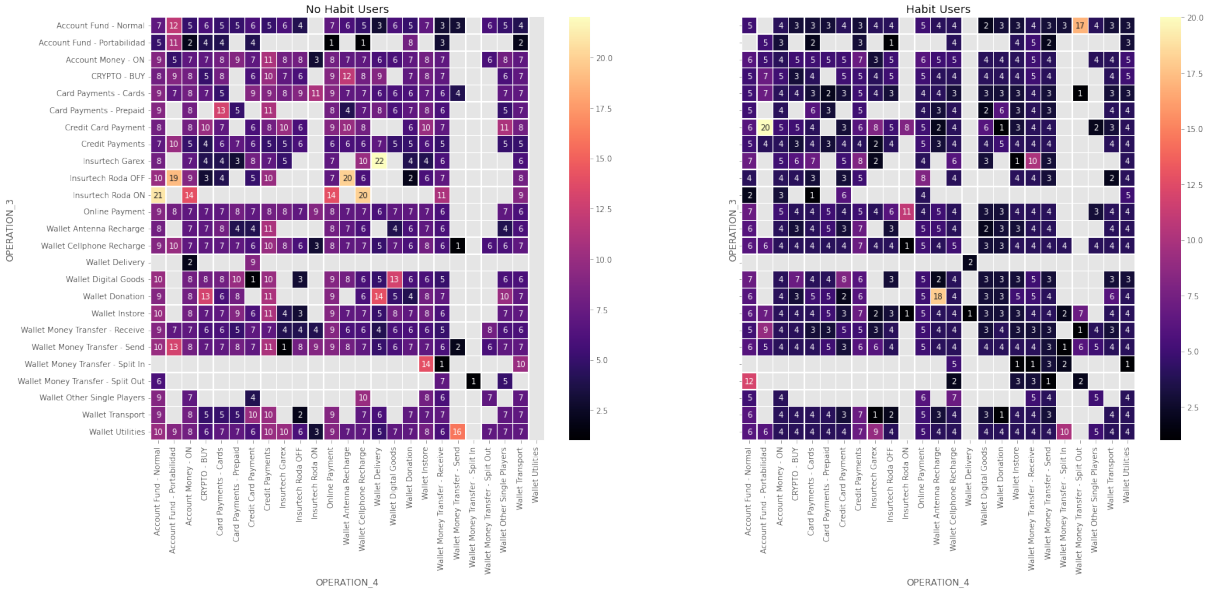


Figure 16. Time matrix - From 3ª to 4ª operation

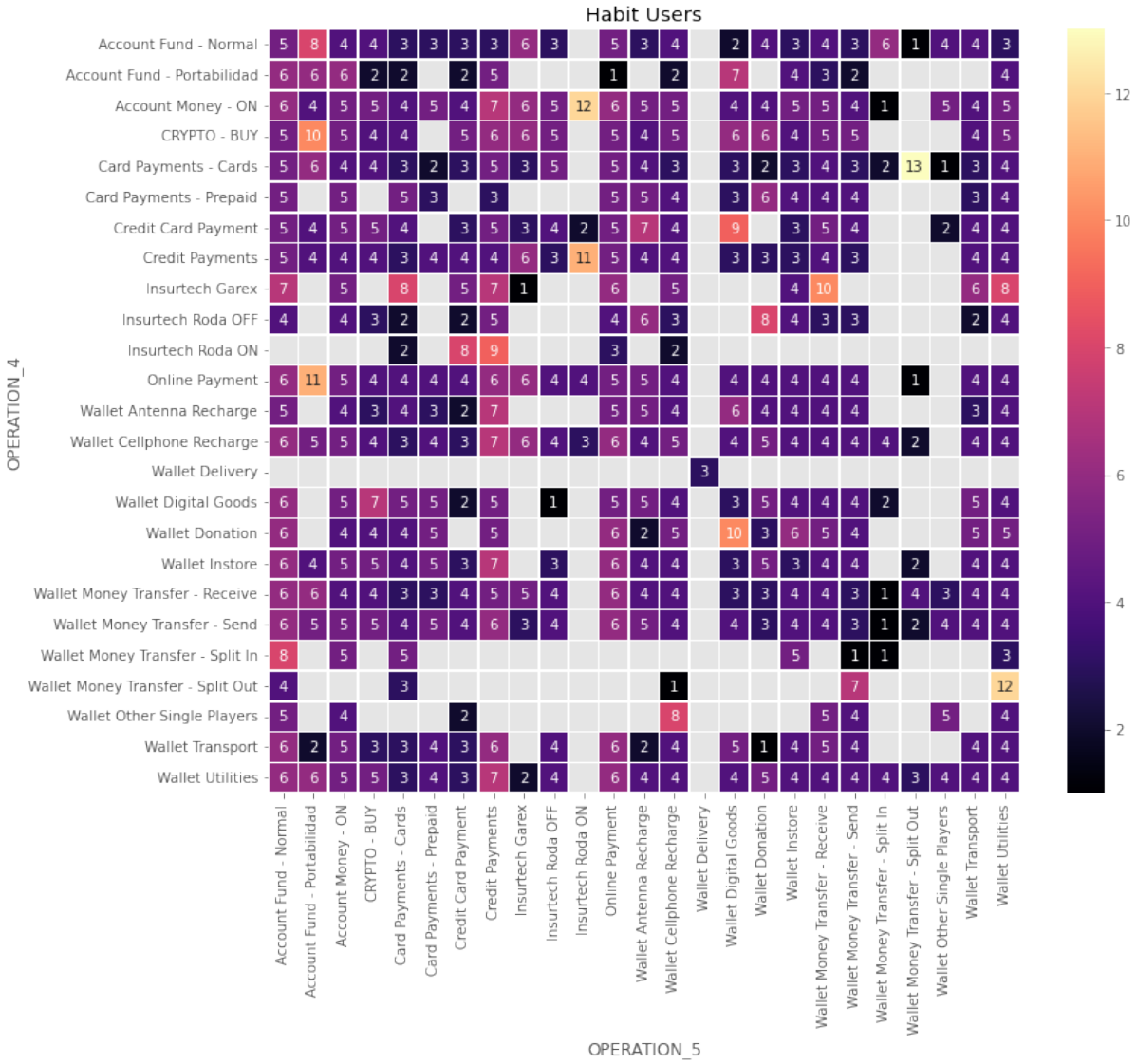


Figure 17. Time matrix - From 4ª to 5ª operation

What figures 14, 15, 16 and 17 are showing is the comparison between the two groups (Habit vs Non Habit users) when looking at the average time that passes between two operations. This is done for every "step" of the way to the fifth and final operation of the 30 days timeframe window.

The reader might notice that some cells in the matrix are blank. That is because not all the interactions between operations have enough occurrences in the real world, and because of that, there is not enough data to estimate the average time.

The color scale goes from darker colors to represent smaller values and brighter colors for bigger ones. Although the numbers are not quite clear, the color visuals can help identify that **Habit** group matrices are darker than **Non Habit** ones, meaning that the average time between payments are lower for the first ones than the latest in general.

Below we present three matrices that compare the average time between the groups, coloring with green those cases where the average time is higher for the **Habit** group and a blue where it is lower.

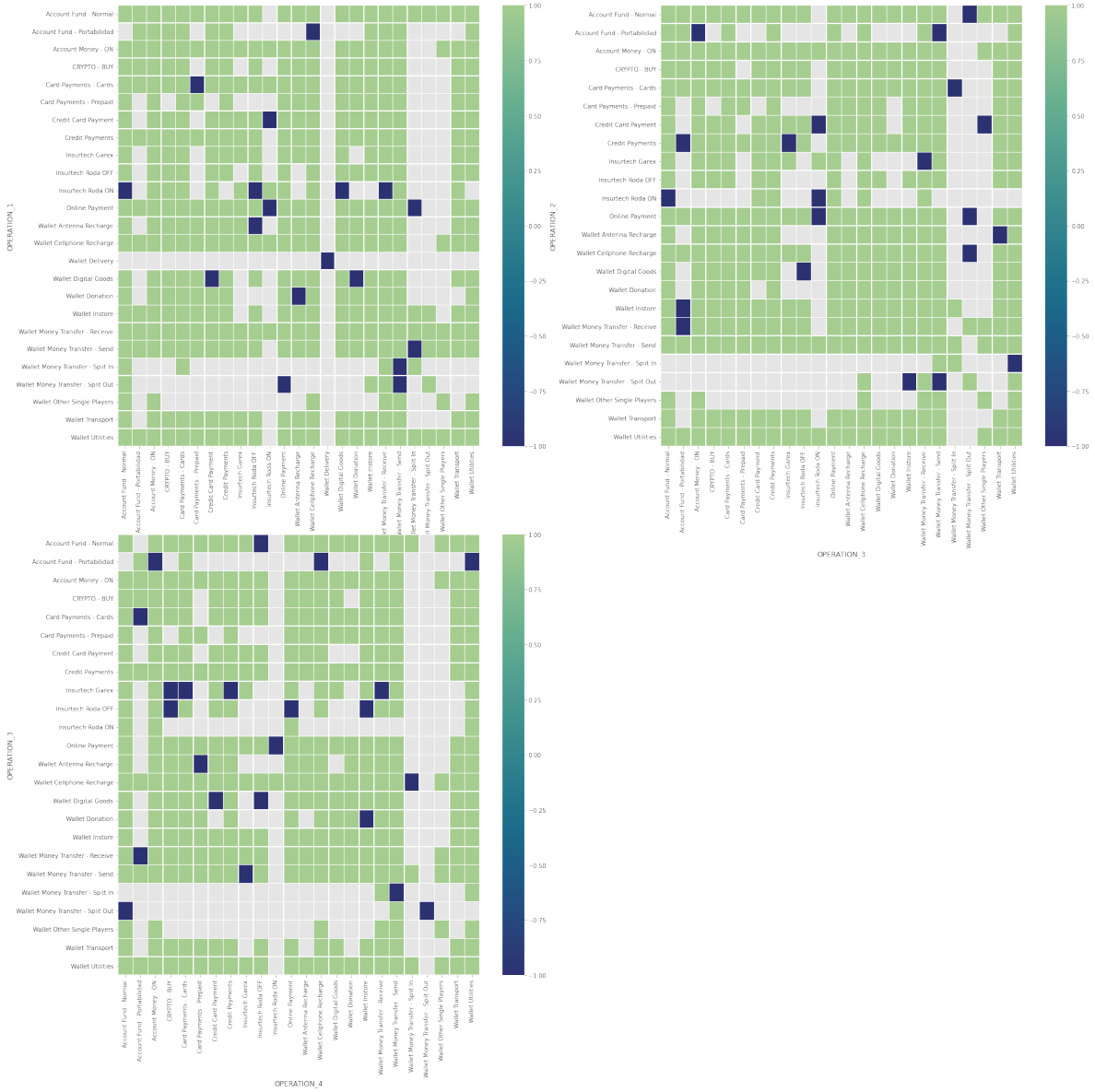


Figure 18. Time matrix comparison

As with Figures 14 to 17, this plots puts into evidence the big difference between the two groups. The fact that almost all the matrix is green shows an enormous time difference, meaning that users who get to the habit state are indeed "faster" at operating than those who do not get there.

Again, this information is valuable as it help us to tell apart the two types of users. Whenever we see someone operating faster than the rest, then there are big chances that he will keep doing them until reaching the habit state.

2.2.1.5 Common Trajectories

Continuing with the exploration of the current status, we wish to understand how does the users get to the habit state, that is, **what are the paths that the users take until reaching the habit state?**

With this knowledge, we would have a full understanding on the journey that the users take, how they move, where they get stuck. With this, the more important operations

will come into light, giving a hint on where are the biggest business opportunities.

Collecting information for the last 6 months about all the users that activated their app (made one operation with it), the following diagram was made:

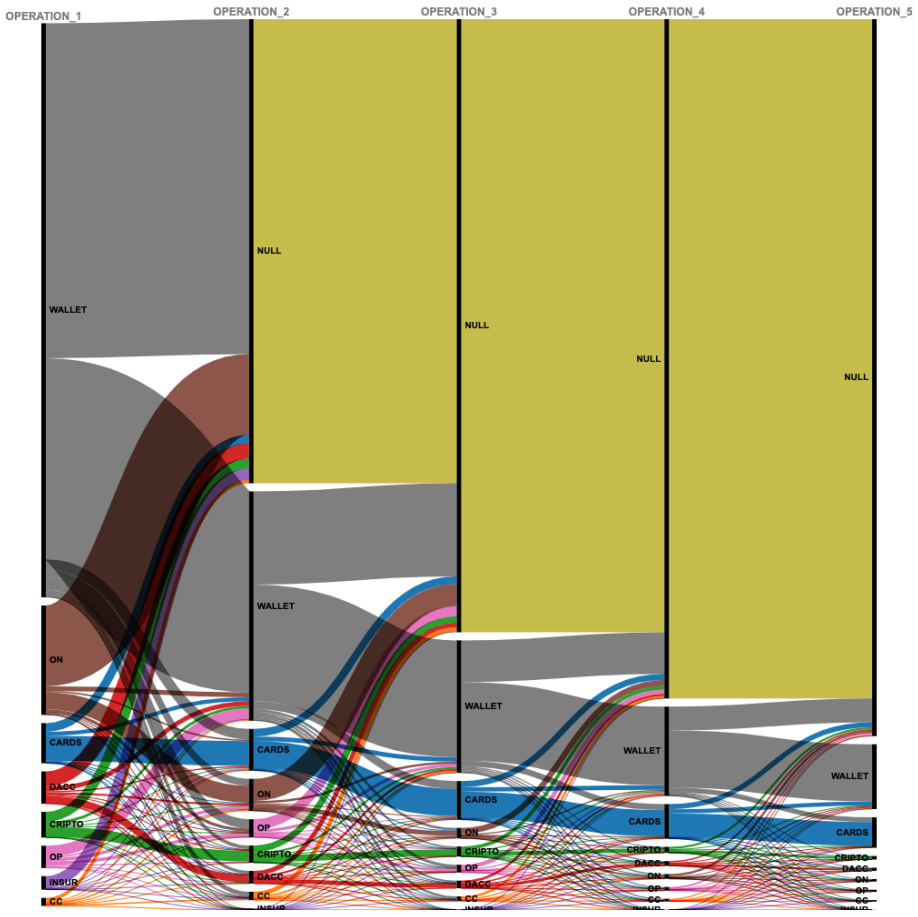


Figure 19. Habit trajectories

Figure 19 presents the distribution of the operations for every one of the five steps necessary to get to the habit state. It is important to mention that the movements were grouped for plot simplicity. If the original operations were to be kept, the abundance of them would have made the diagram illegible for the reader. The presented groups are:

Group	Description
Wallet	All operation related with the fintech digital wallet
DACC	All operations related with funding the digital account
CC	All operations related with credit payments
CRIPTO	All operations related with cripto currency
CARDS	All operations related with payment cards
ON	All operations realized in the marketplace
OP	All operations done in the online world using the app as the payment method
INSUR	All operations related with ensurance

Table 16. Simplified operations table

The lines in Figure 19 that do not have a label represent a *null* movement. In other words, those are the users that did not do anything after a certain movement. We see

that it is very common for the users to end up doing nothing after their first operation, indicating that users do not usually end up engaging with the app, according to the company's engagement definition. One of the biggest insight we can get from Figure 19 is that 56.16% (1.501.771 users) are *one shooters* (Figure 20), that is, they only use the app once in a month, maybe out of curiosity or need, but the fact is that they do not come back to it after their first usage. Of course, that percentage increases when we start looking at subsequent operations as it is rarer for a person to do multiple operations in such a short period of time. In terms of opportunity, the biggest focus should be on those 56.15% of users that never return to the app as we have previously demonstrated that once a user starts "moving" it gets easier for him to do the rest of the operations.

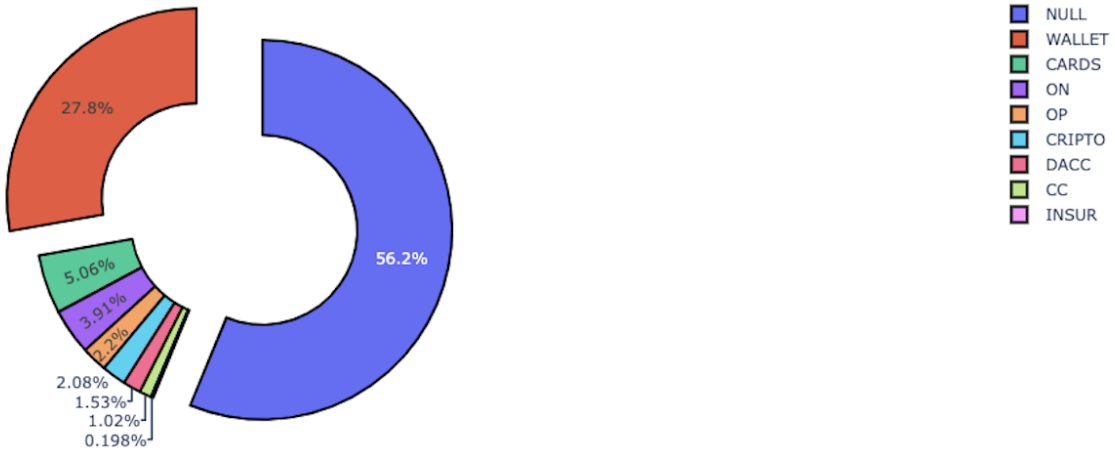


Figure 20. First two operations composition

On the other hand, figure 20 presents the composition of the second operation the users do. In it, we see that most of the user's second movement is related to a *Wallet* operation. This makes sense as that category is the oldest of the available operations that exists in the ecosystem.

Finally, when looking at the most common trajectories that the users are taking to get to the habit state, two routes stand out among the rest. In one hand, 6% of the total users do their five operations in the same category, which is **Wallet**. On the other hand, 1.3% also do it by using the same category, but in this case in the **Cards** category. It might not seem like a big number of cases, but what it tells us is that in order to get the users to do five movements, we should encourage them to stick to the category they started their journey in. However, it is always important to remind the reader that these categories are an aggregation of many others, so what it might seem as one movement classified as *Wallet*, in reality could compose of many other sub-operations.

Up next, we present the *full* trajectories diagram, purely for informative purposes. In this case, the operations were not clustered into groups, but were considered individually. Just like Figure 19, the biggest part of the population is concentrated in the *Nulls* movement, in other words, there is a big part of the users that do not get to the end of the journey. It is also possible to observe the same behaviour as before, where the two most "popular" operations are those associated with the **Wallet** (all the gray trajectories) and

Cards (violet trajectories). As for the rest, it has been established that there are multiple possible combinations and routes that the users could travel, some of them more likely than the others, the endgame of this project is to suggest our users the best one for them.

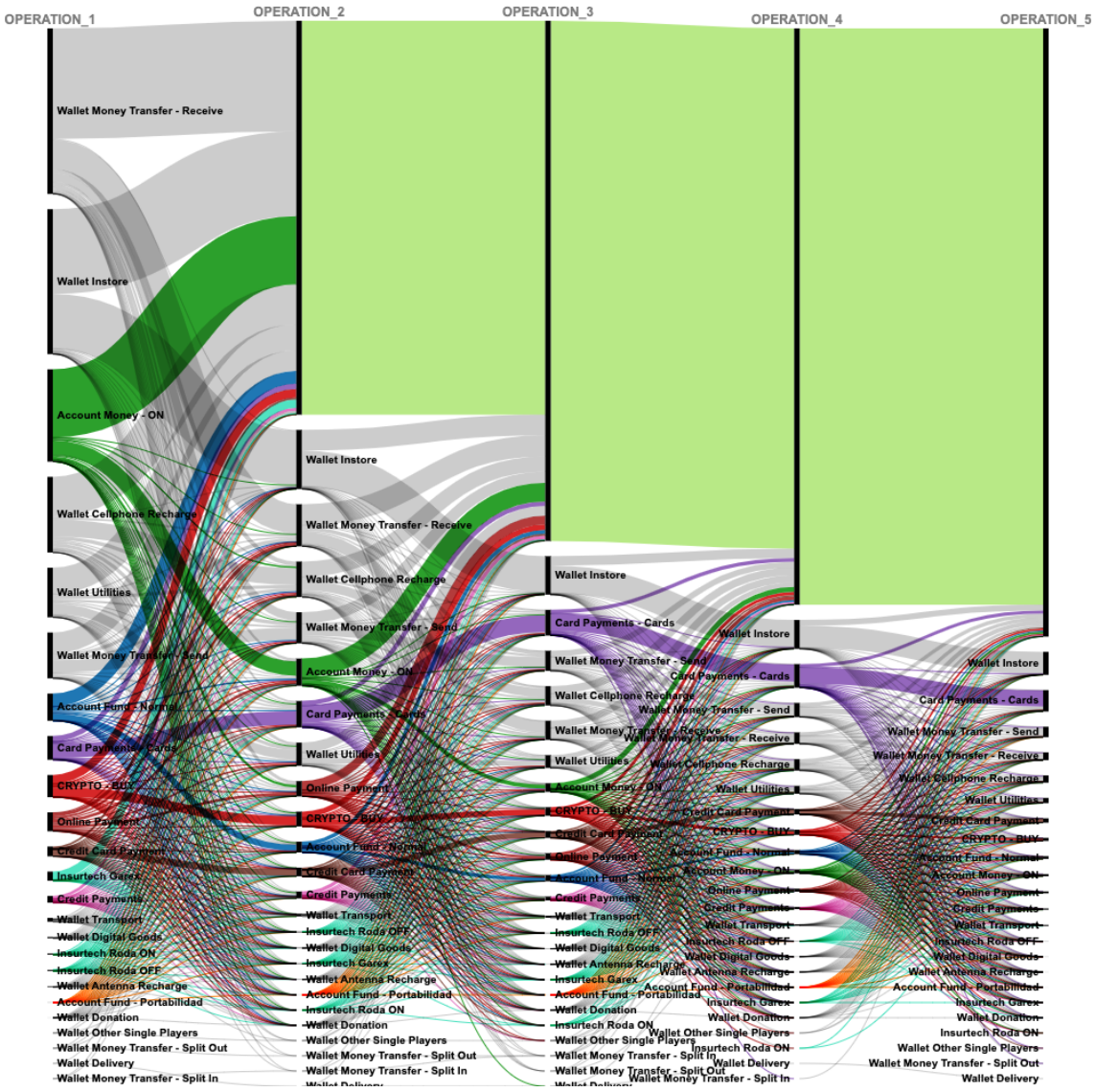


Figure 21. Full Habit trajectories

2.2.1.6 Transition Matrix

Related to the trajectories the users do, we wish to understand **what are the probabilities that a person do an operation *j*, given that he already did operation *i***. In mathematical terms, we wish to understand:

$$P_{ij} = P(Y = j|X = i) \tag{1}$$

Where X and Y are called **states**, and i,j belong to a **state space** of values that they can take. In this case, the possible values are the available operations mentioned earlier.

To solve this, the following pages present the matrices, for every set of operations, that contain the probabilities explained before. Each row is considered the **entry operation** or previous movement. On the other hand, each column represent the **output operation**

or the movement that a user could do next. The values present in figures 22, 23, 24 and 25 are the results to equation 1, for every possible combination of operations and, of course, the values in a row add up to 1.

With this information, it is possible to understand what are the odds of every possible movement, given a previous one. Apart from shinning some light upon the behaviour of the users, these matrices will be the main stone upon which one of the solutions will be developed, more explanation will be done in the Methodology section.

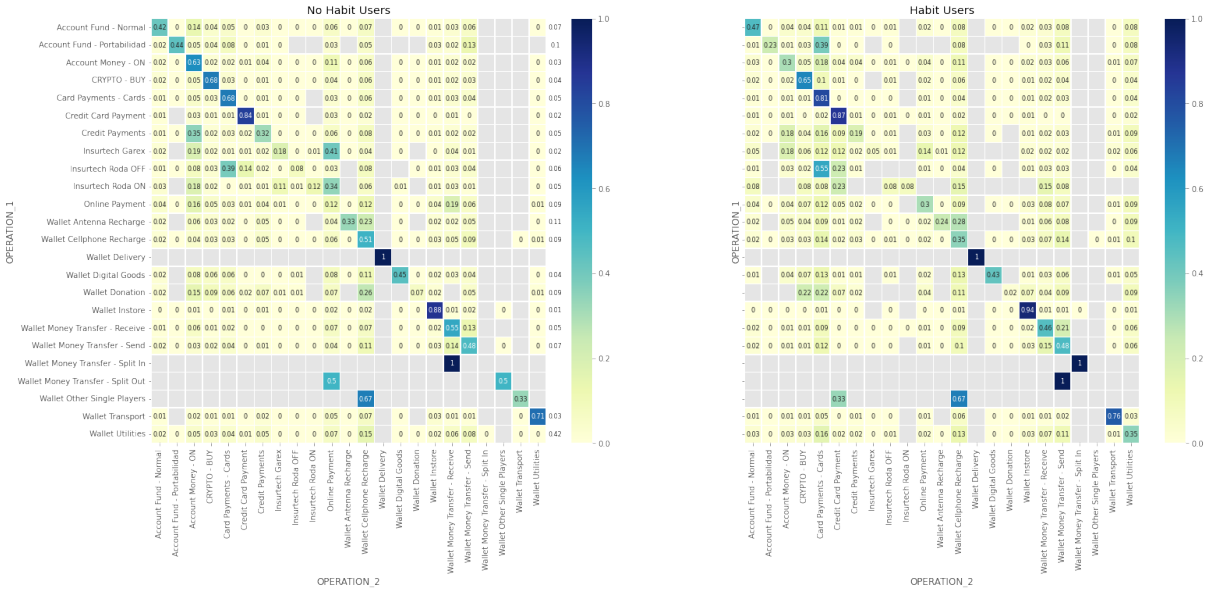


Figure 22. Transition matrix - 1º to 2º operation

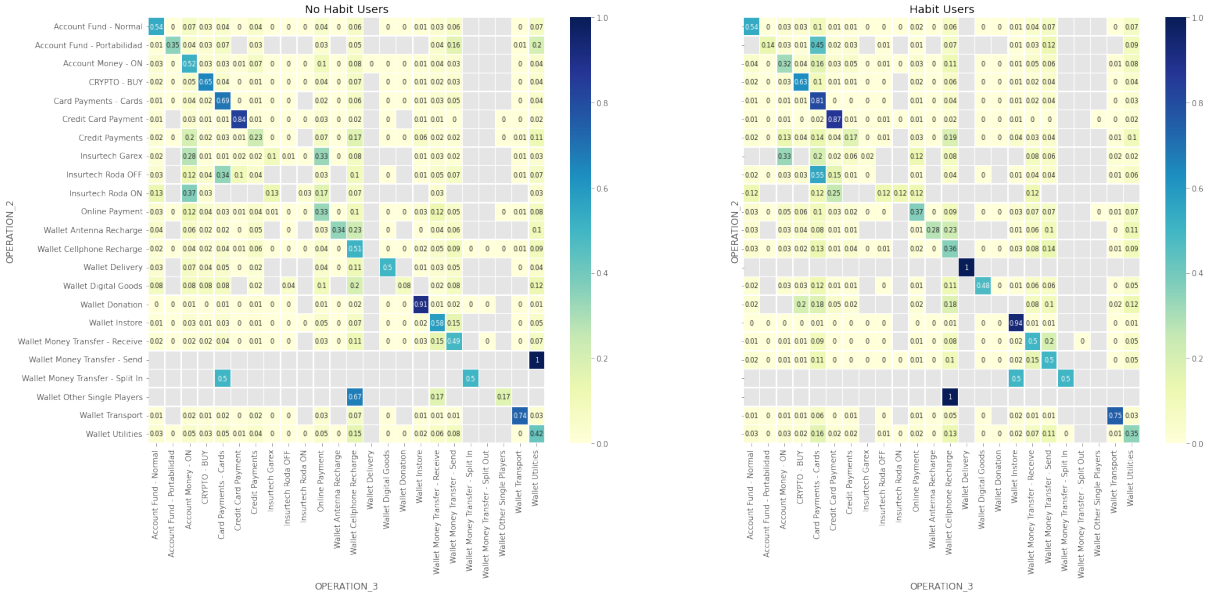


Figure 23. Transition matrix - 2º to 3º operation

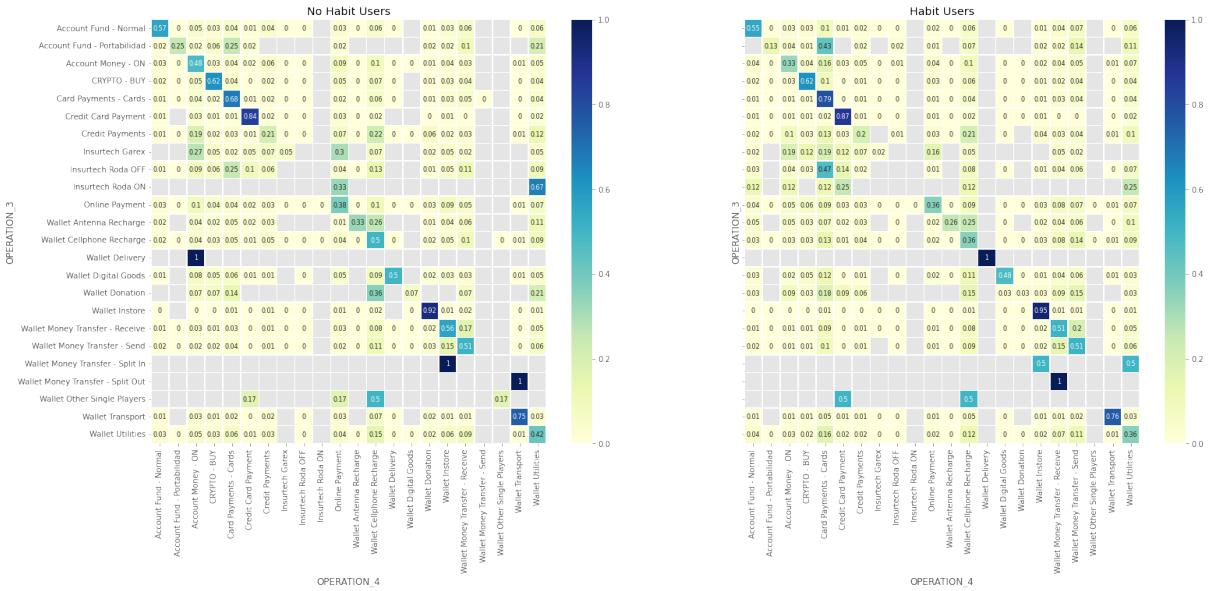


Figure 24. Transition matrix - 3º to 4º operation

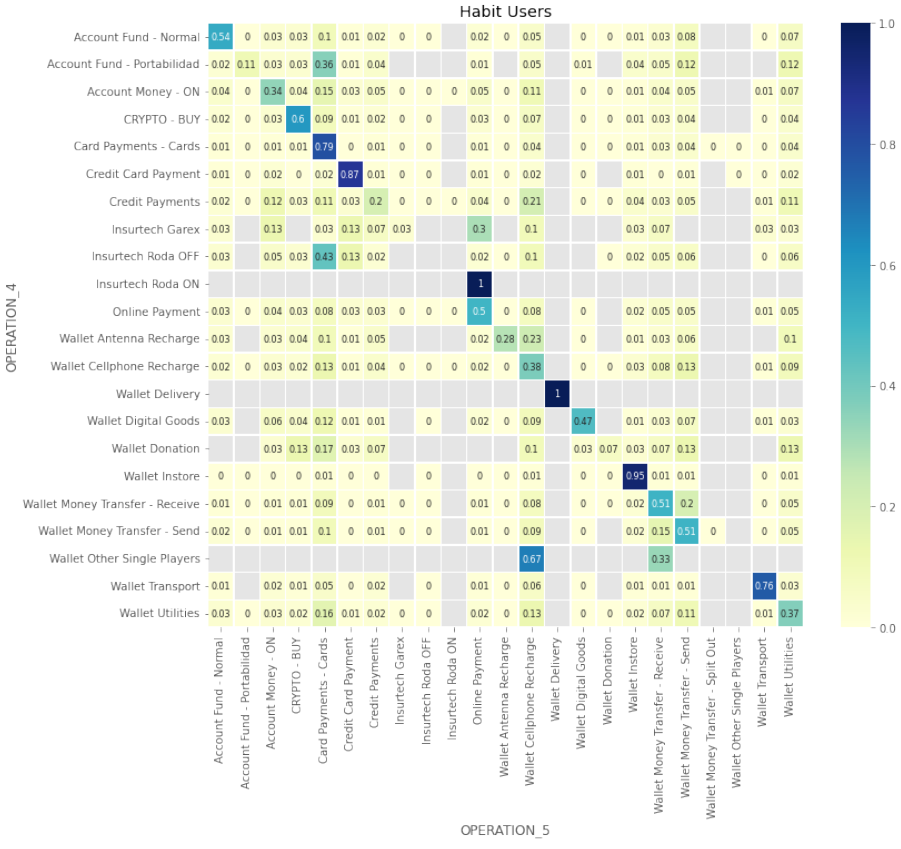


Figure 25. Transition matrix - 4º to 5º operation

The **transition matrices** show what we already suspected from the trajectories: after a certain operation, the probability of a user doing the same movement is higher that for the rest of the available options. That behaviour can be seen in the diagonals of the matrices, where the probabilities are higher. Of course, in some cases we find that the

odds are distributed and there is no absolute path, but in general, staying on the same track is the key to obtain a new operation.

An important conclusion we deduce from the Figures is that not all combinations between operations have enough data in order to obtain a probability distribution; such is the case with the grey boxes in the matrices. What is happening is that there are very few occasions where the user is passing from Operation A to Operation B (here, A and B represent any of the available movements). Because of this, the probability of the user doing that succession of steps ends up being nearly equal to zero. Also, it is because of that shortage that some cases have a 100% of chances to occur, which is impossible, as absolute certainty cannot be achieved. That is why, as will be explained later on, some of these combination of operations will be removed, as there are not enough examples in real life to safely estimate their probability of occurrence.

Each figure presents the matrices for both the **Habit** and the **Non Habit** users. As seen before, there is evidence that suggests that the two types of users might behave differently, this is the reason for such comparison.

Here we see the same behaviour observed in the engaged group, where the chances of the user doing the same operation twice in a row are more likely to occur than the rest. It is not very clear whether one group is over the other. In both cases, some combinations increase their probabilities from one group to the other, and sometimes it is the other way around. From these, we cannot draw any conclusions.

2.2.1.7 Number of operations per day

So far we have only explored the data in terms of individual *operation days*. Naturally, this is because what matters are the different days where the user operates (the reader should remember that the requisite for someone to be considered a **habit user** is to use the app at least in five different days, in the first thirty days since he first used it). However, this does not mean the users are not operating more than once in those days.

We are interested in understanding the number of operations that the users do in each operation day. As the analysis will be done comparing two different user profiles, it will allow us to comprehend whether there is a differential behaviour between the two, some insight that might separate them.

We counted the number of movements that a user did in each operation day, separating the persons that did not get to the habit state from those who did. With the data aggregated for each user and step, the next part consisted of calculating the confidence interval for the average number of payments, for the five operation days.

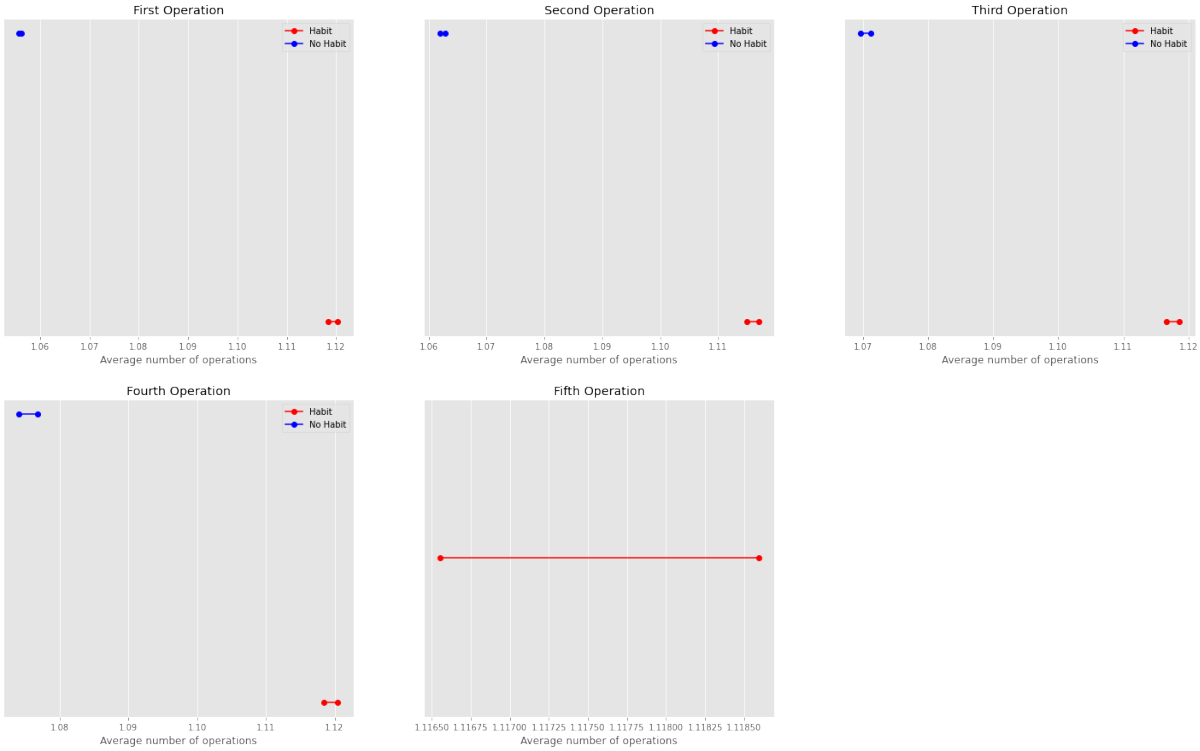


Figure 26. Average number of operations per day

Figure 26 presents all the confidence intervals for the average number of operations that a user does in every operation day until they reach (or don't) the final one. Blue lines represent the intervals for those who did not get to the habit state, and red ones the ones who did. At first glance, it is pretty clear that both groups behave very differently, given the fact that their intervals do not overlap. In itself, that is a good thing. The fact that both groups are separate means that there is a chance for future solutions to find a certain feature that could help them to identify each one better.

However, this plots only tell the statistical significance of the estimate (in this case, the average number of operations), meaning that both groups have *significantly* different means, but, when looking more carefully, we notice that they are very close to each other, with the blue group having a mean of around 1 operation per day, and red one with 1.12 operations per day. From a business point of view, there is no difference, all users do one operation, in average, every day.

3 Methodology

However interesting any theory, research and current work might be, no project would mean anything if they do not work as a stepping stone for the development of a new solution that may or may not outperform the *status quo*. In the corporate world, research can only take you so far, that is why everyone love numbers and results, they are certain, they are *facts*. If you have a theory, you test it, evaluate results and raise conclusions from them. This project is not the exception. The reason for this project to be was to improve the number of engaged users. For that particular problem, we have come with two approaches that we believe can achieve that goal. In the following sections, those solutions will be explained in full detail, along with their implementation in a real world

situation.

3.1 Technical Toolbox

Before introducing the solutions developed in this work, let us detail a bit about the tools we used throughout the project, as they are what allow us to develop all the models explained later on.

As every data science project, this one was composed of an **Analytic section** (previously introduced in the Data analysis section), a **Modelling section**, a **Prescriptive section** and finally a **Follow up section**. Of course, these are not sequential an many times we had to go back and forth in order to arrive at a robust solution. But the main idea of the project is summarised in those sections.

For the development of this project we used a set of tools and platforms that allowed us to work in each one of the previous sections.

- **BigQuery, Google Data Studio & Cloud Storage**

  The most important element of a data science project is the data, without it, there is not much anyone can do. All the data we needed (described in the Data section) was stored in **BigQuery** databases.

  It is a serverless, highly scalable data warehouse that comes with a built-in query engine. The query engine is capable of running SQL queries on terabytes of data in a matter of seconds, and petabytes in only minutes. This scalability is possible because the service distributes the query processing among thousands of workers almost instantaneously. What is interesting is that this performance is obtainable without having to manage any infrastructure and without having to create or rebuild indexes as the SQL queries can filter on any column in the dataset, and BigQuery will take care of the necessary query planning and optimization. Its simplicity of management is a great advantage when focusing on the machine learning project as it free us from thinking about the infrastructure, allowing us to focus on the solutions[21].

  Also, we deal with big amounts of data, so having a tool that can parallelize the processing is a big help because it reduces the time we spend exploring and building data. In BigQuery queries are automatically scaled to thousands of machines and executed in parallel. There is no need to do anything special to enable this massive parallelization. The machines themselves are transparently provisioned to handle the different stages of the job[21].

  Because BigQuery is an SQL engine, it is possible to use Business Intelligence (BI) tools to create meaningful analyses or reports in an easy way from data stored in BigQuery [21]. In our case, we used **Google Data Studio** to create dashboards that allowed us to keep track of our most important KPIs, experiment results and insights, all while automating such reports so that the information would always be up to date.

  Finally, another aspect that was very useful was the capacity to store *objects* in Google Cloud that were relevant to our process. For this, we used the **Google Cloud Storage (GCS)** service. An object is an immutable piece of data consisting of a file of any format and those are stored in containers called buckets. In other words, the **GCS** service works as a big repository where you can store your data,

whether it is a CSV file, JSON, an image or even a full machine learning model. This service was particularly useful as it helped us organize our datasets, data needed for the model and even predictions from those models all in one place.

- **Fury Data Apps**

  After the data was gathered using **BigQuery**, we needed an environment where we could handle, transform and use it to develop our solutions. It goes without questioning that our main tool for work was going to be **Python** as it has a lot of advantages (e.g., flexibility, a big community behind it, many machine learning implementations already developed, etc). However, we needed a workspace or environment where we could not only experiment "locally", but where we could also put our final experiments into productions in an easy way. This is where the **Fury Data Apps** comes into place.

  It is a framework, created in-house, to support data mining, development and deployment of Machine Learning models. In this acronym, the "F" stands for Fury, a PaaS (Platform-as-a-Service) tool for building, deploying, monitoring, and managing services in a cloud-agnostic way, also brewed in-house. Needless to say, it is where Fury Data Apps a.k.a. FDA, is embedded in. Thus, FDA enables creating end-to-end ML apps in a scalable, secure and efficient way, delivering high value to our users [37]

  This framework delivers a pipeline for the most important parts of a Data Science project as ours: **Experimentation, ETL and Training**, as well as a tool for **Automating** those processes.

  For experimentation, FDA provides hosted Laboratories or Labs, a platform for data analysis and sampling with support for Python, with simple access to all available data sources and other analysis tools through an open source library repository [37]. With Labs, it is also possible to push code into production in a very easy way. Ultimately, a Lab offers a Jupyter Notebook ready to start coding and access data using any common data science toolkits. Another important benefit from this tool was that it automatically created a GitHub repository, allowing us to trace our code, and even handle different scripts simultaneously.

  After a script is done, FDA allow us to create an **image** or specific version of it (ETL), one that we can later execute repeatedly, trusting that it would always work the way it was designed to do. The scripts output could be a dataset, a trained model or a prediction, whatever the case, the pipeline is designed to be robust and traceable. This is what gave us **reproducibility**, allowing us to trace back a result set to its original execution and find its source code.

  Finally, the Training step focuses on associating an ETL version to a model. And this is really interesting because we could try out different models on the same dataset. So using this step allowed us to create many training pipeline runs, associated to one same ETL version in the search for the best fit. The outcome of this process is a trained model which can be saved and stored.

  As mentioned before, all of this parts can be automated using a specific version so we can run the scripts that we are interested in.

- **Data Suite**

As mentioned before, developing a Machine Learning project comprises many parts. It is not only about creating the model and obtaining predictions, but there is also extracting the data, visualizations, reporting, automation, etc.

For this reason, **Data Suite** was developed. An in-house product that could be described as a **data platform**. It has tools for solving many of the needs that arise from the previous steps in a machine learning project (or any other data related project). The greatest advantage that it has is that not only it works as the only entry point for all the data oriented tools used in the company, but it also works as an ecosystem, allowing those tools to interact in a fluid way. Some of the use cases it tackles are:

- Find dashboards or metrics
- Run a query to a database
- Visualize data
- Build a data pipeline

As complete as it is, we used this tool because of the ease with which it allowed us to create data processes through which we extracted the necessary information for our analyzes and solutions through a series of steps that were run sequentially. Each of these steps could be a different kind of task: a query, an export to a bucket in GCS, an import from a bucket, etc. On the other hand, it also allowed us to schedule such processes, all while monitoring its executions.

In short, **Data Suite** worked as our coordinator, where we created our data pipelines which generated the data necessary for our solutions to work, keeping in charge of running the processes in order, handling errors and keeping track of everything so we could focus on our solutions.

- **Hermes**

  A very important aspect of our experiments is the communication. It was mentioned that the project was part of a marketing team, therefore, the end goal will always be to **reach** a user. Our main channels of communication are **push notifications (in-app messages)** and **email**. There are one or two other more, but our main levers are those two.

  For this purposes we use **Hermes**, another tool developed in-house for sending emails and push notifications. It allow us to create campaigns just by specifying a few parameters. We have control over the country and business unit that our campaign is going to target, it can also separate between buyers and sellers. On the other hand, Hermes also gives us the chance to segment the audience according to the experiment. For example, we might be interested in only targeting users with a certain loyalty level, or a certain number of transactions, the tool let us segment however we want.

  Of course, the content of the message is written directly in Hermes, and here the tool is quite flexible, allowing for text, images and links.

  All in all, Hermes is our go to tool for sending our push campaigns in an organize manner. It gives us the flexibility to configure the messages as we want, and, it has two other capabilities that makes this tool perfect for our needs: it automatically

creates a **control group** to which it will not send the campaign and it has an **internal capping system** that will prevent us from spamming a user with too many notifications. The first one allow us to have a base against which we can compare when evaluating the results of the campaign, and the latest ensures that we are not polluting the channel.

3.2 Proposed Solutions

3.2.1 Transition Matrix Solution

In the Data section, we have seen that for every "step" that the user takes in the path of becoming engaged, the previous operation that he did influence greatly in the next one. In other words, depending on the movement done before, the probability of the next one will vary.

In a way, this behaviour looks like a sequential problem, where each step taken in the past can impact on the next one. Based on that premise, we decided to develop a **Markov Chain-based** (MC) model.

However, an MC comes with an assumption that is not true in many cases, which is the idea that **the only thing affecting the future states of a system is the present one**, neglecting the influence of all previous states (more of this assumption will be explained in the following section).

That is why it is important to demonstrate how our problem can be approached with this strategy. Essentially, the idea is that we can prove that whether we consider the previous states or we only take into account the present one, it will not affect the outcome.

We can take the same data we used to build the transition matrices illustrated in section 2.2 and figure 21 and, based on it, we can build new matrices that consider all the previous states. An example is presented below for the transition to the third state, considering the previous steps 1 and 2.

OPERATION_1	OPERATION_2\OPERATION_3	Account Fund - Normal	Account Fund - Portabilidad	...	Wallet Money Transfer - Send	Wallet Transport	Wallet Utilities
Account Fund - Normal	Account Fund - Normal	0.707081	0.000191	...	0.042919	0.001083	0.050242
Account Fund - Normal	Account Fund - Portabilidad	0.142857	0.428571	...	0.142857	0	0
Account Fund - Normal	Account Money - ON	0.187215	0	...	0.026941	0.003653	0.044749
Account Fund - Normal	CRYPTO - BUY	0.169759	0	...	0.031615	0.000687	0.039863
Account Fund - Normal	Card Payments - Cards	0.125526	0.000701	...	0.032959	0.001753	0.041024

Table 17. Conditional Transition Matrix sample

Figure 17 shows a sample of the matrix previously described. The data illustrated in it represents the probability of moving to a certain step 3 (all the different operations available in the columns) conditional to the operations 1 and 2 that were done in the past. What this table represents is the idea of considering all previous states when estimating the probabilities of moving to the following.

If we build this matrices for all the remaining steps and take the **individual** ones where the previous states are not considered, we can compare them to prove that the final output will effectively be the same.

For this we ran a simulation using both matrices where we input new users with randomly assigned **past** operations and observed the output they both provided. In other words, we simulated the rows and, based on the probability distribution of the matrices, we obtained the next state. Later we compared the distribution of the "new" assigned operations for both matrices. The results were conclusive as they shown that the

distribution for both methods was almost identical, proving that we could safely apply the **Markov property**. Below we introduce the distributions for all steps.

Next State	Q Conditional	Q Individual	% Conditional	% Individual
Account Money - ON	29623	37894	9.0	8.0
Wallet Instore	12890	32224	4.0	6.0
Wallet Cellphone Recharge	34104	60422	10.0	12.0
Card Payments - Cards	46270	63255	14.0	13.0
Wallet Utilities	27482	39555	8.0	8.0
Credit Card Payment	26898	25014	8.0	5.0
Wallet Transport	15992	17193	5.0	3.0
Wallet Money Transfer - Send	24563	33555	8.0	7.0
CRYPTO - BUY	17781	24299	5.0	5.0
Account Fund - Portabilidad	8845	4795	3.0	1.0
Online Payment	17542	27883	5.0	6.0
Account Fund - Normal	13304	21652	4.0	4.0
Credit Payments	13761	12988	4.0	3.0
Wallet Money Transfer - Split In	1275	17379	0.0	3.0
Wallet Other Single Players	862	2978	0.0	1.0
Wallet Money Transfer - Receive	16077	31997	5.0	6.0
Wallet Antenna Recharge	5779	6977	2.0	1.0
Insurtech Roda OFF	3397	2158	1.0	0.0
Insurtech Roda ON	643	1166	0.0	0.0
Wallet Digital Goods	5700	10542	2.0	2.0
Wallet Delivery	858	20854	0.0	4.0
Insurtech Garex	2285	4239	1.0	1.0
Wallet Donation	713	980	0.0	0.0

Table 18. Step 3 Distribution

Next State	Q Conditional	Q Individual	% Conditional	% Individual
Wallet Utilities	12247	46951	9.0	9.0
Card Payments - Cards	27340	64752	19.0	13.0
Credit Payments	6148	14003	4.0	3.0
Account Fund - Normal	5721	21608	4.0	4.0
Wallet Transport	6508	17819	5.0	4.0
CRYPTO - BUY	9433	24570	7.0	5.0
Wallet Money Transfer - Send	11681	37128	8.0	7.0
Online Payment	6845	25504	5.0	5.0
Wallet Cellphone Recharge	14422	59555	10.0	12.0
Credit Card Payment	13442	37406	9.0	7.0
Account Money - ON	9800	41876	7.0	8.0
Wallet Instore	6195	34720	4.0	7.0
Wallet Digital Goods	2446	11381	2.0	2.0
Insurtech Roda OFF	741	1269	1.0	0.0
Insurtech Garex	115	858	0.0	0.0
Wallet Money Transfer - Receive	7423	35835	5.0	7.0
Wallet Antenna Recharge	1696	6407	1.0	1.0
Account Fund - Portabilidad	1373	3639	1.0	1.0
Wallet Donation	178	978	0.0	0.0
Wallet Delivery	45	10974	0.0	2.0
Wallet Other Single Players	57	2762	0.0	1.0

Table 19. Step 4 Distribution

Next State	Q Conditional	Q Individual	% Conditional	% Individual
Credit Card Payment	1825	29450	7.0	6.0
Wallet Digital Goods	210	11153	1.0	2.0
Account Fund - Normal	1232	19677	5.0	4.0
CRYPTO - BUY	1564	23867	6.0	5.0
Wallet Instore	1249	28679	5.0	6.0
Wallet Cellphone Recharge	3018	57643	11.0	13.0
Wallet Transport	815	18851	3.0	4.0
Wallet Utilities	2395	31760	9.0	7.0
Wallet Money Transfer - Send	2697	37462	10.0	8.0
Card Payments - Cards	6192	67596	23.0	15.0
Account Money - ON	1567	22063	6.0	5.0
Wallet Money Transfer - Receive	1657	36074	6.0	8.0
Online Payment	1196	45509	4.0	10.0
Credit Payments	1262	14513	5.0	3.0
Account Fund - Portabilidad	71	2633	0.0	1.0
Insurtech Roda OFF	131	778	0.0	0.0
Wallet Antenna Recharge	173	6649	1.0	1.0
Insurtech Garex	11	763	0.0	0.0
Wallet Donation	8	1485	0.0	0.0
Insurtech Roda ON	2	21	0.0	0.0

Table 20. Step 5 Distribution

In the previous figures we present the distribution obtained for the **Next Step** in the habit journey. When looking at the percentages (the last two columns) we notice that both methods present similar distributions in terms of the operations that will be done in that step. This similarity is consistent throughout all the steps in the habit journey, thus concluding that the Markov property is applicable in this case.

3.2.1.1 Markov Chains

Markov Chains belong to a sub-universe of models called **Probabilistic Graphical Models(PGM)** that represents a dynamic process. That is, a process which is not static but rather changes with time. In particular, it concerns more about how the state of a process changes with time. In simple terms, Probabilistic graphical modeling is a branch of machine learning that studies how to use probability distributions to describe the world and to make useful predictions about it.[10]

Before going any further into the theory of Markov Models, it is important to remind some basics notions of probability theory.

Random Variable and Random Processes

A **random variable** X is a variable whose value is defined as the outcome of a random phenomenon. This outcome can be a number or not. For example we can define a random variable as the outcome of rolling a dice (number) as well as the output of flipping a coin (not a number, unless you assign, for example, 0 to head and 1 to tail).

We can then define a **random process** (also called stochastic process) as a collection of random variables indexed by a set T that often represent different instants of time. Based on this, there usually are two types of processes[33]:

- A random process $X(t)$ where t can take real values in an interval on the real line, then $X(t)$ is a **continuous-time random process** (e.g. the thermal noise voltage

generated across a resistor in an electric circuit or the temperature in New York City

- **Discrete-time random process** is a process $\{X(t), t \in J)\}$, where J is a countable set. Since J is countable, it can be written as $J = \{t_1, t_2, ...t_n\}$. Therefore, a discrete-time random process is just a sequence of random variables (e.g. flipping a coin every day)[27]

The random variables at different instant of time can be independent to each other or dependent in some way as well as they can have continuous or discrete **state space** (space of possible outcomes at each instant of time)[27]

If we take a discrete-time random process $\{X_m, m = 0, 1, 2...\}$, if the X_m's are independent, the analysis of this process is relatively straightforward. In this case, there is no "memory" in the system, so each X_m can be looked at independently from previous ones.

However, for many real-life processes, the independence assumption is not valid. For example, if X_m is the stock price of a company at time $m \in \{0, 1, 2...\}$, then it is reasonable to assume that the X_m's are dependent. Therefore, it is necessary to develop models where the value of X_m depends on the previous values. Here is where **Markov chain** represent a great opportunity, as X_{m+1} depends on X_m, but given X_m, it does not depend on the other previous values $X_0, X_1, ..., X_{m-1}$. That is, conditioned on X_m, the random variable X_{m+1} is independent of the random variables $X_0, X_1, ..., X_{m-1}$.

As mentioned at the beginning of the section, **Markov chains** are used to model the successions of **states** $S = \{s_0, s_1, ...\}$. Put it simply, a state at time $t + 1$ is dependent only on the current state t and is independent of all previous states from $t - 1, t - 2, ...$ In short, to know a future state, we just need to know the current state.

Definitions: Discrete Markov Models

Considering a random process $X_n, n = 0, 1, 2...$, we say that this process is a Markov chain if:

$$P(X_{m+1} = j | X_m = i, X_{m-1} = i_{m-1}, ..., X_0 = i_0) = P(X_{m+1} = j | X_m = i) \quad (2)$$

for all $m, j, i, i_0, i_1, i_{m1}$. If the number of states is finite, e.g., $S = \{0, 1, 2...r\}$, then it is called a **finite Markov chain**.

Now that it has been introduced, it gets clear why it seemed like a great solution for the problem at hand. The process of getting to do a certain amount of operations in succession is, in itself, a sequential process. If we consider each operation as a new "state", we could model the problem as a Markov Process. Still, this has only helped to introduce the basic theory behind the solution we proposed but, **what elements compose a Markov chain that can help to figure out the next more likely state?**

3.2.1.2 Transition probabilities and Transition Matrix

Remembering the formal definition of a Markov chain presented in equation 2, we establish that when $X_n = j$, then, the process is at state j. The final part of the equation 2 is called the **transition probabilities**. If we assume that these probabilities do not depend on time, then we can define:

$$p_{ij} = P(X_{m+1} = j | X_m = i) \quad (3)$$

In other words, equation 3 says that if the process is in state i, it will next make a transition to state j with probability p_{ij}. This is a useful definition for our habit user problem. Keeping in mind that our goal is to achieve 5 different operation days, **we believe that the best approach is to suggest only meaningful services from the application**. Equation 3 establish that, by considering the sequence of operations as a Markov Process, then each new operation can be seen as a new state and those transitions have a certain probability of happening determined by equation 3.

For simplicity, we can list all the transition probabilities into a matrix called the **Transition matrix** or **Transition probability matrix**, commonly shown by P.

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1r} \\ p_{21} & p_{22} & \cdots & p_{2r} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ p_{r1} & p_{r2} & \cdots & p_{rr} \end{pmatrix} \text{[27]} \tag{4}$$

Because we are handling probabilities, all the elements of the matrix are bigger or equal to zero (and lesser or equal to one), that is, $1 \geq p_{ij} \geq 0$. On the other hand, for every i (for each row) it must be true that:

$$\sum_{k=1}^{r} p_{ik} = \sum_{k=1}^{r} P(X_{m+1} = k | X_m = i)$$
$$= 1$$

The above expression is imposing a very important condition. Mathematically, it is saying that the rows of any state transition matrix must sum to one. However, that must happen because given that we are in state i, the next state must be one of the **possible states**, in other words, all states must be contemplated in the matrix.

It is very common to represent a Markov Chain as a **state transition diagram**, with each node being a possible state of the system and the connections between them the probabilities of transition p_{ij}.

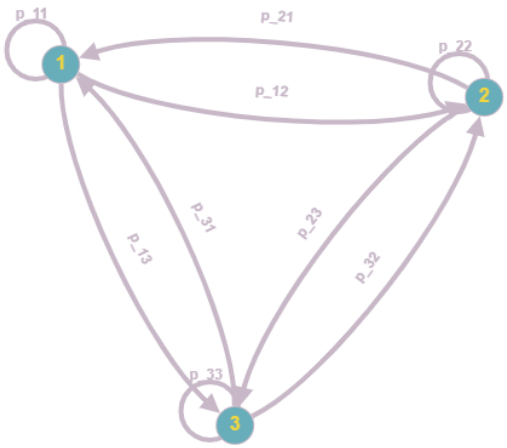


Figure 27. State transition diagram representation of a Markov chain

3.2.1.3 *n*-Step Transition Probabilities

Having defined the transition matrix, the problem of suggesting the next best option for a user is merely a probabilistic one. If we know the probabilities of transitioning for every state, then we would have every scenario possible for us to choose one, base on the likelihood of it.

But the problem addressed by this project consists of 5 steps, and the transition matrix represents only one transition, **how can we determine the probability of being at state r after n steps?**

Here is where the reality deviates from the theory, as we implemented this solution differently than what normal Markov chain problems do. Before explaining our solution (which does not necessarily varies to much from the theory, but changes some considerations) it is important to demonstrate how can we *formally* obtain the probability of transition from state i to state j after n steps.

Starting from a simple case, we are interested in finding the probability of going from state i to state j in two steps, i.e.,

$$p_{ij}^{(2)} = P(X_2 = j | X_0 = i)$$

We can find this probability by applying the law of total probability[27] . Given that we know the initial and final state, the problem is the fact that $X1$ can take one of the possible values in the **state space**. Based on that, the problem can be expressed as:

$$p_{ij}^{(2)} = P(X_2 = j | X_0 = i) = \sum_{k \in S} P(X_2 = j | X_1 = k, X_0 = i) P(X_1 = k | X_0 = i)$$

$$= \sum_{k \in S} P(X_2 = j | X_1 = k) P(X_1 = k | X_0 = i) (\text{by Markov property})$$

$$= \sum_{k \in S} p_{kj} p_{ik}$$

In conclusion:

$$p_{ij}^{(2)} = P(X_2 = j | X_0 = i) = \sum_{k \in S} p_{kj} p_{ik} \tag{5}$$

In order to get to state j, we need to pass through some intermediate state k. The probability of this event is $p_{ik}p_{kj}$. To obtain $p_{ij}^{(2)}$, we sum over all possible intermediate states. This can be illustrated by defining the two-step transition matrix $P^{(2)}$, which is equivalent to squaring the state transition matrix[27], i.e.,

$$P^{(2)} = P^2 \tag{6}$$

$$P^2 = \begin{pmatrix} p_{11} & p_{12} & \dots & p_{1r} \\ p_{21} & p_{22} & \dots & p_{2r} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ p_{r1} & p_{r2} & \dots & p_{rr} \end{pmatrix} \begin{pmatrix} p_{11} & p_{12} & \dots & p_{1r} \\ p_{21} & p_{22} & \dots & p_{2r} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ p_{r1} & p_{r2} & \dots & p_{rr} \end{pmatrix}$$

We could generalize the cases presented in equations 5 and 8 to present the definition for n-step transitions probabilities. Let m and n be two positive integers and assume $X_0 = i$. In order to get to state j in $(m+n)$ steps, the chain will be at some intermediate state k after m steps. To obtain $p_{ij}^{(m+n)}$, we sum over all possible intermediate states[27]:

$$p_{ij}^{(m+n)} = P(X_{m+n} = j | X_0 = i) = \sum_{k \in S} p_{ik}^m p_{kj}^n \tag{7}$$

And the n-step transition matrix is given by:

$$P^{(n)} = P^n, \text{for n = 1,2,3,...} \tag{8}$$

3.2.2 Transition Matrix Solution - Implementation

Now that a solid base on Markov chains has been given, it is possible to continue with the solution we proposed and implemented in a real life situation. As mentioned before, the theory behind the solution is based on the assumption that the process of operating in the app is a Markov process, that is, that the next operation a user does is does not depend on the operations that he did before. That in itself is a very strong assumption. We decided to go with it anyways as we were trying two different solutions, and the other one was more "complex" than the idea of a Markov chain, so we chose to be accept that assumption.

The solution we proposed can be framed as follows:

> *Develop a probabilistic set of rules to recreate the current behaviour of the majority of the users based on a matrix of probabilities (a **Transition Matrix**) in order to determine the chance of a user transitioning from one operation to the next. Based on that probability, an algorithm will suggest an operation through a **weighted random sorting**.*

In other words, we propose to find the transition matrix and use it to select the next operation to suggest using an algorithm that sorts randomly based on the probabilities.

So far, everything seems to be like what the theory said in previous sections, **how does this differ from the theory then?**

Again, the differences are not in the concepts but in **how** we use the matrix, and the structure of it. For starters, when introducing the n-step transition matrix it was established that in order to determine the probability of moving from state i to state j in n steps it was necessary to elevate the matrix to the n, thus obtaining the probabilities. However, we decided to not do that and keep the transition matrix as it was. Instead, for every new "step" given, for every new operation that the user did, there would be a **new** transition matrix. The reason we did this is because every time the user is presented with a choice, the uncertainty is his next move. His past decisions are not important for us, as they are certain. Therefore, we do not estimate the probability of transitioning from operation i to operation j in n steps, we only focus on one step at the time, having built a different matrix for every new step.

By implementing Markov chains in this way, we are assuming that each new operation depends only on the current "state" or the last operation the user did. All previous choices do not matter.

On the other hand, another significant difference from the theory is that our transition matrix would not be square (the number of rows and columns is equal). This was designed

on purpose for the following reasons:

- **Not all operations are communicable**

  Although section 2.1.1 introduced the available operations that business experts consider appropriate to include as valid for the habit journey, not all of them are communicable. That means, that some operations, although valid, are impossible to communicate to the user. In this case, the operations are the **Online Purchases** that occur outside the platform, more specifically, the ones that occur because of a third party involved (e.g., a user buys a product in an online retailer outside the ecosystem, and the payment method is through the fintech application). In this cases, the user cannot choose to pay that way but was "forced" to do so. Because of that, we cannot communicate that operation in an efficient way through our channels.

- **Some operations are not currently relevant for business**

  Another possible reason for us not to include all operations in the matrix is because, at the time, they might not be quite relevant for the business. Meaning that although it is possible to communicate it, the value that it brings to the company is close to none (because of the low number of users that do those operations or because communicating them would be counterproductive). This does not mean that the possibility of these transactions occurring is zero, but it is very low, reason why we excluded them, leaving the matrix with only the most relevant operations. There is another reason why we took this decision. As mentioned, each user in our universe has a very limited window of opportunity for us to target them with our models (30 days). That means that each day counts, all the more reason for suggesting **only** the services that have the biggest probability of success (user operating in that service). The reader should keep in mind that for us to consider that a certain campaign or treatment was successful, we must wait for an **attribution window** to pass (see subsection 2.1.5), if we send operations that we already now are very unlikely, we would be wasting a bullet, as we would have to wait two days before we could try another suggestion. Because of this, we decided to discard the irrelevant operations

Having said that, the features that were effectively tested were:

- Account Funding
- Antenna TV
- Credits
- Cripto
- Digital Goods
- Instore
- Recharge
- Send Money 1
- Send Money 2
- Send Money 3
- Transport
- Utilities

This means we ruled out the following operations in this first version:

- Card - Prepaid
- Credit Card
- Debit Card
- Delivery
- Donations
- Insurtech 1
- Insurtech 2
- Insurtech 3
- Marketplace

operations

- Online Payment 1
- Online Payment 2
- Online Payment 3
- Others
- Portability
- Recieve Money

As a side note, the movements associated with **Money Send** were grouped together as one, because they all represent the same operation, they only differ in the application.

Apart from **Card - Prepaid, Credit Card, Debit Card and Online Payments**, the rest of the operations were excluded because their contribution was marginal in comparison with the rest. Online payments, as mentioned before, is not an operation that we can communicate effectively because we have no control over which sites operate through the app.

The case of the cards is particular as they are in themselves very important movements. However, in order to suggest a user one of those movements we would have to segment the users based on their situation (whether they have a card or not, if they applied but do not have it, etc.). The fact remains that because of all the different scenarios we had to map in order to correctly suggest one of those operations and because those segmentations were determined by other teams that were not us and they were not very clear on them, we decided to exclude cards from the models.

3.2.2.1 Implementation - Practical Approach

Up next, we present the flowchart of the **Transition Matrix approach** as a visual demonstration of how the process was implemented, followed by an explanation.

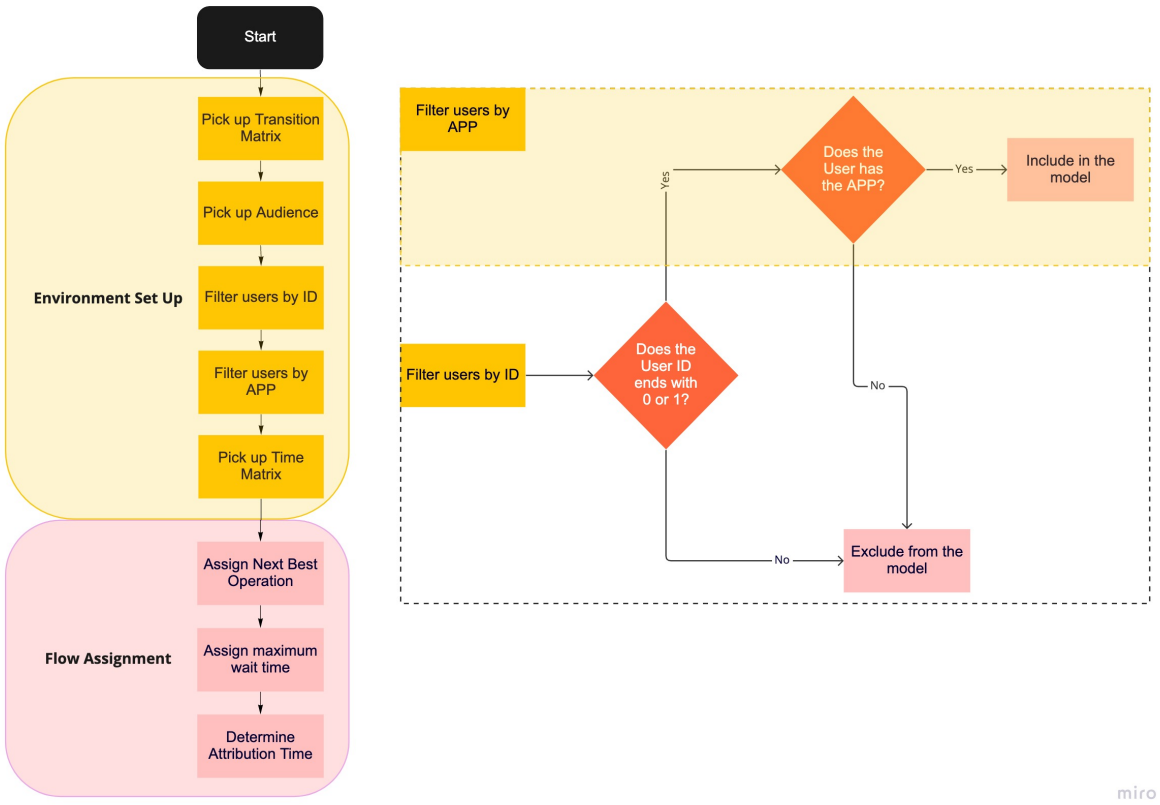


Figure 28. Transition Matrix Approach Flowchart

The flowchart represents the full process developed for the first of the two solutions. As was explained when presenting the tools in section 3.1, the pipeline was built using the FDA (Fury Data Apps) framework. We created a Lab for building all the different boxes presented in figure 28 in a Python script. Next, we explain in detail about the steps that compose our solution:

1. **Environment Set Up**

   The first part of the process comprises of a set of tasks where the necessary data is gathered and uploaded from different sources. In short, the following series of tasks will load and prepare the data for the model to work correctly.

   1.1 *Pick up Transition Matrix*

   Using the transition matrices obtained while analyzing data (see section 2.2) the script downloads them from a bucket in **Google Cloud Storage**, where they were stored by a parallel process running in **Data Flow** which is in charge of gathering the information about the users operations for the last 6 months for later computing the probability of transitioning from state i to state j, that is, of passing from a certain initial operation to another. As mentioned before, a different matrix is generated depending on the step given (e.g., the first operation to the second one, or the third to the fourth one)

   1.2 *Pick up Audience*

   Next, the script makes sure to download the **Audience**, that is, the **Prospect users table**, as they are who we can impact with our models, which is also stored in **GCS.**

   1.3 *Filter users by ID*

   This task and the following one are designed to filter users that do not gather the necessary requirements to be part of the experiment. What we do here is select a sub-universe of the grand total in order to exclude it and be able to experiment with it. This is because we do not want to impact the full base of the prospect users yet and the remaining ones will be used as benchmark for evaluation. More in detail, what we are asking here is whether the User Id (which is a unique number) **ends either with 0 or 1**, which represents a 20% of the full prospect users base that will be used for the experiment.

   1.4 *Filter users by APP*

   After selecting the users for the experiment, we want to do a second filtering where we can keep only those users that we are sure can receive our notifications. For this, we select **only those who have the fintech application**.

   It might seem weird to do this as it would not be possible to do any operation if the user did not have the application. However, some operations do not require the user to have it, like any **Online payment** movement. This is because the user ended up using the service as a result of having operated through a web page that uses the fintech application as a means of payment.

   1.5 *Pick up Time Matrix*

   Similarly to the **Transition Matrix**, there is a process in Data Flow that computes what we call a **Time Matrix** (an example can be seen in figures 14, 15, 16 and 17). The purpose of these is to set a time limit for the waiting

time before sending a notification to a user. Conceptually, these matrices contain information about the average time that pass between operation i and j, further explanation about the usage of the Time matrix will be given in the Deployment section. During this task we are downloading such matrix.

Thus concludes the **Environment Set Up** section of the process. The output is a dataset of users who are treatable (we can send them notifications) with a structure equal to that of the **Prospect users table**. Up next come the core tasks of the solution, the flow assignment ones.

2. **Flow Assignment**

2.1 *Assign Next Best Operation*

Having downloaded the Transition matrix, an algorithm uses it to assign the "next best operation" to each user in the universe. For this, the program first looks at the "current state" of a user (which actually is the last operation that he did), and at what stage of the habit journey he is (is it his second operation, third? etc.). With that, the algorithm selects the correspondent matrix, filter it with the current state, keeping a list of probabilities associated with an operation from where it has to choose which one to suggest. For the selection, the algorithm uses a weighted sorting based on the probabilities from the matrix.

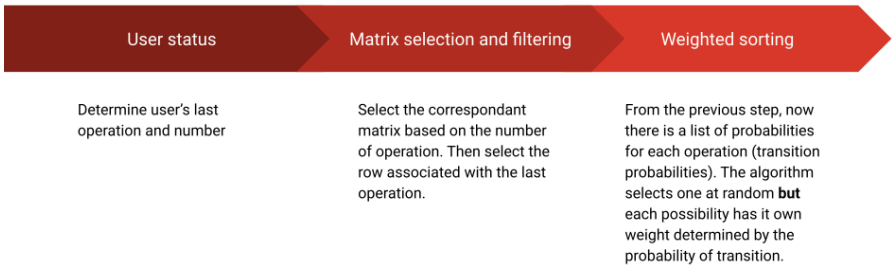


Figure 29. Flow selection process

2.2 *Assign maximum waiting time*

After the algorithm has selected the operation to suggest to each user, the next part is to determine how much time can we wait before sending it. This is related to the prescriptive part of the project where we explain what we do with the suggestions. As we mentioned already, our way to impact the users is through push notifications (messages in-app). However, the moment of notification is also a complex problem in itself, one that we will not cover in the scope of this work. However, we can say that the notification event is triggered by certain actions provided by the user (more into it in section 3.3). How does this relate to this part of the process? The maximum waiting time

is the number of days that we can wait for the user without him triggering the notification event. For example, if the waiting time is of 4 days, that means that if the user does not trigger a notification event in that time, then we will send a notification.

Such matrix is obtained by a process hosted in **Data Flow** which looks at the last 6 months of operations and estimate the average time between two operations (no matter the step in the habit journey), a similar matrix can be seen in figures 14, 15, 16 and 17.

The process of assigning the time is quite similar to the previous step. An algorithm looks at the "new operation" selected and the current state (latest movement done). With those two coordinate points, the program selects the time that is saved at that point in the matrix, assigning it to the user.

2.3 *Determine attribution time*

The final step of the process might seem off when compared to the rest of the actions. In practice, what we do here is assign the **attribution window** that corresponds to the "new operation" chosen before. It has nothing to do with the process of assigning an operation, but, as it will be explained later, it is important for choosing the right time when to send the notification.

Here, we determine the attribution window by selecting it from a python dictionary, previously loaded, where the key is the new operation chosen before.

Thus concludes the process that applies the first of our solutions. When it is finished, the result is a list of users with a new operation to suggest, a maximum time to wait and an attribution time features. Notice that at this point we have not mentioned anything about the process of sending the notifications. That is because the pipeline is the same for all the experiments, that is why it will be explained in its own section.

3.2.3 Uplift Model Solution

So far we have explained our first solution to the problem of improving the user engagement. As mentioned before, the **Transition Matrix** approach was a more "ruled based" one, a simpler one of sorts. However, another option was to lean on a more "intelligent" solution.

Our reasoning was simple, we wanted to suggest options that had the biggest chance of success, that is, we were interested in recommending operations that would really interest the users. We knew that a **propensity** model would be enough as they only estimate the probability of a user doing an operation, but it does not tell you whether you should recommend it or not. This would not be useful to us as a user might had a big chance of operating in some category, but he might operate anyways, without our recommendation. That would mean that our model did not actually generate any kind of impact, it did not change the final result.

For that reason, we decided to make use of a category of supervised models called **Uplift modelling**.

3.2.3.1 Uplift Modelling - Causal Inference Models

Uplift modeling is a branch of machine learning which aims at predicting the causal effect of an action such as a marketing campaign or a medical treatment on a given individual

by taking into account responses in a treatment group, containing individuals subject to the action, and a control group serving as a background. The resulting model can then be used to select individuals for whom the action will be most profitable.

Machine learning is primarily concerned with the problem of classification, where the task is to predict, based on a number of attributes, the class to which an instance belongs, or the conditional probability of it belonging to each of the classes. Unfortunately, classification is not well suited to many problems in marketing or medicine to which it is applied. Let's take, for example, a direct marketing campaign where potential customers receive a mailing offer. A typical application of machine learning techniques in this context involves selecting a small sample of customers who receive the campaign. Next, a classifier is built based on the pilot campaign outcomes and used to select customers to whom the offer should be mailed. As a result, the target will be the customers most likely to buy after the campaign [35]

Unfortunately, this is not always what we want in a model. The reason is because it could happen that some users would have bought even without the need of an incentive campaign, for which targeting them resulted in unnecessary costs. Another possible situation would be that customers were going to make a purchase but were bothered by the campaign, resulting in a loss of a sale, and sometimes a loss of the customer (churn).

For a campaign to be successful, we need to be able to select those customers who will purchase *because* of the campaign, i.e., those who are likely to buy if targeted, but unlikely to buy otherwise.

To describe it better, this situation is normally presented by classifying customers into 4 groups:

1. People who will purchase no matter what or **sure things**
2. People who will purchase only if they are exposed to an advertisement or **persuadables**
3. People who will not purchase no matter what or **lost causes**
4. People who will not purchase if they are exposed to an advertisement or **sleeping dogs**

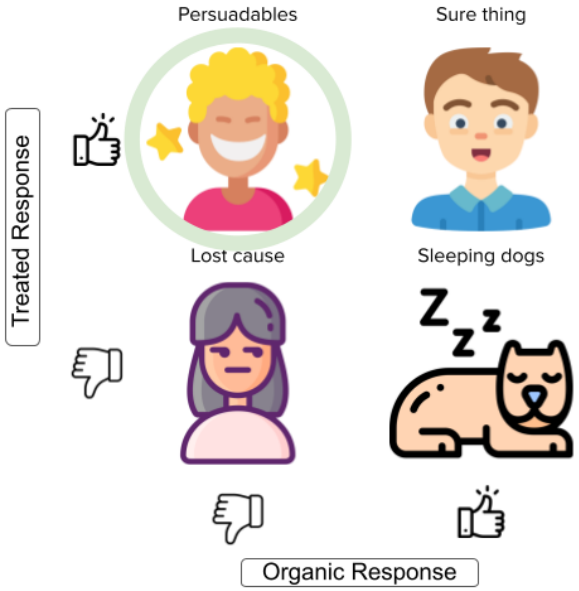


Figure 30. Uplift model users classification

Here is where Uplift modeling can provide a solution. The approach employs two separate training sets: **treatment** and **control**. The subjects in the **treatment group** have been subject to some action, in this case, a marketing campaign. On the other hand, the **control group** is comprised of subjects who did not receive any kind of treatment or action. Their main purpose is to serve as a benchmark against which the effects of the treatment can be assessed. Instead of modeling class probabilities, uplift modeling attempts to **model the difference between conditional class probabilities in the treatment and control groups**. This way, the causal influence of the action can be modeled, and the method is able to predict the **true gain** (with respect to taking no action) from targeting a given individual[35], estimating the customer's uplift, that is, the efect of an action on some customer outcome. To put it simply, the uplift models focus on correctly identifying the **persuadables** users and avoid the rest, either because they are a **lost cause** and no matter what we show them they will never change their mind, or because targeting them would mean a loss for the company (they would purchase anyway or, on the other hand, would not buy if we target them).

Uplift modeling is useful for answering questions like: **Did my advertising cause the customer to purchase from me?, Did I waste money advertising to customers who were already going to purchase from me? Did my advertising make the probability of someone purchasing worse (negative impact)?**

3.2.3.2 Types of algorithms

Having introduced the concept of Uplift models, it is time to mention the biggest challenge one encounters when designing uplift modeling algorithms:

> *For every individual, only one of the outcomes is observed, after the individual has been subject to an action (treated) or when the individual has not been subject to the action (was a control case), never both.*

Essentially this means that we do not know whether the action was beneficial for a given individual and, therefore, cannot assess model's decisions at the level of individuals. This is different from classification, where the true class of an individual is known, at least in the training set.[35]

Estimating customer uplift is both a Causal Inference and a Machine Learning problem. It is a causal inference problem because one needs to estimate the difference between two outcomes that are mutually exclusive for an individual. To overcome this counterfactual nature, uplift modeling relies on randomized experiments, i.e. the random assignment of customers to either receive the treatment (the treatment group) or not (the control group). Uplift modeling is also a machine learning problem as one needs to train different models and select the one that yields the most reliable uplift prediction according to some performance metrics. This requires sensible cross-validation strategies along with potential feature engineering.[13]

In the following sections we will explain the main approaches to combine this Causal Inference aspect with the Machine Learning one. As a side note, current literature proposes some other methods, however, we will explain those which we studied during the development of this work.

- the **Two Model approach**
- the **X-Learner**
- the **R-Learner**
- the **S-Learner**

These approaches are known as **meta-learners** or **meta-algorithms**. Formally, they are defined as the result of combining supervised learning or regression estimators (i.e., base learners) in a specific manner while allowing the base learners to take any form. Meta-algorithms thus have the flexibility to appropriately leverage different sources of prior information in separate sub-problems of the estimation problem: they can be chosen to fit a particular type of data, and they can directly take advantage of existing data analysis pipelines.[20]

3.2.3.3 Causal Inference: Basics

First of all, we will introduce some basic concepts and notation for better understanding the different approaches in the following sections.

We consider a framework with N individuals indexed by i. Denoting $Y_i(1)$ person i's outcome when he receives the active treatment and $Y_i(0)$ person i's outcome when he receives the control treatment, the *causal effect*, τ_i, of the active treatment *vis-'a-vis* the control treatment is given by:

$$\tau_i = Y_i(1) - Y_i(0) \tag{9}$$

Typically, we are interested in estimating the Conditional Average Treatment Effect (CATE), that is, the expected causal effect of the active treatment for a subgroup in the population:

$$CATE : \tau(X_i) = E\left[Y_i(1)|X_i\right] - E\left[Y_i(0)|X_i\right] \tag{10}$$

Where X_i is a $L \times 1$ vector of random variables (features). Of course, we will never observe both $Y_i(1)$ and $Y_i(0)$. Letting $W_i \in (0, 1)$ be a binary variable taking on value 1 if person i receives the active treatment, and 0 if person i receives the control treatment, person i's observed outcome is actually:

$$Y_i^{obs} = W_i Y_i(1) + (1 - W_i) Y_i(0) \tag{11}$$

Uplift modeling amounts to estimating a CATE. However, the fact that we never observe the true τ_i makes it seemingly impossible to use standard supervised learning algorithms to estimate it. If we suppose for a moment that τ_i was indeed observed, we would simply split the data into a train and a test set and use one of the many available algorithms to come up with the approximation of the CATE $\hat{\tau}(X_i)$. We would then evaluate our model using one or more metrics (AUC, F1 score, Accuracy etc) on the test data.[13]

The approaches mentioned in the previous section propose different ways to estimate such $\tau(X_i)$.

3.2.3.4 Two Model approach

Also known as **T-Learner**, this model is used as a baseline model in many uplift research papers. The approach consists in modeling $E[Y_i(1)|X_i]$ and $E[Y_i(0)|X_i]$ separately, using the treatment group data and the control group data, respectively. The learner estimates the CATE by:

$$\hat{\tau}(X_i) = \hat{\mu_1}(X_i) - \hat{\mu_0}(X_i) \tag{12}$$

Where $\hat{\mu_1}(X_i)$ and $\hat{\mu_0}(X_i)$ are model estimators for the conditional mean:

$$\hat{\mu_1}(X_i) = E(Y(1)|X_i) \tag{13}$$

$$\hat{\mu_0}(X_i) = E(Y(0)|X_i) \tag{14}$$

The advantage of the Two-Model approach resides in its simplicity. Because inference is done separately in the treated and control group, state-of-the-art machine learning algorithms such as Random Forest or XGBoost can be used. Both models can achieve good prediction performance, separately. However, for uplift purposes, although the approach has been seen to perform well, it is often outperformed by other methods. One reason is that the two models focus on predicting the outcome separately and can therefore miss the "weaker" uplift signal.[13]

3.2.3.5 X-Learner

This is an extension of T-learner, and consists of three stages. First it starts by estimating the response functions $\mu_0(X_i)$ and $\mu_1(X_i)$ using any suitable regression methods and the data from the control and treatment groups, respectively, as in equations 13 and 14.

It then proceeds to estimate "pseudo-effects" D_i for the observations in the control group as:

$$\tilde{D}_i^0 = \hat{\mu}_1(x) - Y_i \tag{15}$$

and for the individuals in the treatment groups as:

$$\tilde{D}_i^1 = Y_i - \hat{\mu}_0(x) \tag{16}$$

Where Y_i is the observed value for the user. The pseudo-effects are then used as the outcome in another pair of regression methods to obtain the response functions $\hat{\tau}_0(x)$ and $\hat{\tau}_1(x)$ for the control and treatment groups, respectively. That is, we estimate:

$$\hat{\tau}_0(x) = E\left[D^0 | X = x\right] \tag{17}$$

$$\hat{\tau}_1(x) = E\left[D^1 | X = x\right] \tag{18}$$

Using machine learning models and the pseudo-effects as target.

Finally, we define the CATE estimate by a weighted average of the two estimates:

$$\hat{\tau}(x) = \hat{e}(x)\hat{\tau}_0(x) + (1 - \hat{e}(x))\hat{\tau}_1(x) \tag{19}$$

Where $\hat{e}(x)$ is a propensity score $P\left[W_i = 1 | X_i = x\right]$, with W_i indicating the assigned treatment. [44]

3.2.3.6 R-Learner

This meta-learner is a bit more complicated that the rest, and because a formal demonstration of the formulas is not in the scope of this work, we will directly explain how it works without detailing how some specific expressions came to be.

First, to explain the R-Learner, we formalize the problem by establishing that we have n independent and identically distributed examples $(X_i, Y_i, W_i), i = 1, ..., n$ where X_i denotes per-person features, $Y_i \in \mathbb{R}$ is the observed outcome, and $W_i \in \{0, 1\}$ is the treatment assignment. So far, notation remains the same as in the other learners.

By now it should be evident that the main goal of these learners is to estimate the conditional average treatment effect or CATE function $\tau(x)$. Here is where the R-Learner diverges from the others, as it defines such function differently. Through **Robinson** decomposition (Robinson, 1988) we define the following expression for the CATE function, in terms of the conditional mean outcome $m(x) = E(Y|X = x) = \mu_{(0)}(X_i) + e(X_i)\tau(X_i)$, where $e(x)$ is the **treatment propensity** and $\mu_{(w)}, w \in (0, 1)$ are the conditional responses:

$$Y_i - m(X_i) = \{W_i - e(x_i)\}\tau(X_i) + \varepsilon_i \tag{20}$$

The goal of the R-Learner is to use equation 20 for flexible treatment effect estimation that builds on modern machine learning approaches such as boosting or deep learning. We can use this representation to construct a loss function that captures heterogeneous treatment effects, and then accurately estimate treatment effects—both in terms of empirical performance and asymptotic guarantees—by finding regularized minimizers of this loss function.[25]

Expression 20 can be written as (Robins, 2004):

$$\tau(.) = argmin_\tau \left\{E\left([\{Y_i - m(X_i)\} - \{W_i - e(X_i)\}\tau(X_i)]^2\right)\right\} \tag{21}$$

Given the previous expression, if we knew both the functions $m(x)$ and $e(x)$ a priori could estimate the heterogeneous treatment effect function $\tau(.)$ by empirical loss minimization:

$$\tilde{\tau}(.) = argmin_{\tau} \left(\frac{1}{n} \sum_{i=1}^{n} \left[\{Y_i - m(X_i)\} - \{W_i - e(X_i)\} \tau(X_i) \right]^2 + \Lambda_n \{\tau(.)\} \right) \tag{22}$$

Where the term $\Lambda_n(\tau(.))$ is interpreted as a regularizer on the complexity of the $\tau(.)$ function. This regularization could be explicit as in penalized regression, or implicit, e.g., as provided by a carefully designed deep neural network. The difficulty, however, is that in practice we never know the weighted main effect function $m(x)$ and usually don't know the treatment propensities $e(x)$ either, and so the estimator 22 is not feasible.

The R-Learner solves this problem with a two step process:

1. Divide up the data into Q (typically set to 5 or 10) evenly sized folds. Let $q(\cdot)$ be a mapping from the $i = 1, ..., n$ sample indices to Q evenly sized data folds, and fit $\hat{m}$ and $\hat{e}$ with cross-fitting over the Q folds via methods tuned for optimal predictive accuracy, then

2. Estimate treatment effects via a plug-in version of 22, where $\hat{e}^{(-q(i))}(X_i)$, etc. denote predictions made without using the data fold the i-th training example belongs to,

$$\hat{\tau}(.) = argmin_{\tau} \left[\hat{L}_n \{\tau(.)\} + \Lambda_n \{\tau(.)\} \right]$$

$$\hat{L}_n \{\tau(.)\} = \frac{1}{n} \sum_{i=1}^{n} \left[\left\{ Y_i - \hat{m}^{(-q(i))}(X_i) \right\} - \left\{ W_i - \hat{e}^{(-q(i))}(X_i) \right\} \tau(X_i) \right]^2 \tag{23}$$

In other words, the first step learns an approximation for $\hat{m}$ and $\hat{e}$, and the second step optimizes the r-loss function $\hat{L}_n \{\tau(.)\}$.[25]

3.2.3.7 S-Learner

The S-Learner estimator models $Y(0)$ and $Y(1)$ through one model that receives the treatment assignment W as an input feature (along with the features X. The estimated CATE is given by:

$$\hat{\tau}(x) = E[Y|X = x, W = 1] - E[Y|X = x, W = 0] = \hat{\mu}(x, 1) - \hat{\mu}(x, 0) \tag{24}$$

In other words, in the S-Learner, the treatment indicator is included as a feature similar to all the other features without the indicator being given any special role. Thus, $\hat{\mu}(x, w)$ can be estimated using any base learner (supervised machine learning or regression algorithm) on the entire data set.

3.2.3.8 Uplift Modeling - Model Selection

In any machine learning project, there are multiple options to try when it comes to **training** a model. In the previous section we described four possible approaches to the Uplift problem, each of them focusing on one thing: estimate the **Conditional Average Treatment Effect**.

The CATE is used as a way to solve the already mentioned **fundamental problem of causal inference**: If causal effects are statements about the difference between what

happened and what could have happened, then **causal effects cannot be measured**. That's bad news. You can arrange things so that you can observe either what happens if someone gets a treatment or what happens if they do not get the treatment. Yet, for the same person, you will never be able to observe both of these outcomes and hence also not the difference between them.

However, even though we cannot observe whether X causes Y for any given unit, it can still be possible to figure out whether X causes Y on **average**. The key insight here is that the average causal effect equals the difference between the average outcome across all units if all units were in the control condition and the average outcome across all units if all units were in the treatment condition.[15]

The problem with looking at average treatment effects only is that it takes attention away from the fact that treatment effects might be very different for different sorts of people. While the fundamental problem of causal inference suggests that measuring causal effects for individual units is impossible, making inferences on groups of units is not.

Random assignment ensures that treatment is independent of potential outcomes and any (observed and unobserved) covariates. Sometimes, however, we have additional information about the experimental units as they existed before the experiment was fielded, say X_i, and this information can can help us understand how treatment effects vary across subgroups. For example, we may suspect that men and women respond differently to treatment, and we can test for this heterogeneity by estimating conditional ATE (or CATE) for each subgroup separately (equation 10)

If our covariate is continuous, we can test its moderating effects by interacting the continuous variable with the treatment. Note, however, that the treatment effect is now conditional on both treatment status and the value of the conditioning variable at which the effect is evaluated.[38]

The question remains, **which approach provides the best results for estimating the CATE?** To answer this question we will explain the training process we applied in general (as it was common to all the approaches, only changing the model trained), followed by the performance comparison, which gave us the necessary insight to choose one.

3.2.3.9 Machine Learning Training Basics

Before explaining the implemented process for the Uplift solution, it is important to introduce some concepts, complementary to those provided in the Machine Learning section, in order to gain a general understanding in the process of training a Machine Learning model.

We have already explained about what are Machine learning models, and in the previous section we dived into the sub-universe of Uplift modeling. Now it is time to explain some core concepts that will help to us to understand better the next section in where the actual process will be introduced.

Model Selection

After gathering your data (let us assume that it is clean and it is ready to use), comes the moment of what is called **modelling**, which is the process of defining the model we will use, how the data is going to be introduced, etc.

When modelling, there are a lot of decisions to be taken,e.g.:

- What pre-processing should we do to the data?

- What model should we use?
- What hyper-parameters should we train the model with?

Picking a good model that has a good performance in unseen data is a complex task. In general, we will have many observations, many variables and models to choose.

The end goal of a model is to perform well in new, unknown data. That is, after we train it to **learn** from what we call a **training** dataset, we would like to use it to predict an outcome for **new** data. If it only performed well on data that it has already seen, then it is not a very useful model because it is has not learned anything, only memorised.

Of course, we cannot know for sure the output of a truly unseen data register, as we do not have it yet, thus making the evaluation of a model impossible for those cases. For that reason, we would like to have an estimation of how our modelling decisions are going to impact in the model's performance, but in the **unknown data performance.**

Such problem can be faced by simulating, in some way, the division between "known data" and "unknown data". The "known data" will be used for training the model with certain parameters and characteristics. After that, we will use the "unknown data" (from now on known as **test set** to validate such parameters.

What we are proposing is to *hold* out a subset of the training observations from the training process, and then applying the statistical learning method to those held out observations.

The simplest way to simulate the behaviour on unknown data is the **validation set approach**. It involves randomly dividing the available set of observations into two parts, a **training set** and a **test set** or **hold-out** set. The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the test set. The resulting test set error rate provides an estimate of the test error rate.[18]

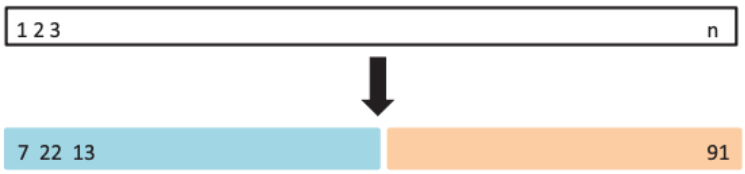


Figure 31. Schematic display of the validation set approach.[18]

However simple, the hold out set approach has two main drawbacks:

- The performance prediction can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the test set.
- Given the fact that in this approach we are only using a subset of the available data for training (those observations included in the training set), it could happen that if it is not big enough (few observations) then the performance could be pessimistic Although this is not a problem if there are a lot of data.

An alternative to the **hold out set** approach that tries to solve its drawbacks is the **Leave-one-out-cross-validation**

However, instead of creating two subsets of comparable size, a single observation (x_1, y_1) is used for the test set, and the remaining observations $(x_2, y_2), ..., (x_n, y_n)$ make

up the training set. The statistical learning method is fit on the $n-1$ training observations, and a prediction $\hat{y}_1$ is made for the excluded observation, using its value x_1. Since (x_1, y_1) was not used in the fitting process, we get an unbiased estimate for the test error. However, although it is an unbiased estimate, it is a poor one as it is highly variable since it is based upon a single observation.

To solve this, we can repeat the process n times, using a new observation each time. The model performance can be estimated as an average of all the n fitting models:

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} Perf_i \tag{25}$$

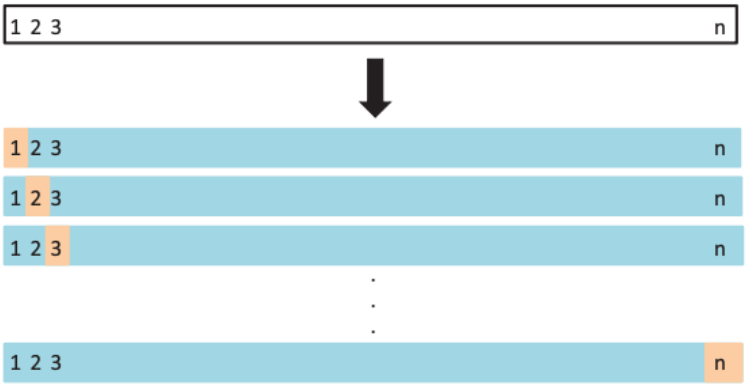


Figure 32. Schematic display of LOOCV approach.[18]

LOOCV technique has many advantages, for example, the results do not depend on a random subset of the data as it uses almost all of the data for training, taking advantage of learning from many data registers. Also, each observation is used for validation. However, the main drawback is that it requires a lot of processing (it runs as many trainings as observation exists) and, when looking at new data, the performance has high variance.

Finally, a middle ground between the two previous approaches is the ***k-fold* cross validation** approach. This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a test set, and the method is fit on the remaining $k-1$ folds. The performance, is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a test set.

The most obvious advantage is computational. LOOCV requires fitting the statistical learning method n times. This has the potential to be computationally expensive especially if n is extremely large. Instead, ***k-fold* cross validation** requires to fit a model only k times (number of folds). Of course, if k is big enough (equal to n), then it would be the same as LOOCV. But if the number of folds is reasonable (between 3 and 10 folds is considered acceptable), then it requires much less processing.

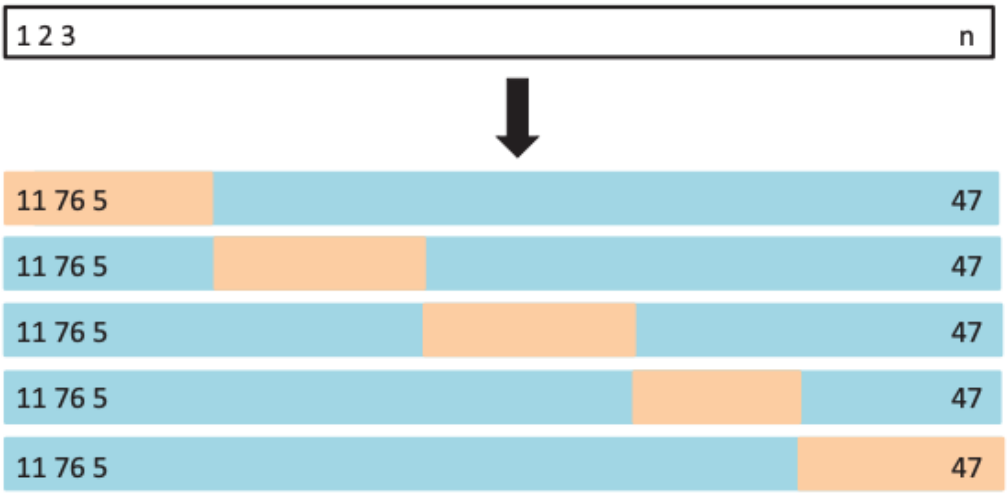


Figure 33. Schematic display of k-fold CV approach.[18]

Overfitting & Underfitting

When training a model, it is important to avoid two possible scenarios: **overfitting** and **underfitting**.

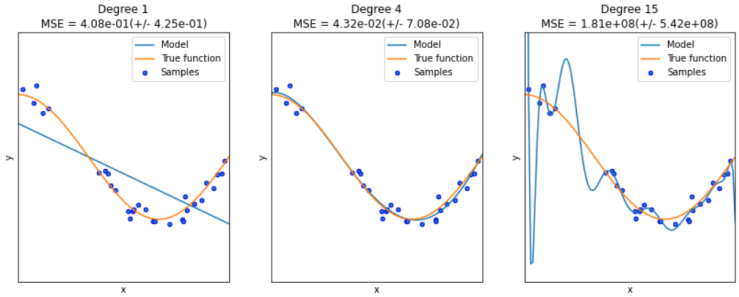


Figure 34. Overfitting and Underfitting[40]

- Overfitting

  It is an over-generalization that can occur when training a model: it means that it performs well on the training data, but it does not generalize well. Overfitting happens when the model is too complex relative to the amount and noisiness of the training data, meaning that it adjusts too much to the peculiarities of the training data and, consequently, performs poorly in the test set.

  Usually, this problem arises when there is not enough data, the model is too complex (too many parameters or features) or when the data is too noisy (too many missing data, outliers, etc.). Therefore, the most common ways to manage this problems are:

  – Simplify the model by reducing the number of features, constraining it or reducing the number of parameters

  – Increment the training data

– Cleaning the data, thus reducing its noise.

- Underfitting

  It is the opposite case of overfitting. Here, the algorithm is too rigid, too inflexible to such a level that it cannot capture relevant patterns in the data. It is for this reason that models that are underfitted perform poorly both in training and testing. Some possible actions to be taken when facing this problem would be:

  – Test a more complex model: Add features, parameters and reduce the constrains to its learning

  – Produce better features through *feature engineering*

3.2.3.10 Uplift Modeling - Training Process

The pipeline for training uplift models was built in a **Lab** in **FDA**, this ensured a hosted environment where we could easily push scripts into production without wasting time. The following process was designed in a way that it was usable for any of the uplift approaches mentioned earlier, this way, we could explore different alternatives without so much code refactoring.

A general view of the training process is presented in figure 35 below:

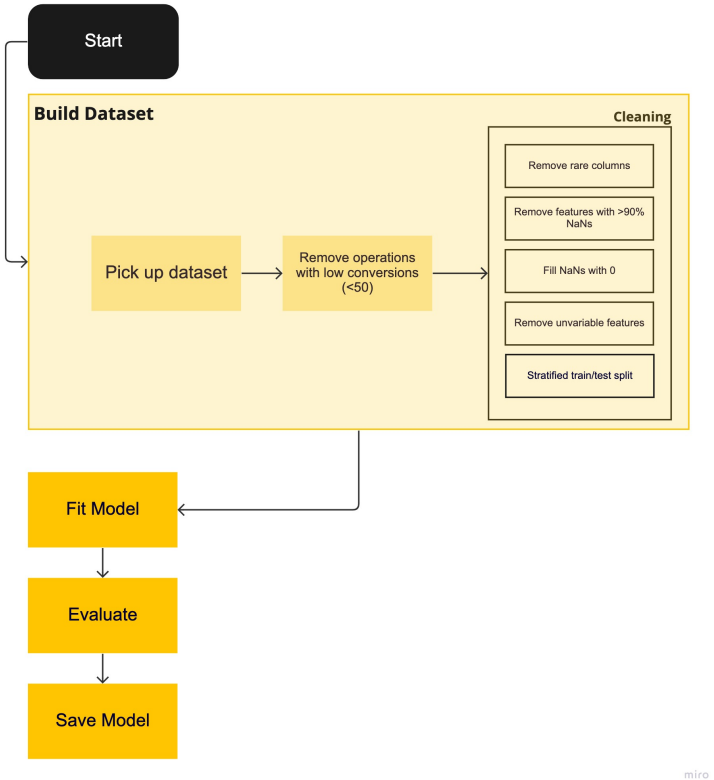


Figure 35. Uplift training process

It is clear that the process itself is not complex, however some of it tasks require more explaining as they are particular to this problem and this dataset.

Balancing Data

For starters, we could say that one of the most important, if not the most, is the first one, that is, the **building of the dataset**. As it has been said throughout this work, data is the raw material which we start from when building a machine learning project. As such, it is important that it is curated, clean and in the best shape possible for it to actually be useful for the models, otherwise, the results could be disappointing.

The first step then is the preparation of the dataset, which includes downloading and cleaning it. The dataset in use was built from two tables: **feature set table** and **target set table**. However, before downloading it, it pass through a **balancing** process where we take the positive cases (output feature in **target set table** equals 1) up to a "good" proportion when compared to the negative cases. This is needed because when studied, the balance of the classes presented the following results:

Operation	Converted	Non Converted	Total	%Non Converted	% Converted
Account Fund - Normal	1186	274523	275709	99,57%	0,43%
CG	3135	435158	438293	99,28%	0,72%
CRYPTO - BUY	690	49213	49903	98,62%	1,38%
Card Payments - Cards	5222	768542	773764	99,33%	0,67%
Credit Application	5657	381093	386750	98,54%	1,46%
Wallet Antenna Recharge	20	52855	52875	99,96%	0,04%
Wallet Cellphone Recharge	3853	223829	227682	98,31%	1,69%
Wallet Digital Goods	37	76052	76089	99,95%	0,05%
Wallet Instore	1945	360411	362356	99,46%	0,54%
Wallet Money Transfer - Send	1695	120312	122007	98,61%	1,39%
Wallet Transport	364	79681	80045	99,55%	0,45%
Wallet Utilities	1649	135549	137198	98,80%	1,20%

Table 21. Unbalanced Data

Table 21 shows the number of users that **"Converted"** and those who did not in each treatment. This checking process is common in a machine learning project as it helps to understand how balanced are the target classes. If the positive cases are very low when compared to the negative cases (for reference, a percentage under 4% or 3% is considered low) then we say that the dataset is **imbalanced**. Why is this a problem? Because when training is done, the abundance of negative cases will generate a bias in the model towards those cases, meaning that it will predict more zeros than ones (in a simple classification model) making more difficult to correctly identify the positive cases.

There are many techniques for handling these cases that fall out of the scope of this work. However, one of the most common is known as **down-sampling**, which consists of removing some of the negative cases in a random manner in order to achieve an increase in the proportion of positive cases. As the total number of cases is reduced while keeping the positive ones, their ratio increases. One important thing to keep in mind when doing this technique is that there is a trade-off between achieving a balanced dataset and altering it to the point where it is no longer representative of the real world. This means that if we down-sample too much, for example to a point o 50/50 ratio, although it will help the training, when tested in real unseen data, that ratio will not be real and therefore, it will not perform correctly.

The results presented in table 21 shows a big imbalance in the data, where in most of the treatments we see that there are not many positive cases. This would be troublesome for the training. For this reason we did a down-sampling process in order to take the converted ratio up to a 7%. Such ratio was established intentionally as a "good practice". We wanted to have a good signal for the model to train correctly, thus the selected percentage. The result were the following:

Operation	Converted	Non Converted	Total	%Non Converted	% Converted
Account Fund - Normal	929	13271	14200	93,46%	6,54%
CG	2208	31542	33750	93,46%	6,54%
CRYPTO - BUY	752	10742	11494	93,46%	6,54%
Card Payments - Cards	3439	49128	52567	93,46%	6,54%
Credit Application	4852	69314	74166	93,46%	6,54%
Wallet Antenna Recharge	17	242	259	93,44%	6,56%
Wallet Cellphone Recharge	2987	42671	45658	93,46%	6,54%
Wallet Digital Goods	33	471	504	93,45%	6,55%
Wallet Instore	1574	22485	24059	93,46%	6,54%
Wallet Money Transfer - Send	1326	18942	20268	93,46%	6,54%
Wallet Transport	285	4071	4356	93,46%	6,54%
Wallet Utilities	1331	19014	20345	93,46%	6,54%

Table 22. Balanced Data

This new, balanced, dataset is the one downloaded in the first step of the training process in Figure 35 and is the one we use throughout the pipeline. This resulted in a dataset of **295 features and around 302.000 observations.**

Cleaning Process

After balancing the data, the next natural step is to revise and clean it. The reason for this, as will be mentioned, is because if kept some features will affect the training process as they will provide incoherent or incomplete information. Other type of cleaning is not related with removing features but with handling missing data. The cleaning process consists of the following tasks:

- Removing operations with few conversions

  Some operations that were included for the model at first, turned out to have a very poor signal, meaning that the number of users that operated in those categories was very low. Although the reduce number is a problem in itself (as explained during the *Balancing Data* section), and even though the dataset was down-sampled to a 7% for the positive cases, the low number also affects the training process as it makes splitting the data practically impossible for those operations. As will be explained later in the *Training* task, we trained the Uplift model using a **Hold out set** (see Machine Learning Training Basics) approach for validating the model performance in unseen data. This method implies splitting the data into two subsets, both the features Xi and outputs y_i (whether a user converted or not). However, if the number of $y_i = 1$ is too low (as in this case) then splitting is not possible as there would be no observations available for the test set.

  This problem was happening for the operations of **Antenna Recharge** and **Digital Goods**, where we see that there are 17 and 33 positive observations respectively (see table 22). For this reason, in this part of the process of the training, we designed an algorithm that removed any operation that had less than 50 conversions in the whole dataset. Such threshold was established through trial and error while we tried to split the data but kept getting errors.

- Removing unnecessary features

  This step is a consequence of the *building dataset* task as it ran a query to the **target** and **feature** datasets. As a result, some duplicated columns were created. Also, in the process of *balancing* the data, which was done from a Lab that ran a query to

BigQuery, some columns were created as support for achieving the down-sampling and were not removed. On the other hand, there were some features that were not duplicated but required to be removed as they were not relevant for the model,e.g., the **user id** and **country** columns. The first one because it is an identifier with no valuable information in it and the latest because, as we will train a different model for each country, there was no need for that information. As the name suggests, we identified such columns and removed them from the dataset.

- Handling missing data

  The "appearance" of missing data is not something rare but quite common in all data-related projects. The concept of missing data is implied in the name: it's data that is not captured for a variable for the observation in question. Missing data reduces the statistical power of the analysis, which can distort the validity of the results[14]. Missing values are usually attributed to: human error when processing data, machine error due to the malfunctioning of equipment, respondents refusal to answer certain questions, drop-out in studies and merging unrelated data. The missing values problem is usually common in all domains that deal with data and causes different issues like performance degradation, data analysis problems and biased outcomes lead by the differences in missing and complete values. Moreover, the seriousness of missing values depend in part on how much data is missing, the pattern of missing data, and the mechanism underlying the missingness of the data. There are many ways to handle the presence of missing data

  When dealing with missing data, there are usually two possible ways to deal with that error: **imputation** of a certain value or **removal** of data.

  - *Imputation*

    It basically consists of replacing the missing data for "reasonable" guesses. Of course, it is most useful when the percentage of missing data is low, otherwise, the dataset would not be real anymore as there would be more "guessed" data than actual information, meaning that the results would lack natural variation, affecting the model.

    There are different ways to impute missing data, we will just mention a few without going into details:

    * Mean, Median and Mode: Replace missing data with any of those statistics.
    * LOCF & NOCB: In Last Observation Carried Forward and Next Observation Carried Backward methods every missing value is replaced with the last observed value. This method may introduce bias when data has a visible trend. It assumes the value is unchanged by the missing data.
    * Linear Interpolation: It is often used to approximate a value of some function by using two known values of that function at other points. It is useful in a time series that exhibits a trend line, but it's not appropriate for seasonal data.
    * K-Nearest Neighbours: A certain parameter is defined, known as the *neighbourhood* k. It is a distance that is used to determine the nearest observations that fall in that neighbourhood. Once it is defined, the missing data will be imputed by estimating the average of the values in the proximity determined by k.

– *Deletion*

The other option is to remove data. It may not be the best option if there are not enough observations to result in a reliable analysis. Removal can be done at an observational level (remove rows with a certain level of missing values) or at feature level (completely remove a column because of its percentage of missing information). However, in most cases it is not wise to remove data as it always implies the lost of information, which we have already established is very important. Some literature will recommend that after a certain volume of missing data (in a feature) it is wise to remove it. Nevertheless, sometimes, the absence of data is information in itself, and discard it so rapidly could end up affecting the final model.

In our case, we chose to go for the **deletion** solution. We explored the number of empty observations in each feature and decided to remove those which had more than 90% of their observations missing. This approach was taken because of the nature of the problem, it was normal to not have all the information available. In some cases, the fact that we did not have data could mean that a user did not interact with the app at a certain time, therefore, by imputing some hardcoded value would have been inventing an interaction that never happened. Because of this reason, we chose to keep the data, removing only those extreme cases. For us, more than a 90% of missing data simply means that a feature is not informative. After removing such features, we filled the remaining missing data with zeros, as the Uplift models described before do not support missing data. The result of this task is the removal of a total of **89 features**.

- Removing invariable data

  The presence of variables that remain immutable throughout the entire dataset is detrimental to the model because they do not provide any valuable information, therefore adding noise. For this reason we evaluated all features and determined their variability by obtaining the frequency of the values that composed each variable. For those cases in which the frequency exceeded the 98%, that is, features containing a value that repeated itself in more than 98% of the total observations, were removed as they were practically constant.

  The result of such cleaning was that **50 more features** were excluded from the dataset.

Train/test split

When introducing the basic concepts of a Machine Learning project, we mentioned the different methods that exists for simulating the performance of any model over **unseen** data (observations it did not use for training). Because of the number of observations we had for training, we decided to go with a **hold-out set** approach. For this, we splitted the data into two groups of observations:

1. The observations that were going to use for training, called the **training set**
2. The observations that were going to use for evaluating the model, called the **test set**

Because we did not have too many observations (less that 500.000) we could not hold out too many observations for evaluation as it would affect the training process. For this reason, we decided for a 70/30 approach: 70% of the observations for training, and the remaining 30% for testing.

As for the methods for such splitting, we used **scikit-learn**, a python library dedicated to machine learning. In it we found a function that can automatically split the data according to the ratio the user specifies. An important aspect to keep in mind was related to the way in which we should separate the data. Some problems can exhibit a large imbalance in the distribution of the target classes: for instance there could be several times more negative samples than positive samples. In such cases it is recommended to use **stratified sampling** to ensure that relative class frequencies is approximately preserved in each train and validation fold.[26]

When looking at table 22 we noticed that the operations are distributed as follows:

Operation	**Portion related to total**
Account Fund - Normal	4,71%
CG	11,19%
CRYPTO - BUY	3,81%
Card Payments - Cards	17,43%
Credit Application	24,59%
Wallet Antenna Recharge	0,09%
Wallet Cellphone Recharge	15,14%
Wallet Digital Goods	0,17%
Wallet Instore	7,98%
Wallet Money Transfer - Send	6,72%
Wallet Transport	1,44%
Wallet Utilities	6,75%

Table 23. Operations distribution in the dataset

Table 23 is proving that the operations, which in the scenario of Uplift modelling are considered **treatments**, are not evenly distributed, meaning that the splitting should be done carefully to ensure data consistency. If we also consider that the number of positive observations are very scarce, we get a complex problem where we must consider two variables: **treatment** and the **output**. We want to separate the observations in a way that can preserve the distribution of both features in training as well as in testing. By applying a **stratified sampling**, we guarantee that each set contains approximately the same percentage of samples of both operation and target classes as the complete set. The results can be seen in tables 24 and 25 below, were the percentages are kept in the **output** and the **treatment** variables, both in training and testing datasets.

Operation	Converted	Q	Output Ratio	Treatment Ratio
Account Fund - Normal	0	7825	93,29%	4,96%
Account Fund - Normal	1	563	6,71%	
CG	0	17968	93,50%	11,37%
CG	1	1249	6,50%	
CRYPTO - BUY	0	2991	93,50%	1,89%
CRYPTO - BUY	1	208	6,50%	
Card Payments - Cards	0	34251	93,52%	21,66%
Card Payments - Cards	1	2374	6,48%	
Credit Application	0	41783	93,67%	26,38%
Credit Application	1	2825	6,33%	
Wallet Cellphone Recharge	0	18836	93,29%	11,94%
Wallet Cellphone Recharge	1	1355	6,71%	
Wallet Instore	0	14031	93,14%	8,91%
Wallet Instore	1	1034	6,86%	
Wallet Money Transfer - Send	0	9079	93,66%	5,73%
Wallet Money Transfer - Send	1	615	6,34%	
Wallet Transport	0	1536	93,26%	0,97%
Wallet Transport	1	111	6,74%	
Wallet Utilities	0	9780	93,65%	6,18%
Wallet Utilities	1	663	6,35%	

Table 24. Training set distribution

Operation	Converted	Q	Output Ratio	Treatment Ratio
Account Fund - Normal	0	3374	93,85%	4,96%
Account Fund - Normal	1	221	6,15%	
CG	0	7689	93,36%	11,37%
CG	1	547	6,64%	
CRYPTO - BUY	0	1280	93,36%	1,89%
CRYPTO - BUY	1	91	6,64%	
Card Payments - Cards	0	14648	93,32%	21,66%
Card Payments - Cards	1	1049	6,68%	
Credit Application	0	17774	92,97%	26,38%
Credit Application	1	1344	7,03%	
Wallet Cellphone Recharge	0	8121	93,85%	11,94%
Wallet Cellphone Recharge	1	532	6,15%	
Wallet Instore	0	6083	94,21%	8,91%
Wallet Instore	1	374	5,79%	
Wallet Money Transfer - Send	0	3863	92,99%	5,73%
Wallet Money Transfer - Send	1	291	7,01%	
Wallet Transport	0	663	93,91%	0,97%
Wallet Transport	1	43	6,09%	
Wallet Utilities	0	4162	93,01%	6,18%
Wallet Utilities	1	313	6,99%	

Table 25. Test set distribution

Training process - Base learner: XGBoost

Having prepared the dataset, everything is ready for the training of the Uplift model. Up to this point, we still have not chosen any specific learner to estimate the CATE, however, we will make use of the test set we separated. We intend to try the different approaches explained in the previous sections and evaluate their performance through some metrics we will introduce later on. Such performance must be estimated over the "new" observations reserved in the test set, giving us an idea of how well they are generalizing.

For the implementation of the models we used a Python library called **CausalML**, it is a package that provides a suite of uplift modeling and causal inference methods using machine learning algorithms based on recent research. It provides a standard inter-

face that allows user to estimate the Conditional Average Treatment Effect (CATE) or Individual Treatment Effect (ITE) from experimental or observational data.[6]

This package is not the only one that can solve this kind of problems, however, the fact that it was very well documented, with many practical examples and that it supported many methods for estimating the CATE (besides the ones we studied) made this the best course of action. It is important to remember that as part of a company project, the time was of the essence, for this reason we decided to focus on one tool and understand it deeply, rather than trying multiple packages, without really knowing how to use it properly.

When studying the different methods for estimating the CATE, it was explained that they all had an estimator at its core , a base model which basically predicted a propensity score. It is then expected that for the CausalML package to work, it requires to be specified some kind of model. In one line we declared the kind of **classifier** that we wanted to use as base model.

As base model we used an **XGBoost Classifier**, one of the most popular models used in important machine learning competitions. This model has demonstrated to produce great results in most applications, reason why it was our first choice. The user should remember that we were not trying to find the best base model but the best CATE estimator, the first one being a simple mean to an end.

XGBoost is a **decision-tree-based ensemble** Machine Learning algorithm that uses a **gradient boosting framework**. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now.[24]

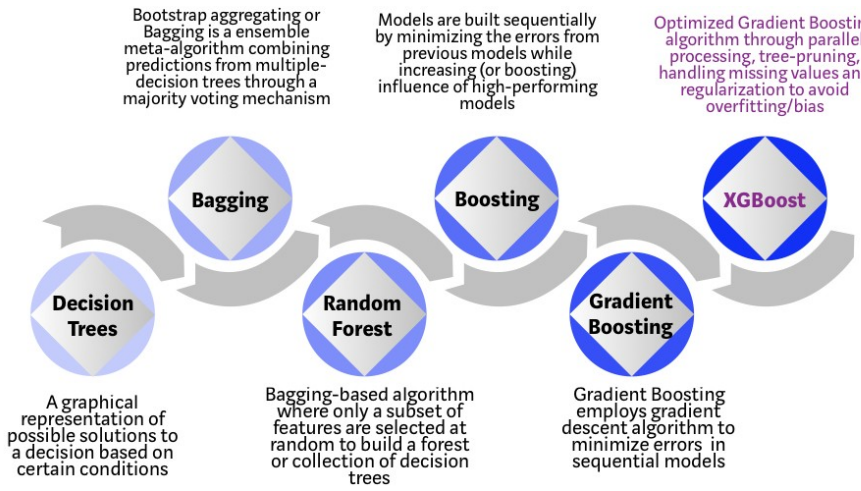


Figure 36. Evolution of XGBoost Algorithm from Decision Trees [24]

What is an **ensemble** method?. Sometimes, it may not be sufficient to rely upon the results of just one machine learning model. Ensemble learning offers a systematic solution to **combine the predictive power of multiple learners**. The resultant is a single model which gives the aggregated output from several models.

The models that form an ensemble, also known as base learners, could be either from the same learning algorithm or different learning algorithms. Bagging and boosting are two widely used ensemble learners. Though these two techniques can be used with several statistical models, the most predominant usage has been with **decision trees**.

Decision trees create a model that predicts the label by evaluating a tree of if-then-else true/false feature questions, and estimating the minimum number of questions needed to assess the probability of making a correct decision. Decision trees can be used for classification to predict a category, or regression to predict a continuous numeric value.[43]

Let's briefly discuss bagging before taking a more detailed look at the concept of boosting.

- **Bagging**

  While decision trees are one of the most easily interpretable models, they exhibit highly variable behavior. For example. if we were to train two decision trees models on two subsets of data from the same origin, both models would yield different results.

  Decision trees are said to be associated with high variance due to this behavior. Bagging or boosting aggregation helps to **reduce the variance** in any learner. Several decision trees which are generated in parallel, form the base learners of bagging technique. Data sampled with replacement is fed to these learners for training. The final prediction is the averaged output from all the learners.

- **Boosting**

  In boosting, the trees are built **sequentially** such that each subsequent tree aims to **reduce the errors of the previous tree**. Each one learns from its predecessors and updates the **residual errors** (the difference between the expected value and the predicted value). Hence, the tree that grows next in the sequence will learn from an updated version of the residuals. The base learners in boosting are weak learners in which the bias is high, and the predictive power is just a tad better than random guessing. Each of these weak learners contributes some vital information for prediction, enabling the boosting technique to produce a strong learner by effectively combining these weak learners. The final strong learner brings down both the bias and the variance.

The term **"gradient boosting"** comes from this idea of improving a single weak model by combining it with a number of other weak models. It is an extension of boosting where the process of additively generating weak models is formalized as a **gradient descent algorithm** over an objective function. This approach sets targeted outcomes for the next model in an effort to minimize errors. Targeted outcomes for each case are based on the gradient of the error (hence the name gradient boosting) with respect to the prediction. For more information on the math behind these cutting-edge models, the reader could explore the original document of XGBoost in the reference section.

Training Process - Fitting models & Evaluation

After selecting the base estimator for the meta-learners to estimate the CATE, we **fitted** them (term used to say that a model started training on some data) to the training set we created and proceeded to evaluate the results on the test set. Of course, to be able to compare the different models, a metric needs to be established that allow us to evaluate them equally. For this purpose we chose one "technical" metric and developed a few "business" like metrics:

- **Technical metric**: Area under the uplift curve **(AUUC)**

As mentioned before, we stumble with the problem that it is not possible to observe both the control and the treatment outcomes for an individual, which makes it difficult to find a loss measure for each observation. The AUUC is a common approach that consists in first predicting uplift for both treated and control observations and compute the average prediction per decile in both groups. Then, the difference between those averages is taken for each decile. This difference thus gives an idea of the uplift gain per decile. To have a clearer idea of the models performance, this approach then calculates the cumulative decile chart, where the leftmost bar corresponds to the uplift in the first 10 percent, the following bar corresponds to the 20 first percent and so on. A well performing model features large values in the first quantiles and decreasing values for larger ones. This is useful because it allows to easily see if the treatment has a global positive or negative effect and if we should expect a better gain by targeting part of the population. We can thus choose the decile that maximizes the gain as the limit of the population to be targeted. Finally, to produce an *actual* metric to compare between models, we generalize the cumulative gain chart for each observation of the test set with the following parametric **uplift curve** defined for each t as:

$$f(t) = \left(\frac{Y_t^T}{N_t^T} - \frac{Y_t^C}{N_t^C}\right)\left(N_t^T + N_t^C\right) \tag{26}$$

Where Y_t^T (respectively Y_t^C) and N_t^T (respectively N_t^C) are the sum of the treated (respectively control) individual outcomes and the number of treated (respectively control) observations in t, the first t observations, sorted by inferred uplift value. The continuity of the uplift curves makes it possible to calculate the area under the curve as a way to evaluate and compare the different uplift models.[13]

- **Business-like metric**: Top 1 Accuracy and Top n Accuracy

  Both of this metrics are not "official" as they were produced during the development of this project. They were created with the idea of an **Accuracy** metric, that is, how good was the model at suggesting an operation the user would interact with. This separates from the concept of uplift that the model aims for (whether or not the suggestions can impact the user), but it allows to understand how good are the suggestions by looking at the actual action the user did.

  The idea behind is simple: It determines how many "good predictions" the model had over the total number of predictions. If the model said someone would do A and he actually did, then it counts as one, otherwise, it is a zero. Summing all the ones over the total number of predictions would give the accuracy of the model. However, the Uplift model do not "predict" in the conventional way, it merely "suggest" what would be good, bad or indifferent for a user, meaning that could be more than one operation. For that reason we built two metrics:

  - *Top 1 Accuracy*: The model "predicted" correctly if a user operated in the **first** of the model suggestions. If he used the second choice of the model or other, then the model failed.
  - *Top n Accuracy*: It is similar to Top 1, but we will consider that the model "predicted" correctly if a user operated in **any** of the model suggestions.

By comparing these metrics between the learners, we will choose the "best" one and use it to generate recommendations once the process is pushed into production.

For the Accuracy metric, a threshold had to be set, just like in a classification model, as the learners returned a certain uplift/gain value, however it does not say up to what decile of the population we should consider, as that is a business decision. Another way to put it is that although the learners produce a gain value for each operation, we do not know up to what level of gain we should consider that the operation will impact the user. In order to do that, we based on the Uplift Curves (from the AUUC metric), which look like something as follows:

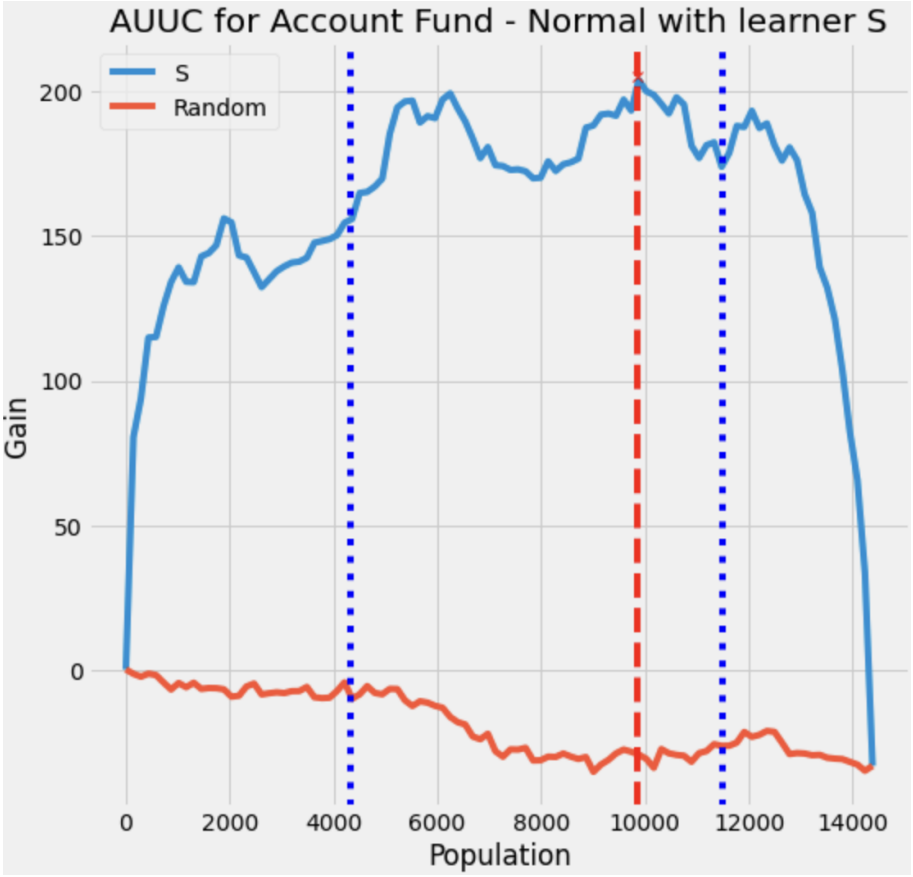


Figure 37. Uplift curve example

The values located more to the left are related to the highest values of CATE, and from there it starts decreasing to the right. The ascending parts of the blue curve indicate that the operation has a positive impact on the user, while the descending ones generate a negative effect. The first logical option would be to set the threshold at the point where the CATE reaches its highest point as that part would guarantee the best gain we could achieve. However, in some cases that meant selecting less than 10% of the population, which is not ideal as we were trying to reach all the users we could. Because of this, we set boundaries in where we could chose. Such limits were placed based on the Business Experts knowledge. For starters, we asked what was the minimum percentage of population they wanted to target and, on the other end, what was the maximum level of population that they were willing to keep. This way, we made sure that we were not selecting too few users nor the whole population (as it would be a useless model). Their answer was that those boundaries should be between 30% and 80% of the total population.

That meant that any CATE that gathered less or more than those population limits should not be used as threshold. With those limits in place, we set a rule where we would choose the highest CATE found **between** them.

With the thresholds in place, we obtained the metrics for all learners, for the **test set**, which we registered in table 26.

Learner	Mean AUUC	Top 1 Accuracy	Top n Accuracy
S	16.8194	45.7365	77.8346
T	15.1954	31.0409	61.8146
X	18.8875	31.7728	65.8225
R	15.3358	15.9038	72.1654

Table 26. Uplift metrics results

The mean AUUC is estimated taking the individual metric from each operation (as the Uplift model returns a value for each operation available). From only looking at this metric, it would seem as if the meta-learner X was the best choice. However, when looking at the other two, the meta-learner S excelled when compared to the rest, not to mention that it rated second on the AUUC scale. Also, in terms of **training time**, both learners S and T were much faster that their opponents, X and R, by several minutes. Taking all of this into account, and the fact that it is one of the simplest models of the lot, we determined that **meta-learner S** was the estimator that performed better and therefore, our choice for a final Uplift model.

Usually, a normal Uplift curve is shaped like the one in Figure 37, sometimes a bit skewed to the left or to the right, depending on how well the model is performing. The ideal shape would be one that has the higher gains located to the left of the chart, and reduced gains on the right. That way it would mean that the best scores are located in the users with the highest CATE. A shape like 37 is not so clear as it shows that some parts perform better that others. However it still indicates that the model can detect certain regions where it can produce more gain. All the Uplift curves produced by the learners are illustrated in Apendix A.

3.3 Deployment

Both models have been explained in full detail, we got our predictions for both solutions, now what remains is their integration, how does it all work together and how does we finally communicate our users about our suggestions.

The end to end process can be summarized as follows:

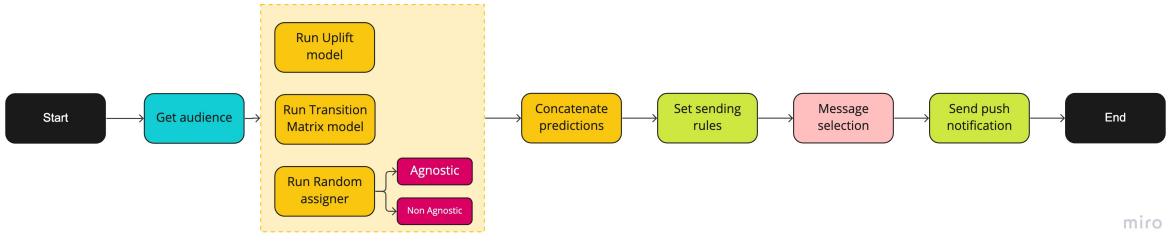


Figure 38. Push Notification process

The whole process runs **every day**, meaning that each day we score users, split them and send them notifications. However, the part of sending notifications runs once every hour. In other words, at the beginning of the day, we get the users, get a suggestion, set the rules, select a message and send notification (if possible at that moment). After that, by the end of the process we save those suggestions predicted by our models, as well as the users that we could not send any notification. We do this because for the rest of the day, for every hour, the process will begin again, without predicting again, only for evaluating if the users that we did not reach in the previous run are available now. This will make more sense when the concept of the **Trigger** is explained in section 3.3.5.

As presented, the process is pretty straightforward and simple, however, some of its tasks have yet to be explained for everything to make sense.

3.3.1 Get Audience

The first thing to do is to get the users that we want to make our suggestions to. Because of the nature of the habit problem introduced throughout the project, we know that we are interested in the users which satisfy the following conditions:

- Have installed the fintech application at the time the process runs.
- The user had **activated**, meaning he did **at least** one operation **in the last 30 days**.
- The user have less than five different operations days at the time the process runs.
- The user is over-age (older than 18 years old).
- The user belongs to the country we are running the process for.

If they check all the boxes, then we consider them for the models. An important aspect of the experiment to be mentioned is the proportion of the total population we are addressing with our tests. This project coexists with many other processes that are running in production as part of the company's strategies. For this reason, we could not interfere with the totality of the users. Before going big, we needed to hold out a subset of users so we could run our models, determine whether or not they were successful and then move on to a full scale model. It is for this reason that from the total population, we restrict only to the 20% of it. We select that sample based on the users id, ensuring that we can track them later.

Such subset of users leaves us with 345.000 users approximately to score every day, the number might vary as new users come in and others come out.

3.3.2 Scoring

After identifying the audience, each one gets a suggestion for their next best operation from each of the solutions explained before - **transition matrix** and **uplift model** - However here we introduce two new elements to the predictions.

First of all, we developed a **Random model** that assigns a suggested operation to the users following a random sample process. In it, we take the audience, select an operation randomly and assign it to a user. This "model" was created for evaluation purposes as we wanted to have a benchmark against which we could contrast our solutions. The idea

was that if our models performed worst than a random assignment, then they were not good enough.

On the other hand, two new elements were introduced in the Uplift model module: **Agnostic** and **Non Agnostic**. These two exists because of a business management decision where they wanted to answer the following question: **is it OK to trust entirely on a machine learning model?**

Of course this question is not trivial, besides, because how it was designed, each time a user gets in the process and remains in the same step in the habit journey, then the uplift model would always (or almost always) recommend the same operations. From a business perspective, this made no sense, if a user do not respond to a suggestion, the logical thing to do would be to pass on to the next option.

However, because we also wanted to test how good was the model by itself, we created these two versions of the Uplift model. In one of them, we would trust the model entirely, without questioning the suggestion. Meaning that we would suggest the operation with the highest CATE in the list the model provided. That model was called **Agnostic**. For the other model, **Non Agnostic**, we would take the first suggestion from the model as before, however if the user did not respond to it, then the next time we would take the second best choice, discarding the first one. This process would repeat itself until there were no other options left, in which case we would no longer send notifications to that user, or the user change his state, therefore generating new suggestions.

This four models (Random, Transition Matrix, Uplift Agnostic and Non Agnostic) were ran for all users, without distinction, meaning that each user would have 4 suggestions by the end of this section. But which one should the process select?

In this case, again, we chose to divide the population by the user id. Because we had four models, we splitted the users in four groups. Each group would get the suggestions from one model only. This way, each user gets one option, and we ensure a simple way to track the experiments.

3.3.3 Setting rules for sending

Before sending a push notification, some cases should be excluded as the user's situation indicates that he is not yet ready to receive a message. For that, a series of rules are applied to the users in order to determine their situation and if whether or not they are available for a new notification.

These rules classify users into three sections:

1. **Do not send today**
2. **Send today**
3. **Send if trigger is activated**

In the first group are those users which have already been sent a notification, but the **attribution window** associated with it is yet to pass, meaning that the user could still operate in the category we sent him before. If another notification was sent to him, we would not be able to distinguish which one originated the conversion (if there was one). In essence, this group are users that could convert, therefore cannot receive anything yet.

The second group is composed of the users whose time since their last operation exceeded the **waiting time** assigned to them when scored. This means that they are out

of their attribution window but they have not done anything more since then, not even opened the app, therefore it is mandatory to send them the notification.

Lastly, those users who are tagged as **"send if trigger is activated"** are those users who are outside the attribution window, but their wait time is not over yet, meaning that we are observing if they activate our trigger (more of it in section 3.3.5) or not.

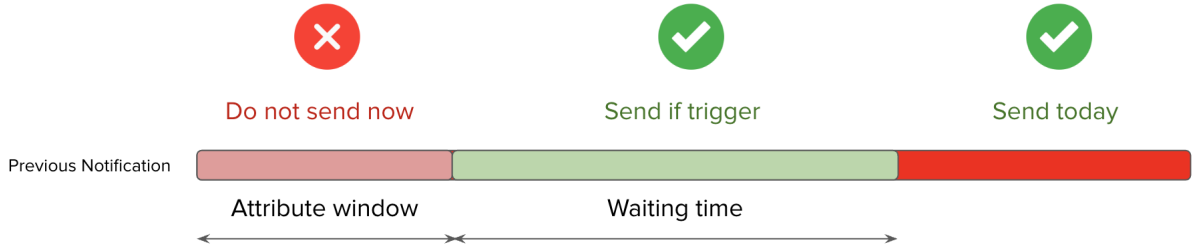


Figure 39. Rules for sending

3.3.4 Content selection - Thompson Sampling

So far all the concepts introduced in this work have always been related to the best way of suggesting a course of action to a user. Just as important as the operation is the **message** sent. A good idea with a bad message will not generate the same results as if the content was optimal. The message communicated to the users through the push notification needs to be selected in the best possible way.

For this task we chose another model that could find the best content to send by balancing two fundamental tasks: **exploiting** and **exploring**. The algorithm in question is called **Thompson Sampling (TS)**. It is an algorithm for online decision problems where actions are taken sequentially in a manner that must balance between **exploiting** what is known to maximize immediate performance and investing to accumulate new information that may improve future performance (**exploring**).

TS was born trying to solve a problem called **multi-armed bandit** problem. The colorful name for our problem comes from a motivating story in which a gambler enters a casino and sits down at a slot machine with multiple levers, or arms, that can be pulled. When pulled, an arm produces a random payout drawn independently of the past. Because the distribution of payouts corresponding to each arm is not listed, the player can learn it only by experimenting. As the gambler learns about the arms' payouts, she faces a dilemma: in the immediate future she expects to earn more by exploiting arms that yielded high payouts in the past, but by continuing to explore alternative arms she may learn how to earn higher payouts in the future.[34]

To understand how it works, lets build the following context. There are K actions. When played, an action k produces a reward of one with probability θ_k and a reward of zero with probability $1-\theta_k$. Each θ_k can be interpreted as an action's success probability or mean reward. The mean rewards $\theta = (\theta_1, ..., \theta_K)$ are unknown, but fixed over time. In the first period, an action x_1 is applied, and a reward $r_1 \in (0, 1)$ is generated with success probability $P(r_1 = 1|x_1, \theta) = \theta_{x1}$. After observing r_1, the agent applies another action x_2, observes a reward r_2, and this process continues.

We let the agent begin with an independent prior belief over each θ_k. These priors are beta-distributed with parameters $\alpha = (\alpha_1, ..., \alpha_K)$ and $\beta \in (\beta_1, ..., \beta_K)$. As observations are gathered, the distribution is updated according to Bayes rule, only the parameters of

a selected action. A beta distribution with parameters (α_k, β_k) has mean $\frac{\alpha_k}{\alpha_k+\beta_k}$ and the distribution becomes more concentrated as the denominator grows.

What TS does is, in each time period t, the algorithm generates an estimate $\hat{\theta}_k$, which is randomly sampled from the posterior distribution, which is a beta distribution with parameters α_k and β_k. The action xt with the largest estimate $\hat{\theta}_k$ is then applied, after which a reward r_t is observed and the distribution parameters α_k and β_k are updated. An illustrative algorithm of the process is presented in Figure 40.[34]

Algorithm 2 BernTS(K, α, β)

for $t = 1, 2, \ldots$ **do**
  #sample model:
  for $k = 1, \ldots, K$ **do**
    Sample $\hat{\theta}_k \sim \text{beta}(\alpha_k, \beta_k)$
  end for

  #select and apply action:
  $x_t \leftarrow \text{argmax}_k\, \hat{\theta}_k$
  Apply x_t and observe r_t

  #update distribution:
  $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t} + r_t, \beta_{x_t} + 1 - r_t)$
end for

Figure 40. Thompson Sampling Algorithm [34]

Using the success and failure rate of the previous contents sent to the users (here we consider success if a user opened a notification) as α_k and β_k parameters, we designed a Thompson Sampling algorithm that could determine the best content to send to a user based on the previous successful cases, in other words, it selected contents that had previously shown evidence of being suggestive to the users. The advantage of this method is that we could be sure that the text we were sending was optimized to obtain the highest chance of being noticed.

3.3.5 When should we communicate: The trigger

At the end of the process, after each user has been assigned a suggestion, marked with their situation (to send or not to send) and a specific text for the notification was chosen, it is time to send that message. Although the process is straightforward as it sends the information through API to the Hermes tool, the **moment** at which we should send it is not defined yet, should it send it immediately, tomorrow, a day after tomorrow? From previous projects, we have determined that the best indicator for determining the right time is what we call the **Trigger**. In essence, we explore the user's interaction with the app. If he had opened any of the apps in the ecosystem, then we will call that interaction a **navigation**. From our experience, the best time to send a user a notification is when he is more connected with the application, and the fact that the user is navigating gives us a strong signal that, at that moment, the user is undoubtedly connected. For this reason, we defined our trigger action as **the navigation in the last hour.** It is for this reason that we run the process once every hour for the whole day, as we are constantly checking if the users that did not raised the trigger before are doing it now. Also, it is here when the rules introduced in section 3.3.3 are of essence. Before checking for the trigger, we

discard those users that are not supposed to receive a notification in the day. Next, we check the navigation for the users in the last hour for both the other two groups (send today and send if trigger activated). The difference is that if a user from the first group does not navigate until a certain hour (4 p.m.) then we send a notification. On the other hand, if someone from the second group do not raise the trigger that day, we send no notifications.

There are some others considerations we take at the moment of checking the trigger. As mentioned, our first priority is selecting those users that navigated in the last hour. However, if there is no such interaction, we go for selecting the **most common navigation hour in the last week**. Meaning that if the user did not interact with the app in the last hour, we check if he interacted in the last week and, if so, what was the most frequent hour. If signal is found, then we send that user a notification at that hour of the day. However, if none of the two instances occur, then we simply drop those users, hoping that in the last hour we will get a better signal.

Figure 41 presents the process described before.

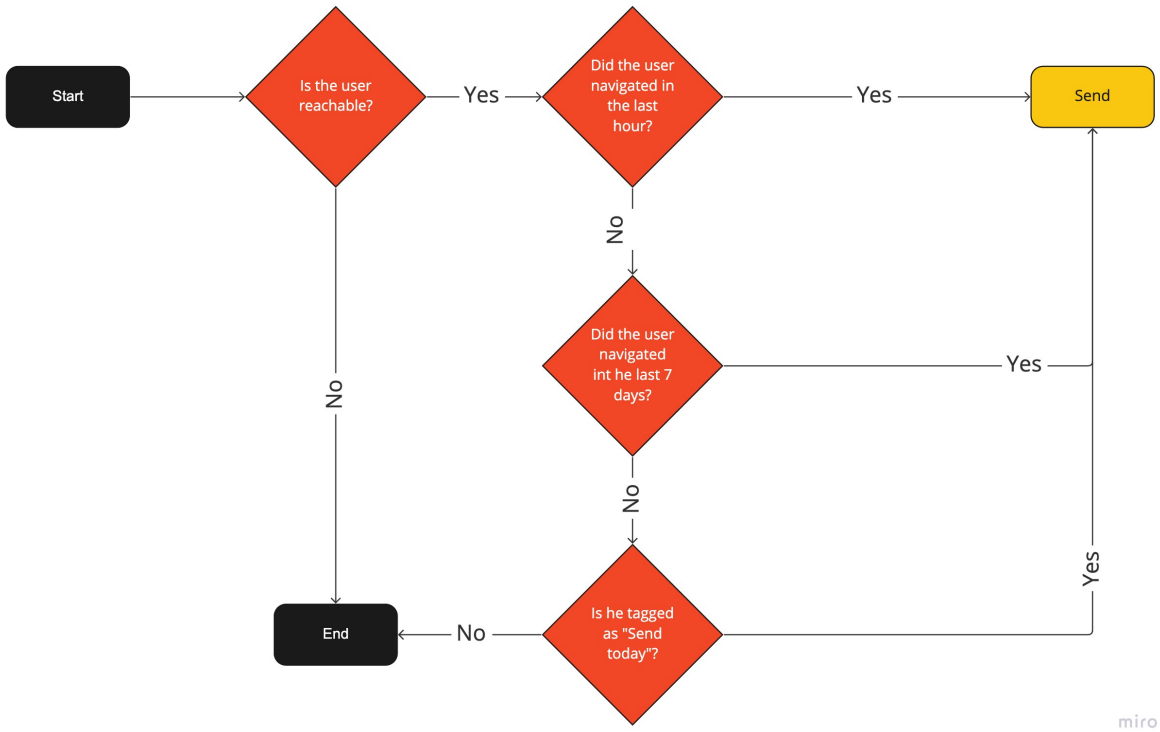


Figure 41. Communication process

4 Results

We let the experiments ran their course for a few months in order to gather significant data, given the fact that the habit process takes 30 days, we let them on for at least three months.

When looking at the distribution for the first operation (that is the one that introduced them into the habit process pipeline) we obtained the results illustrated in figure 42.

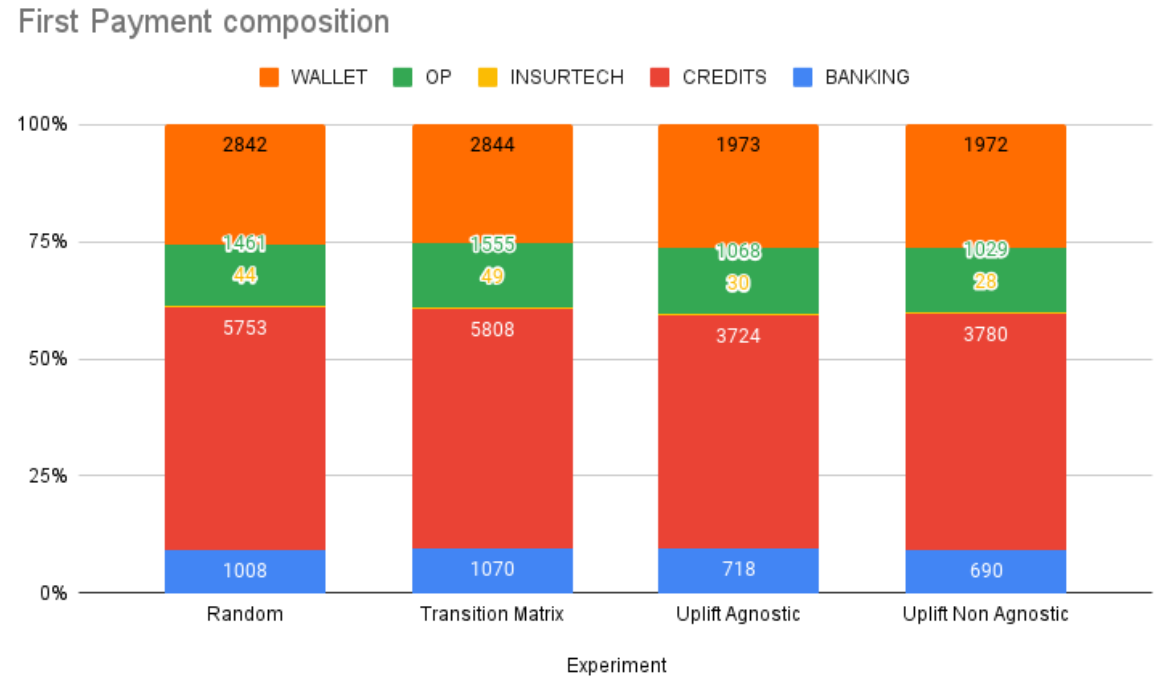


Figure 42. First operation distribution

What we notice is that the operations are evenly distributed amongst the experiments, which is something we were expecting to see, as otherwise it would benefit one over the others. This does not mean that the operations **inside** an experiment are evenly distributed. As it evident from figure 42, most of the users **begin** their habit journey by operating in the **Credit** category, followed by **Wallet** operations as paying services, transport, etc.

Moving along the timeline, if we take a look at **how the users are reaching the end of their 30 days journey** in figure 43, we start noticing some results. The figure illustrates at what number of operation the user got by the end of the 30 days process. We notice that some people never pass the first operation, indicating a first barrier we should take into account (more of it later).

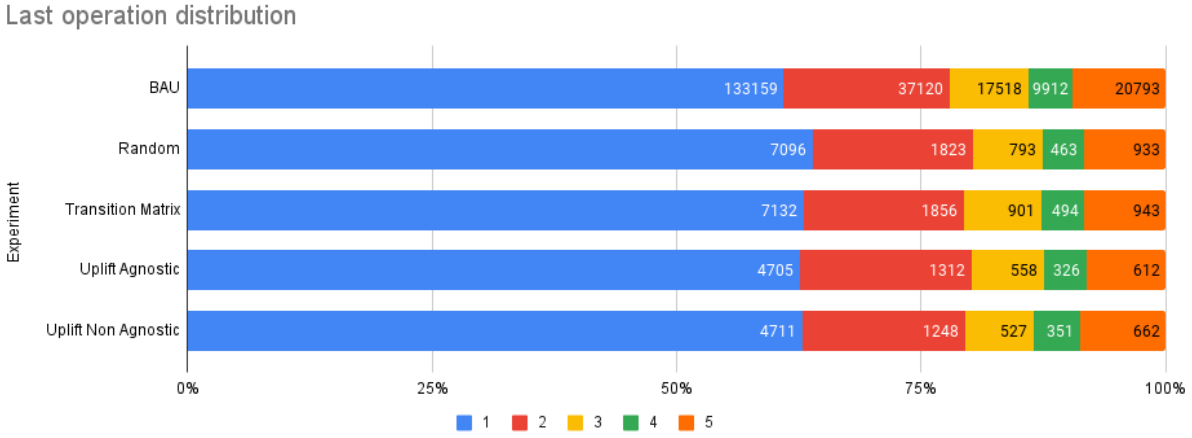


Figure 43. Last operation distribution

If we compare the four experiments, we notice that the number of users that got to the last operation day is very similar between the them, however, by looking at table 27 we see that the **Non Agnostic Uplift** model is achieving a little bit more users to get to the final operation, thus achieving the habit state. However, when compared to what is **currently** being done, represented by the **BAU** group (users who do not receive any of our notifications, but any other), it is evident that none of the experiments manage to get more users. Another possible reading is the number of users ending their 30 days process in the first operation day. Ideally, a good experiment should have more users in the more advanced stages and less in the first ones. Of course, the first operation day is always the one with the most number of users because of we saw in section 2.2.1 about the big percentage of people who only used the application once, only to never return again.

	Operation Day				
Experiment	1	2	3	4	5
BAU	60,94%	16,99%	8,02%	4,54%	9,52%
Random	63,88%	16,41%	7,14%	4,17%	8,40%
Transition Matrix	62,97%	16,39%	7,96%	4,36%	8,33%
Uplift Agnostic	62,62%	17,46%	7,43%	4,34%	8,15%
Uplift Non Agnostic	62,82%	16,64%	7,03%	4,68%	8,83%

Table 27. Last operation distribution

However, if we take a look at the composition of the first operation columns, we notice that all the experiments are managing to "move" the same amount of users from the first operation to the next one. However, just like it was for the final operation day, none of them is better at that than what is currently being done.

Although this preliminary results do not seem favorable for any of the models, we checked the core business metrics that would tell us how good our experiments actually were.

Experiment	**Open Rate**	**Conversion Rate**	**Lift**
BAU	8,77%	2,53%	0,70%
Random	6,03%	0,55%	0,10%
Transition Matrix	5,70%	1,84%	0,10%
Uplift Agnostic	5,18%	1,75%	0,60%
Uplift Non Agnostic	5,82%	1,93%	0,60%

Table 28. Experiment results - Business Metrics

Table 28 introduces new metrics that are not related to the technical performance of the models like the AUUC for the uplift estimators. Instead, because we are interested in the model working in the **real world**, we want to see if our business KPIs are being affected by them. In the day to day, we define the business situation by the following metrics:

- **Open rate**: The ratio of notifications opened over the number sent. It indicates, in a way, how successful at being noticed our campaigns are.
- **Conversion rate**: The ratio of users who operated in the category sent to them over the number of notifications sent.

- **Lift**: Measured in percentage points, it is the difference between the conversion rate of the "treated" group and the "control" group. It is one of our most important metrics as it really determines whether the experiments are working or not. It measures the incremental users the model can bring, users that otherwise would not have converted.

It is worth mentioning that all the results presented in this section are checked through a significance process. Conceptually, we did a **two sample Z test** of proportions to determine whether the difference between two groups is significant. Every number has been validated in order to make the right conclusions with data that is statistically significant.

Like in the other cases, the usual business operation (BAU) seems to excel at almost every metric of interest. Still, although the open rate and conversion rate are low for the experiments, when we see the way the conversion rate drops harder for the BAU and random experiments, it is clear that, at some point, the campaigns sent by the transition matrix and uplift experiments are a little better at selecting users and choosing the right suggestions. Another important observation is that the random experiment has a very low conversion rate, meaning that although it has the second highest open rate, the quality of the users it is targeting is not as good. This is not because the users are bad themselves, but probably the operations are not good enough for them.

It is evident that both the Random and Transition matrix experiments are not very good at generating an incremental difference, this could be because the operations they are sending is either not working for the users, or maybe it does not generate an uplift effect, because they still would have operated without any suggesting model. On the other hand, if we are only considering the experiments developed in this work, we would say that the Uplift models are clearly superior to the rest, proving that choosing the operation through a causal model that can determine what is important to the user and what is not is a better approach than simply following "the normal behaviour".

Although the BAU is generally better than the rest of the experiments, up next we present the same metrics, for every experiment, opened for each operation, as we try to identify if there are some of them at which any of the experiments is better.

Operation	Open Rate - BAU	Open Rate - Random	Open Rate - Transition Matrix	Open Rate - Agnostic	Open Rate - Non Agnostic
Account Funding	-	5,79%	6,37%	6,28%	5,82%
Antenna TV	-	3,47%	2,07%	-	-
Cards	10,22%	-	-	-	-
Credits	13,06%	11,98%	7,26%	10,96%	6,55%
Cripto	-	4,98%	6,25%	5,35%	5,70%
Digital Goods	-	4,49%	4,67%	-	-
QR	8,07%	4,91%	4,74%	3,55%	4,67%
Recharge	5,89%	5,49%	4,57%	4,62%	5,60%
Send Money	-	5,62%	5,60%	4,24%	4,64%
Transport	4,83%	4,20%	4,20%	3,14%	2,45%
Utilities	7,66%	5,44%	5,31%	5,09%	6,03%

Table 29. Open rate by experiment

Operation	Conversion Rate - BAU	Conversion Rate - Random	Conversion Rate - Transition Matrix	Conversion Rate - Agnostic	Conversion Rate - Non Agnostic
Account Funding	-	0,20%	1,20%	1,08%	0,57%
Antenna TV	-	0,05%	0,00%	-	-
Cards	0,00%	-	-	-	-
Credits	1,28%	0,53%	0,86%	0,73%	1,09%
Cripto	-	0,90%	2,44%	1,26%	1,18%
Digital Goods	-	0,03%	0,00%	-	-
QR	0,59%	0,25%	1,72%	0,86%	1,14%
Recharge	4,99%	1,53%	1,98%	1,87%	3,80%
Send Money	-	1,19%	2,26%	5,52%	5,40%
Transport	3,22%	0,84%	1,64%	0,39%	2,34%
Utilities	2,99%	0,80%	1,29%	1,72%	1,54%

Table 30. Conversion Rate by experiment

Operation	Lift - BAU	Lift - Random	Lift - Transition Matrix	Lift - Agnostic	Lift - Non Agnostic
Account Funding	-	0,00%	-0,49%	0,31%	0,12%
Antenna TV	-	0,05%	-5,74%	-	-
Cards	0,00%	-	-	-	-
Credits	0,57%	0,40%	0,57%	0,73%	0,50%
Cripto	-	0,08%	-0,27%	0,57%	-0,07%
Digital Goods	-	0,03%	-3,56%	-	-
QR	0,07%	0,12%	-0,23%	0,05%	0,77%
Recharge	1,55%	0,16%	0,66%	0,45%	0,87%
Send Money	-	-0,50%	-0,47%	2,35%	3,20%
Transport	-0,75%	0,84%	-2,21%	-3,80%	2,07%
Utilities	0,38%	0,52%	0,02%	0,01%	0,52%

Table 31. Lift by experiment

Not all the operations are present in all the experiments, and the reasons were explained throughout the work. For the BAU, which are campaigns that we cannot control, evidently not all the operations are equally relevant, hence the lack of some of them. For cards, we explained that because of the multiple segments and rules that we had to apply, and that were out of our control and understanding, in order to determine what class of notification should be sent, we chose not to send it for this version of the experiment. Finally, both Antenna and Digital Goods categories had a very low signal in the uplift models, meaning that the information we had about their conversions was too low for them to estimate correctly, reason why they were excluded.

In general, we see that in terms of Lift, for those operations where the BAU does not have campaigns, either Agnostic and Non Agnostic present better performance, with **Send money** and **Transport** being the two best operations with more than 2% of Lift. Besides, the conversion rate in both operations in Non Agnostic experiment were among the top of the lot, indicating that the model is correctly suggesting them.

On the other hand, **Cards** operations perform poorly in general, even though we did not train our experiments with it, even in the usual business pipeline it is not bringing new incremental users.

An important "metric" we wish to control is the **frequency** of the notifications. In other words, the number of notifications we send to the user in the course of his 30 days. In figure 44 we see that, to our surprise, all of our experiment presented a frequency of 6 to 5 push notifications in 30 days, meaning an average of one notification a week. This made sense to us because with that frequency, the user could get to the habit state in time, providing that each notification ended up with a conversion. However, when looking at the BAU frequency, we see 3 messages as the average. What is surprising is that given the fact that this last one generally outperformed our models, we would expect it to send much more messages than that. Evidently, some aspect of the current processes are generating a big impression on the users in such few interactions.

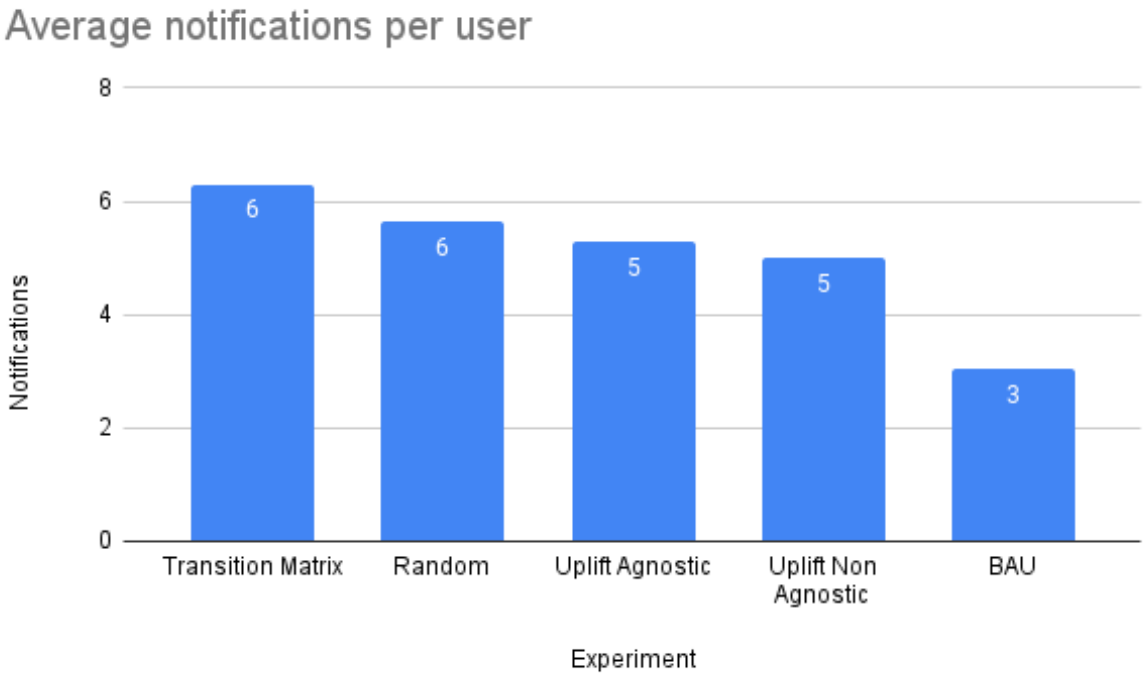


Figure 44. Frequency of notifications

Finally, we wanted to test how well our Uplift model was working. This was because the rest of the experiments were simple in their operation. Instead, the S-Learner, being a machine learning model, needed a deeper study to determine if it was correctly identifying the operations to be suggested, and if the established threshold were adequate.

For such study, we evaluated the CATE distributions that the model returned when predicting, for each operation day, splitting the analysis into those who converted and those who did not. Over that distribution, we draw a line where the threshold is located and from there studied how well was the learner identifying both groups. The result was a plot like figure 45. The rest of the plots are presented in Apendix A.

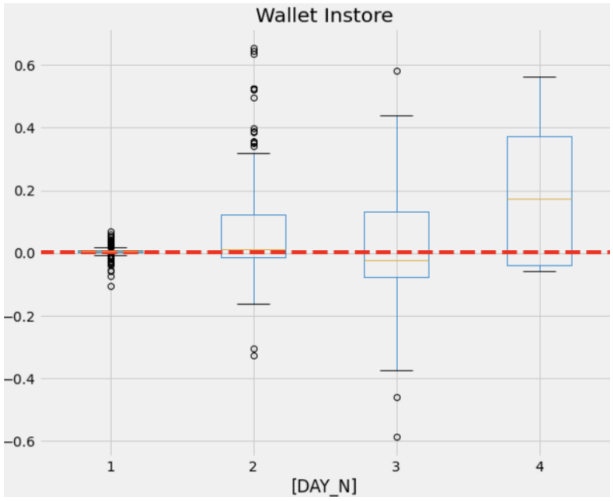


Figure 45. Threshold analysis example

From studying the plots we concluded that the S-Learner model was correctly identify-

ing users who converted and those who did not because the threshold was placed correctly for each operation available. This means that the the CATE of users who converted are located mostly on the upper part of the threshold, and those who did not convert are place under it. This was especially true for the operation days 2,3 and 4, leaving the first operation day in a much more gray area. In other words, although the model is apparently separating correctly, this is not true for the first operation day where the model encounters a certain level of difficulty.

In figure 46 we also studied the number of operations we predict, in average, for the same user, for each experiment. This was done with the idea of understating whether the experiments were locked in sending always the same operation, regardless of the user's response, or if they tried different options in order to obtain a conversion.

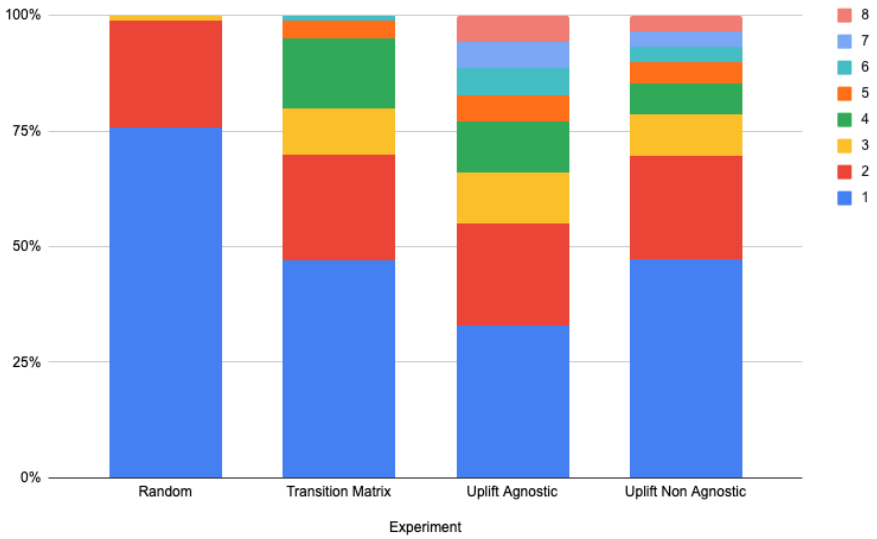


Figure 46. Average number of flows predicted for a user

As expected, the Random experiment has a bigger proportion of distinct operations predicted (the ones are majority in the figure). This is normal as it assigned the operations randomly, for what one would assume that it would give each one a chance.

The Transition Matrix experiment, although it still shows a tendency to predict the same operation once or twice, starts to show some cases of three or four repetitions of the same prediction.

On the other extreme we have the Uplift models showing that for up to 33% of the population they repeat the predicted operations.

This analysis shows that the more intelligent the model, the more it sticks to the same prediction.

5 Conclusions

5.1 Limitations

Before detailing some of the conclusions we extracted from this project, I believe it is important to first remark and remind the reader the limitations this project had at the time of production and the ones that presented long after.

Firstly, it is important to remark that this study was a first approach to the problem of Habit generation, meaning that it was in no way a final experiment. On the contrary, we hope to build better solutions from the one we presented here. On the other hand, it is important to note that this work was part of a company project, with all that that implies: dealing with agendas, not being able to take some decisions or approaching a problem in a different way that one, individually would and, the biggest downside to it all was the fact that the development time was restricted, we had due dates to fulfill and for that, sometimes we had to compromise. For example, we would have wanted to do a parameter optimization for the uplift models but, because of time limitations, we developed a model that produced a good enough output, knowing that there would be time for improvement in the future.

Secondly, the movements table was incomplete because the Data Engineering team did not include some operations that were relevant for the definition of a **Habit user**. Because of that, we cannot blindly trust in the results obtained for any of the models, as we would never know if a user is or is not in the habit process. This seriously affected our work because we were never sure about the validity of the analysis or the results. Not only that, but it prevented us from further improving the first versions of the solutions since any change we did would still be based on incomplete data, thus making any attempt at it futile.

5.2 Learnings

From the results presented before, I can elaborate some conclusions and next steps that I hope can help to improve in future iterations of these project.

Having said that, we can draw some interesting points that can work as the bases for the future.

This experience helped us reinforce the idea of the importance of data. Sometimes we underestimate the databases, believing that just because data is in abundance it does not matter if it is a little "dirty". That idea could not be any more wrong; good data prevails over a good model.

Being part of a Marketing experiment, the definition of a good **control group** is of the upmost importance. While the development of this project, we faced many problems of corrupted bases, influenced groups and the creation of a clean, pure group of users that were not affected by our campaigns was a difficult task, but without it, we would notice confusing results, incoherent metrics that would throw us towards the wrong conclusions.

As for the experiments, considering the previously mentioned incomplete base, we conclude that although the results were not in favor of the models, the problem of getting the users to do five operations in thirty days can be modelled, meaning that an algorithm, whether it is ruled based or a complex machine learning model, can figure out a pattern that will allow it to determine the next best suggestion for the users. The results proved that in general, when comparing the models between themselves, an Uplift approach provided better results (although it was not that much big of a difference, it was significant). Such insight tell us that we were on the right track and future work should keep working on improving such approach. The line of work in this case is of course related with a recommender system approach. Given the nature of the problem, how personal and specific to each person it is, the solution should be on pair with it. It is indeed a complex problem that should not be taken lightly. Many aspects of it were surely left out because of the way we decided to begin attacking this problem.

For starters, all of our variables were related to either transactional or navigational information of the users. Such data is not wrong or useless but, they were focused on operations in the ecosystem, **literal** operations. Actions like clicks to certain adds, links, filters applied to queries, text explicitly searched for in the app, etc. Many interactions were not taken into consideration because of time or complexity.

Man is a complex being and cannot me modelled at all, however, the decision process is something that we can and must aspire to understand if we want to establish continuity with the app. We are constantly making decisions: what to wear, what transportation to take to work, what to eat, etc. And when we take them to their simplest aspect, we comprehend that there are no random events but only consequences. Each decision made can be described as a product of past actions and decisions, a map that allows us to retrace our steps to understand how we got to where we are.

This approach, translated into the world of data, calls us to discover that map for each user: **what has the user done that has brought him to where he is today?** If we could put that path back together, its next step would cease to be a mystery and would become a problem of probability.

5.3 Further Work

As mentioned before, this project is far from finished, in fact, it should be considered as a stepping stone, from where to start building new and better solutions.

As a first idea, I would say that the data should be gathered and wrangled the "consequential" way of seeing the problem introduced before. Our first approach was tabular, too structured and, in a way, "simple". But we should find a way to keep track of all what the user does, study the events that interest us and retrace back a few steps and ask: **What is it that got you here? Is it the same for all the other users? Can we find a pattern in the history?** To answer such questions we require a different infrastructure, a different way of processing data, closer to the user and as real time as possible. Along with the data, an adequate solution should be tested. Of course I can only propose as if the answer was so simple the results would have been conclusive: LSTM, DQN, Graph based models, Reinforcement Learning models are only a few of the proposed solutions for a next version of the models. They all share the capacity of learning from **previous states** and keep doing so dynamically, something that a "classical" solution could not do by itself.

Still, even the models developed in the course of this work can be improved substantially. For starters, a hyper-parameters tuning was not done as we did not have the time for it. Touching and modifying the XGBoost trees in the learners could produce better results.

The results showed that the first steps of the process are the hardest for the models, and in there we found a great opportunity to improve. Separating the steps is a viable option, as each operation is a new state for the users.

Sometimes, as Data Scientist we focus so much on developing a perfect model that can surpass whatever any other team is doing for a specific problem. However, part of the job is to learn to recognize when something is being done well, especially if it is not done by us. Those are the moments when we have the chance to learn and overcome ourselves. For that reason, the BAU operations should be looked into as they are clearly performing very well. What audience are they targeting? What is their frequency? Do they use discounts? Do they randomly target users? What messages are they using? Learning

from the way it works could push the solutions proposed in this work further into a better path.

References

[1] . *2019 Product Benchmarks Report*. Sept. 7, 2021. URL: https://mixpanel.com/content/product-benchmarks/2019-report/ (visited on 04/11/2022).

[2] Charu Aggarwal. *Recommender Systems: The Textbook*. Spanish. 2016 ed. Springer, 2016. DOI: 10.1007/978-3-319-29659-3.

[3] *Anomaly Detection*. Nov. 15, 2021. URL: https://www.goalprofit.com/resources/tpost/1ptttuulf1-anomaly-detection (visited on 02/05/2022).

[4] Banco Interamericano de Desarrollo. *Innovations you may not know were from Latin American and the Caribbean*. May 2017. DOI: 10.18235/0000703. URL: https://publications.iadb.org/publications/english/document/FINTECH--Innovations-You-May-Not-Know-were-from-Latin-America-and-the-Caribbean.pdf.

[5] Instappy Blogger. *Instappy - App User Lifecycle: The Five Stages of App Growth*. Oct. 6, 2016. URL: http://www.instappy.com/blog/app-user-lifecycle-the-five-stages-of-app-growth/#:%7E:text=We've%20identified%20five%20key,some%20more%2C%20driving%20exponential%20growth. (visited on 02/21/2022).

[6] Huigang Chen et al. *CausalML: Python Package for Causal Machine Learning*. 2020. arXiv: 2002.11631 [cs.CY].

[7] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *CoRR* abs/1603.02754 (2016). arXiv: 1603.02754. URL: http://arxiv.org/abs/1603.02754.

[8] Chollet. *Machine learning: a new programming paradigm*. 2018.

[9] François Chollet. *Deep Learning with Python, Second Edition (English Edition)*. Spanish. Manning, 2021.

[10] Ermon. *Probabilistic Graphical Models*. Spanish. Jan. 4, 2022. URL: https://ermongroup.github.io/cs228-notes/preliminaries/introduction/.

[11] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Spanish. 2nd ed. O'Reilly Media, 2019.

[12] Carlos A. Gomez-Uribe and Neil Hunt. "The Netflix Recommender System: Algorithms, Business Value, and Innovation". In: *ACM Trans. Manage. Inf. Syst.* 6.4 (Dec. 2016). ISSN: 2158-656X. DOI: 10.1145/2843948. URL: https://doi.org/10.1145/2843948.

[13] Pierre Gutierrez and Jean-Yves Gérardy. "Causal Inference and Uplift Modelling: A Review of the Literature". In: *Proceedings of The 3rd International Conference on Predictive Applications and APIs*. Ed. by Claire Hardgrove et al. Vol. 67. Proceedings of Machine Learning Research. PMLR, Nov. 2017, pp. 1–13. URL: https://proceedings.mlr.press/v67/gutierrez17a.html.

[14] *How to Deal with Missing Data*. Feb. 9, 2021. URL: https://www.mastersindatascience.org/learning/how-to-deal-with-missing-data/ (visited on 05/26/2022).

[15] Humphreys. *10 Things to Know About Causal Inference – EGAP*. 2021. URL: https://egap.org/resource/10-things-to-know-about-causal-inference/ (visited on 05/25/2022).

[16] Tricon Infotech. *Improving Customer Engagement with Recommender Systems*. Dec. 9, 2021. URL: https://medium.com/@triconinfotech/improving-customer-engagement-with-recommender-systems-b423bdbb4e55 (visited on 01/26/2022).

[17] F.O. Isinkaye, Y.O. Folajimi, and B.A. Ojokoh. "Recommendation systems: Principles, methods and evaluation". In: *Egyptian Informatics Journal* 16.3 (2015), pp. 261–273. ISSN: 1110-8665. DOI: https://doi.org/10.1016/j.eij.2015.06.005. URL: https://www.sciencedirect.com/science/article/pii/S1110866515000341.

[18] Gareth James et al. *An Introduction to Statistical Learning: With Applications in R*. Spanish. 2nd 2021 ed. Springer, 2021.

[19] Pavel Kordík. *The value of personalized recommendations for your business*. Aug. 23, 2016. URL: https://medium.com/recombee-blog/the-value-of-personalized-recommendations-for-your-business-6b2e81ce0a4d#:%7E:text=Generated%20recommendations%20typically%20reduce%20the,users%20with%20your%20web%20services. (visited on 12/12/2021).

[20] Sören R. Künzel et al. "Metalearners for estimating heterogeneous treatment effects using machine learning". In: *Proceedings of the National Academy of Sciences* 116.10 (Feb. 2019), pp. 4156–4165. DOI: 10.1073/pnas.1804597116. URL: https://doi.org/10.1073%2Fpnas.1804597116.

[21] Valliappa Lakshmanan and Jordan Tigani. *Google Bigquery: The Definitive Guide: Data Warehousing, Analytics, and Machine Learning at Scale*. Spanish. O'Reilly Media, 2019.

[22] Chris Mahoney. *Reinforcement Learning: A Review of Historic, Modern, and Historic Developments ... — Chris Mahoney — Towards Data Science*. Jan. 6, 2022. URL: https://towardsdatascience.com/reinforcement-learning-fda8ff535bb6 (visited on 02/06/2022).

[23] Bernard Marr. *A Short History of Machine Learning – Every Manager Should Read*. Dec. 10, 2021. URL: https://www.forbes.com/sites/bernardmarr/2016/02/19/a-short-history-of-machine-learning-every-manager-should-read/?sh=3a586f1b15e7 (visited on 02/17/2022).

[24] Vishal Morde. *XGBoost Algorithm: Long May She Reign! - Towards Data Science*. Dec. 9, 2021. URL: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d#:%7E:text=What%20is%20XGBoost%3F,all%20other%20algorithms%20or%20frameworks. (visited on 05/28/2022).

[25] Xinkun Nie and Stefan Wager. *Quasi-Oracle Estimation of Heterogeneous Treatment Effects*. 2017. DOI: 10.48550/ARXIV.1712.04912. URL: https://arxiv.org/abs/1712.04912.

[26] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[27] Hossein Pishro-Nik. *Introduction to Probability, Statistics, and Random Processes (English Edition)*. Spanish. Kappa Research LLC, 2014.

[28] Quarrie. *Fintech Innovations*. 2020. URL: https://bootcamp.cvn.columbia.edu/blog/the-beginners-guide-to-fintech/.

[29] Nathaniel Quarrie. *The Beginner's Guide to Fintech*. Mar. 26, 2020. URL: https://bootcamp.cvn.columbia.edu/blog/the-beginners-guide-to-fintech/ (visited on 01/01/2022).

[30] James M. Robins. "Optimal Structural Nested Models for Optimal Sequential Decisions". In: *Proceedings of the Second Seattle Symposium in Biostatistics: Analysis of Correlated Data*. Ed. by D. Y. Lin and P. J. Heagerty. New York, NY: Springer New York, 2004, pp. 189–326. ISBN: 978-1-4419-9076-1. DOI: 10.1007/978-1-4419-9076-1_11. URL: https://doi.org/10.1007/978-1-4419-9076-1_11.

[31] Peter Robinson. "Root- N-Consistent Semiparametric Regression". In: *Econometrica* 56.4 (1988), pp. 931–54. URL: https://EconPapers.repec.org/RePEc:ecm:emetrp:v:56:y:1988:i:4:p:931-54.

[32] Baptiste Rocca. *Introduction to recommender systems - Towards Data Science*. June 2, 2019. URL: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada (visited on 12/12/2021).

[33] Joseph Rocca. *Introduction to Markov chains - Towards Data Science*. Dec. 8, 2021. URL: https://towardsdatascience.com/brief-introduction-to-markov-chains-2c8cab9c98ab (visited on 05/16/2022).

[34] Daniel Russo et al. "A Tutorial on Thompson Sampling". In: (2017). DOI: 10.48550/ARXIV.1707.02038. URL: https://arxiv.org/abs/1707.02038.

[35] Michał Sołtys. *Ensemble methods for uplift modeling*. Sept. 17, 2014. URL: https://link.springer.com/article/10.1007/s10618-014-0383-9?error=cookies_not_supported&code=722759e1-8f27-47a4-a97c-dedc10e380da (visited on 05/22/2022).

[36] Venkatesh Tata. *Simple Image Classification using Convolutional Neural Network — Deep Learning in python.* July 21, 2019. URL: https://becominghuman.ai/building-an-image-classifier-using-deep-learning-in-python-totally-from-a-beginners-perspective-be8dbaf22dd8 (visited on 02/05/2022).

[37] Mercado Libre Tech. *Why and when to build a Machine Learning Platform (part 2)*. Jan. 7, 2022. URL: https://medium.com/mercadolibre-tech/why-and-when-to-build-a-machine-learning-platform-part-2-66eae56c4b35 (visited on 05/21/2022).

[38] Testa. *10 Types of Treatment Effect You Should Know About – EGAP*. 2021. URL: https://egap.org/resource/10-types-of-treatment-effect-you-should-know-about/ (visited on 05/25/2022).

[39] Gellert Toth. *An Introduction to Clustering Techniques*. Dec. 22, 2020. URL: https://www.datasklr.com/segmentation-clustering/an-introduction-to-clustering-techniques (visited on 02/05/2022).

[40] Tripathi. *Underfitting and Overfitting in Machine Learning*. June 13, 2020. URL: https://datascience.foundation/sciencewhitepaper/underfitting-and-overfitting-in-machine-learning (visited on 05/25/2022).

[41] undefined. *Supervised vs Unsupervised vs Reinforcement Learning — Machine Learning Tutorial — Simplilearn*. Nov. 20, 2020. URL: https://www.youtube.com/watch?v=1FZ0A1QCMWc.

[42] Stephanie Walden. *What Is Fintech And How Does It Affect How I Bank?* Aug. 4, 2020. URL: https://www.forbes.com/advisor/banking/what-is-fintech/ (visited on 01/01/2022).

[43] *What is XGBoost?* URL: https://www.nvidia.com/en-us/glossary/data-science/xgboost/ (visited on 05/28/2022).

[44] Zhenyu Zhao and Totte Harinen. *Uplift Modeling for Multiple Treatments with Cost Optimization*. 2019. DOI: 10.48550/ARXIV.1908.05372. URL: https://arxiv.org/abs/1908.05372.

6 Apendix A

6.1 Uplift Curves

Up next we present the Uplift curves obtained during the training of the meta-learners. This curves were used for determine the adequate threshold that would indicate whether an operation was going to impact positively or negatively on a user.

6.1.1 Learner S

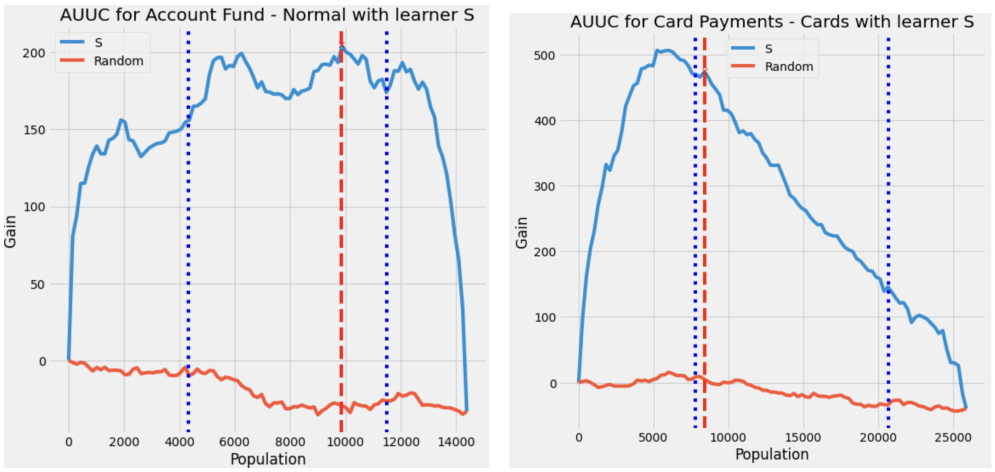


Figure 47. Learner S: Account Funding & Cards operations

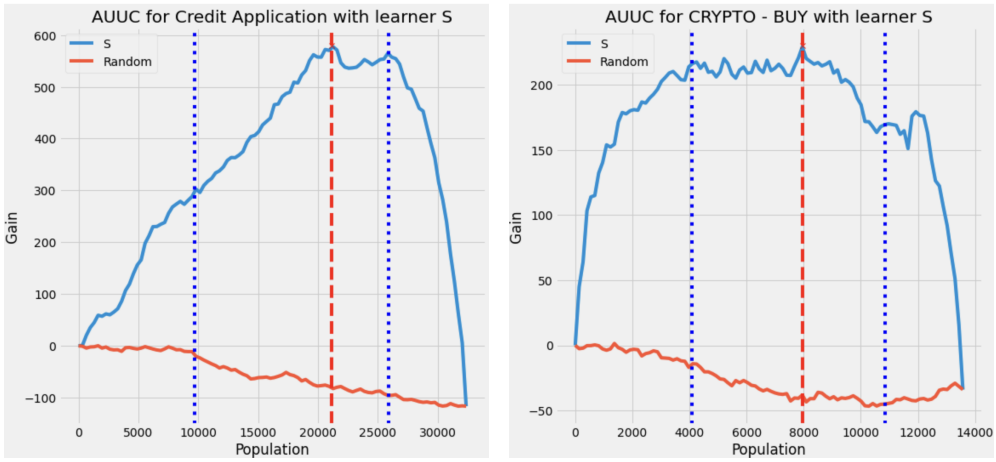


Figure 48. Learner S: Credit & Cripto operations

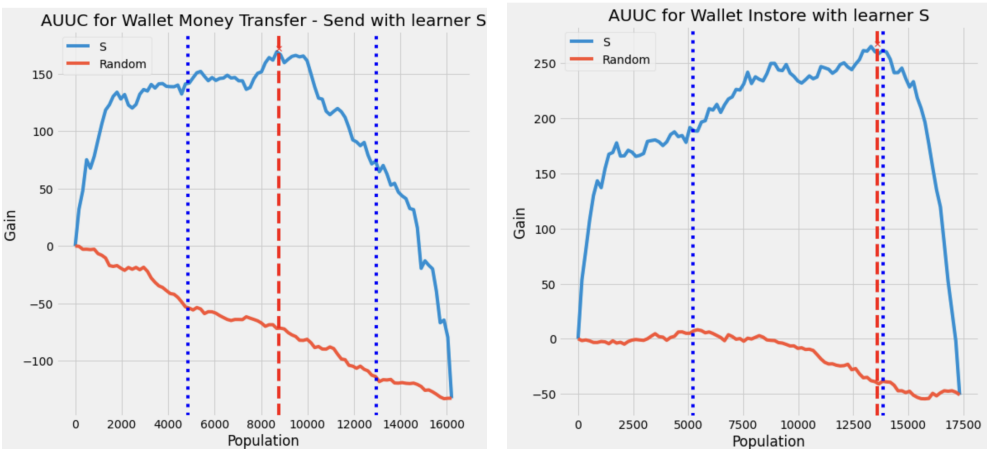


Figure 49. Learner S: Money Sending & QR operations

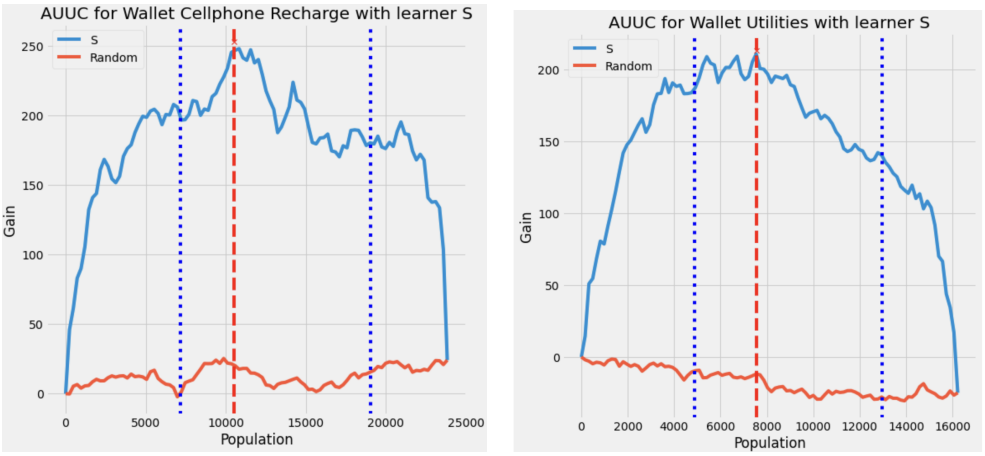


Figure 50. Learner S: Recharge & Utilities operations

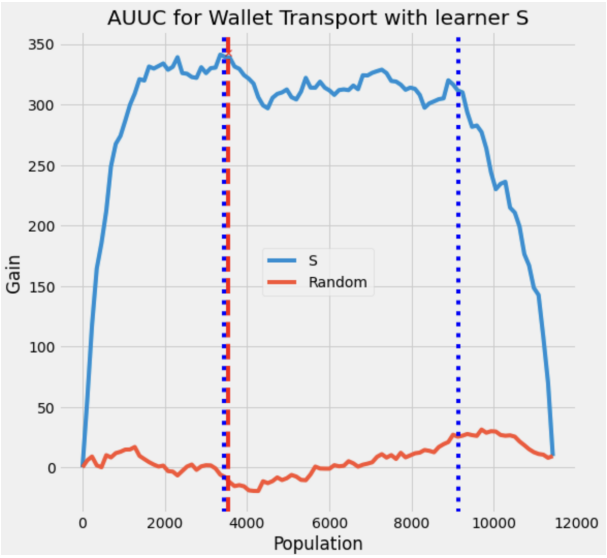


Figure 51. Learner S: Transport operation

6.1.2 Learner T

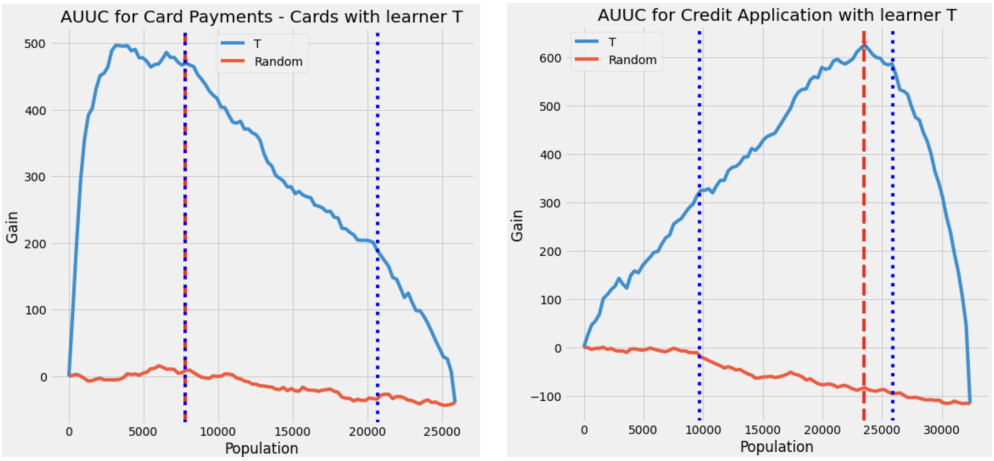


Figure 52. Learner T: Cards & Credit operations

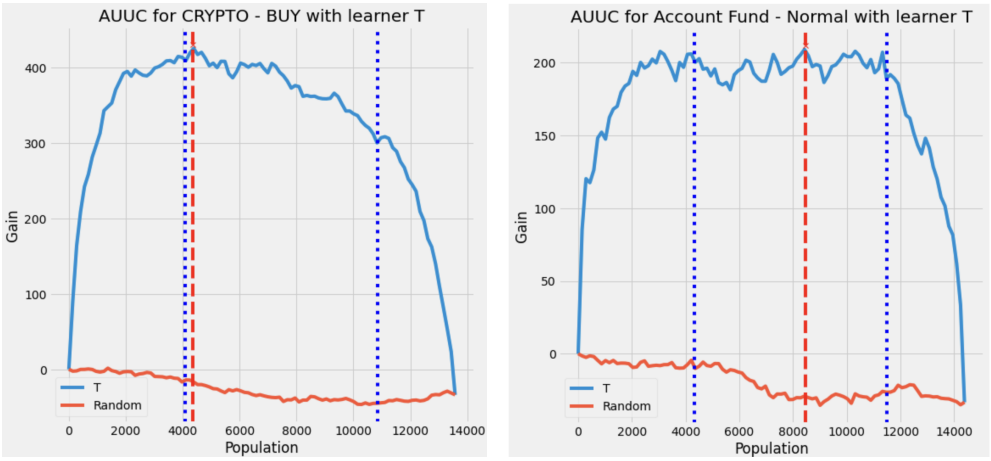


Figure 53. Learner T: Cripto & Account Funding operations

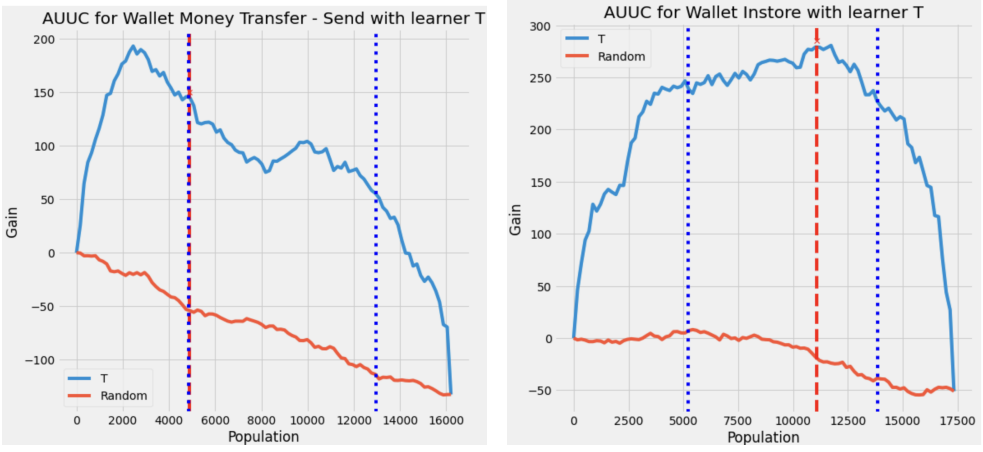


Figure 54. Learner T: Money Sending & QR operations

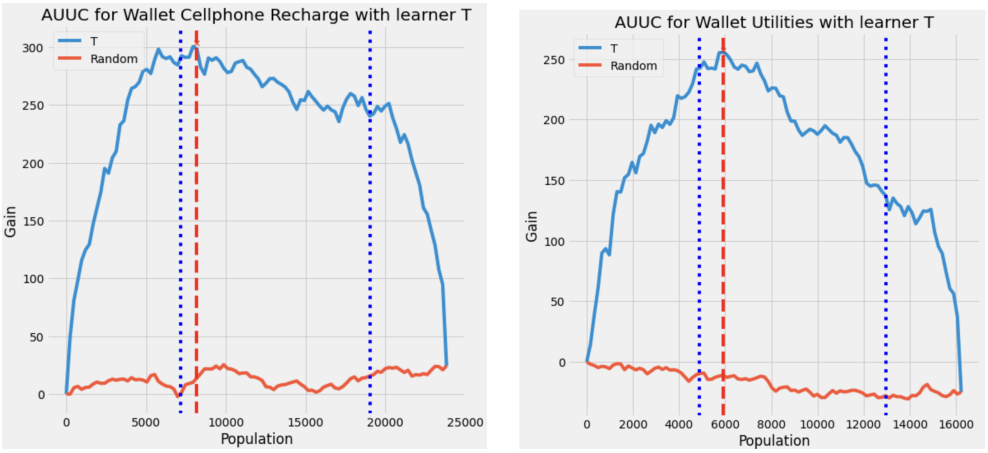


Figure 55. Learner T: Recharge & Utilities operations

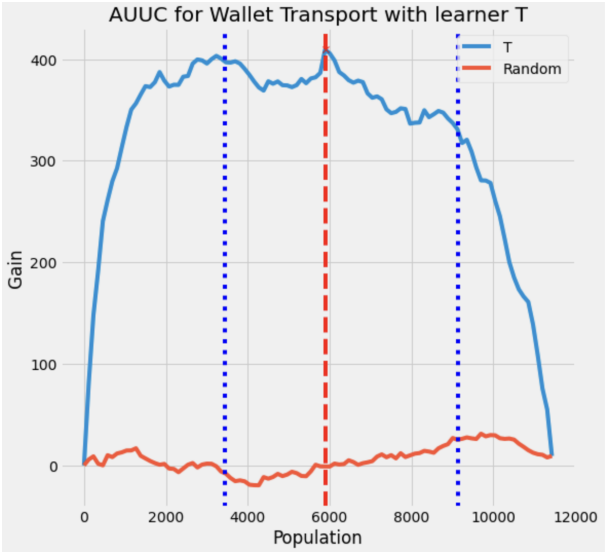


Figure 56. Learner T: Transport operation

6.1.3 Learner X

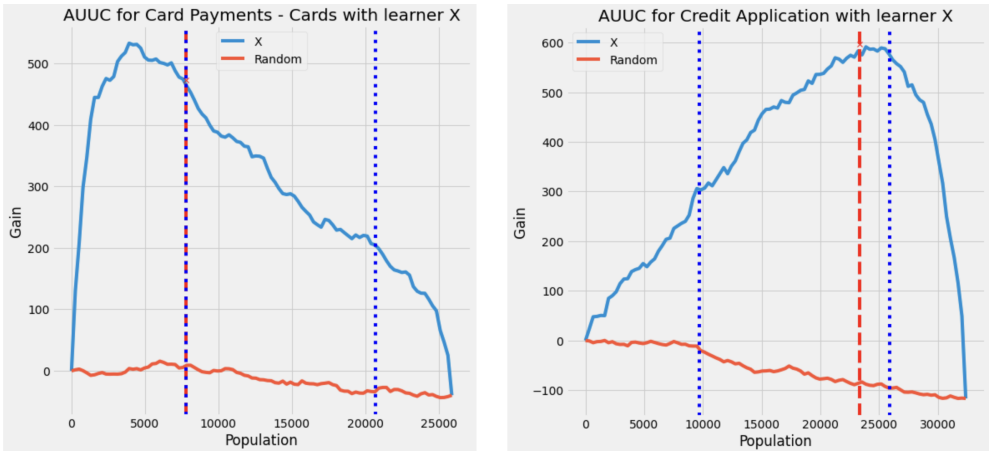


Figure 57. Learner X: Cards & Credit operations

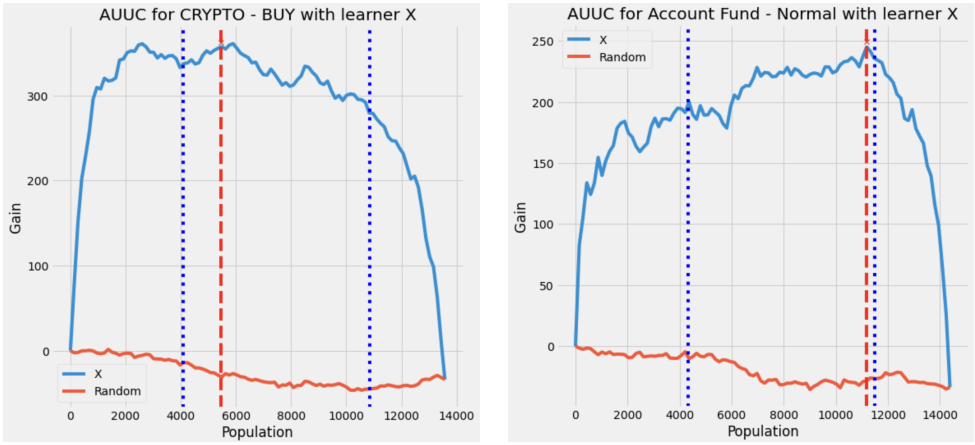


Figure 58. Learner X: Cripto & Account Funding operations

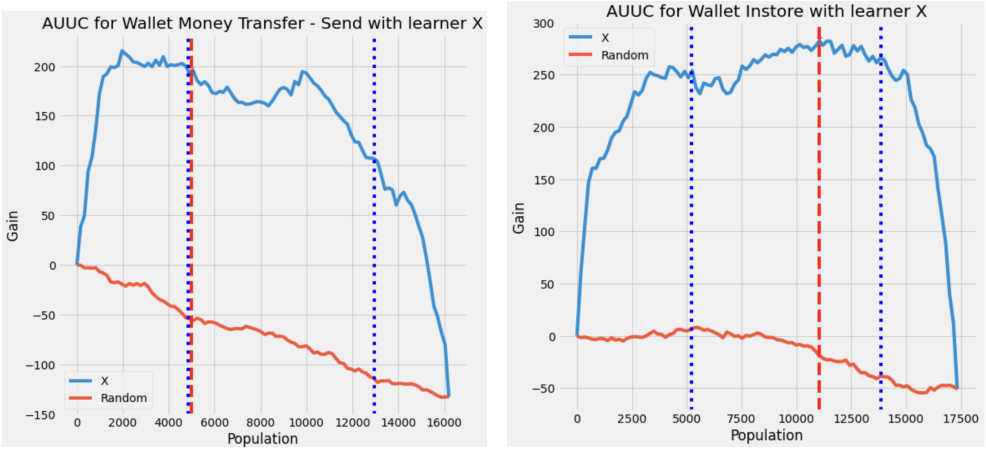


Figure 59. Learner X: Money Sending & QR operations

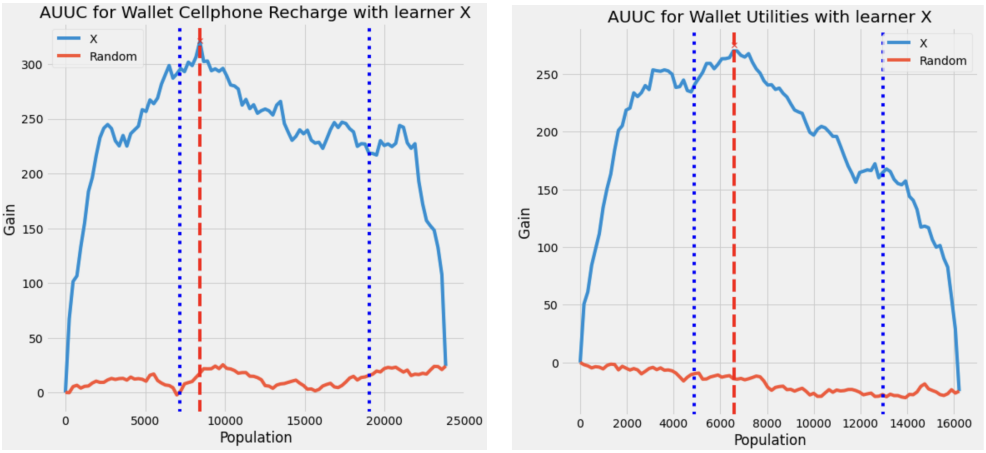


Figure 60. Learner X: Recharge & Utilities operations

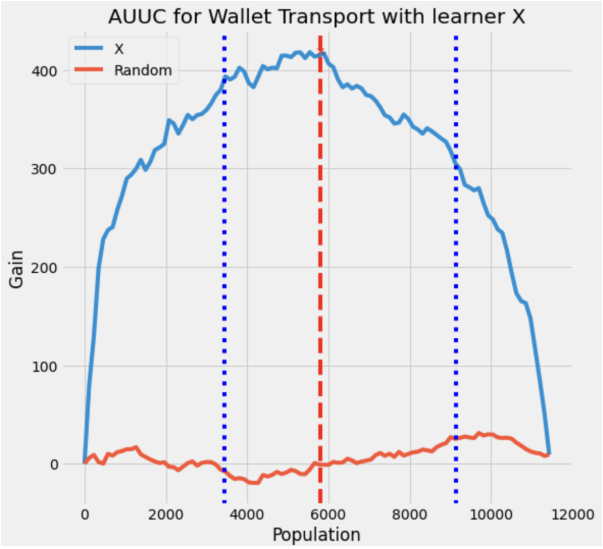


Figure 61. Learner X: Transport operation

6.1.4 Learner R

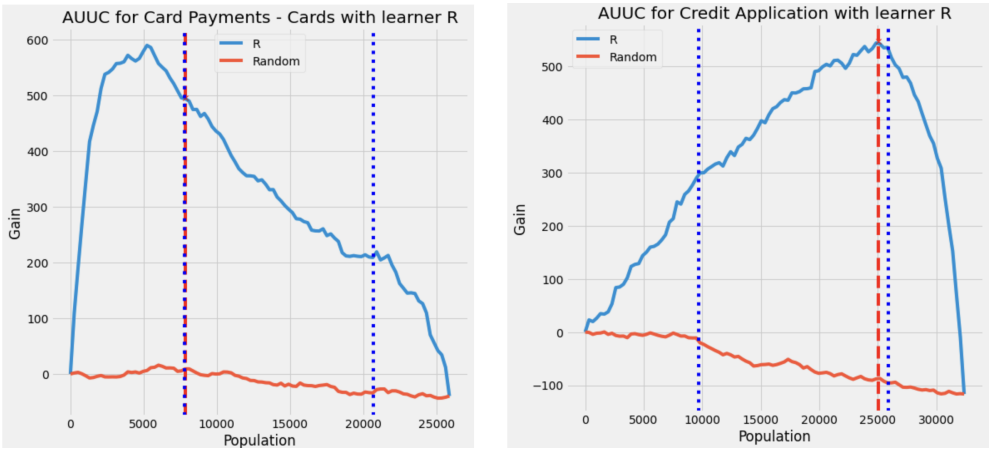


Figure 62. Learner R: Cards & Credit operations

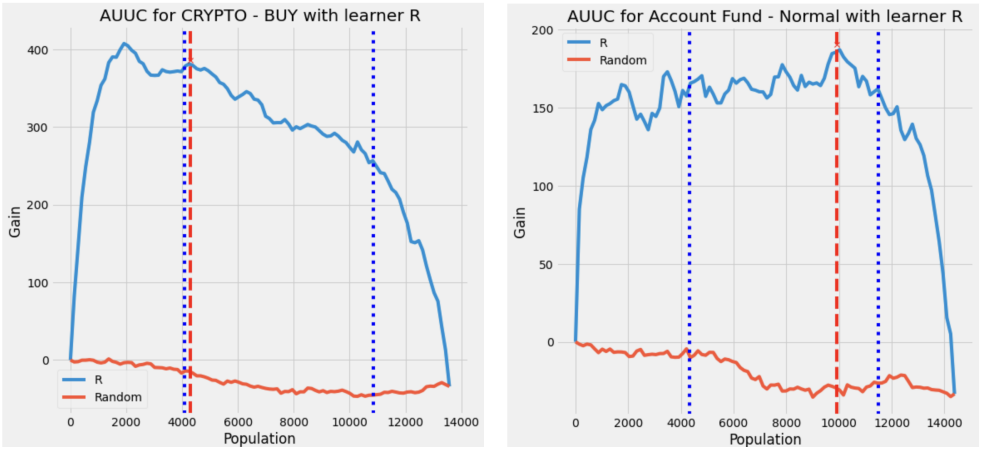


Figure 63. Learner R: Cripto & Account Funding operations

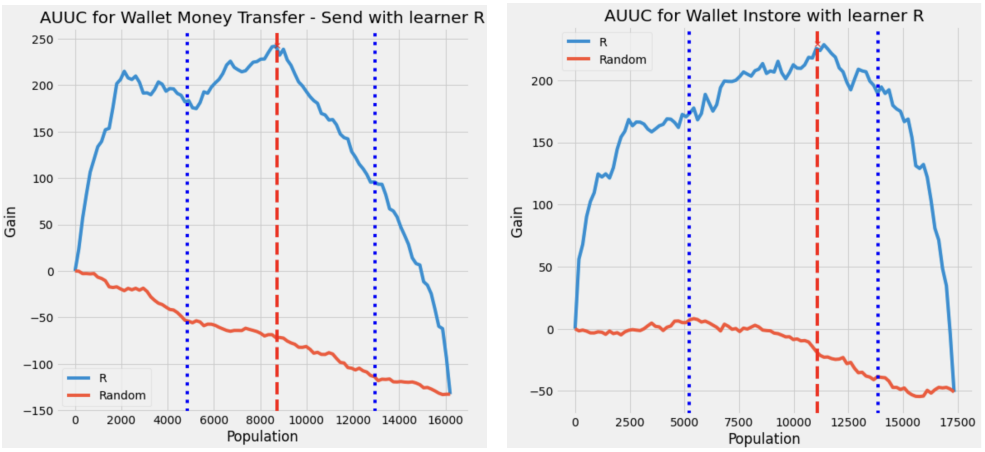


Figure 64. Learner R: Money Sending & QR operations

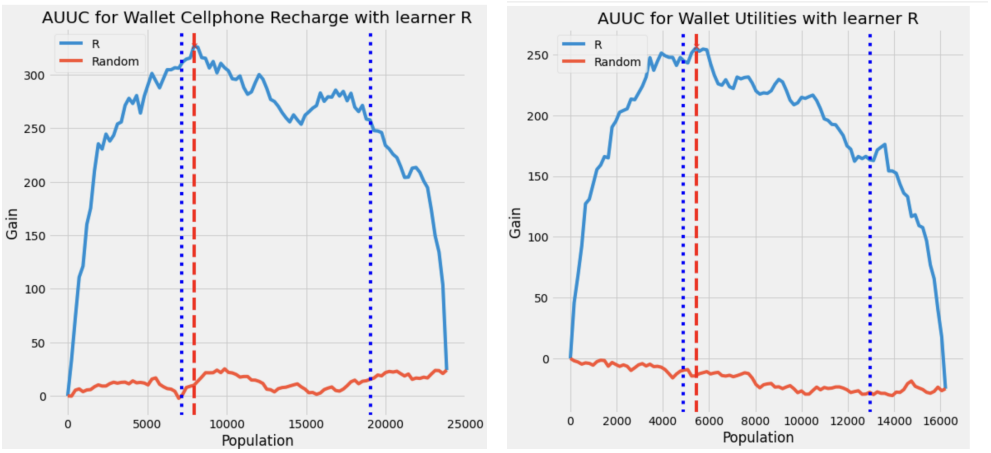


Figure 65. Learner R: Recharge & Utilities operations

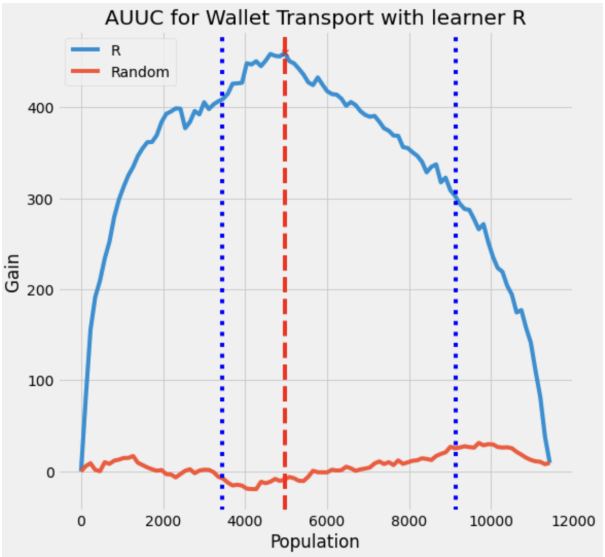


Figure 66. Learner R: Transport operation

6.2 Threshold Analysis

During the study of the solutions result, we performed an analysis on the threshold of the learner S, asking whether is was correctly placed or not. The following figures show the distribution of the CATE for each operation day for both the users that converted and those who did not. Ideally, a good threshold would separate both groups leaving the first ones in one side and the last on the other.

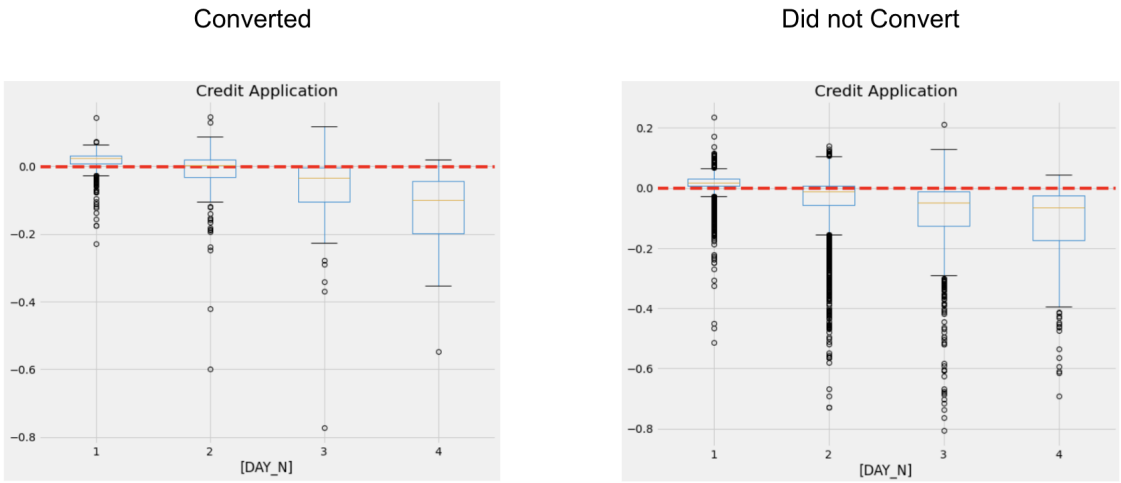


Figure 67. Credit operation

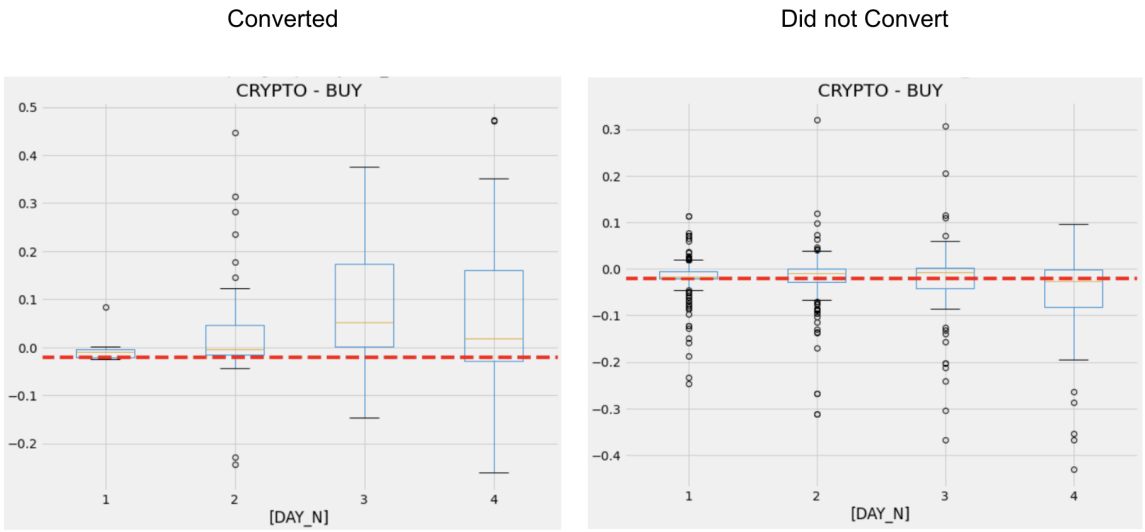


Figure 68. Cripto operation

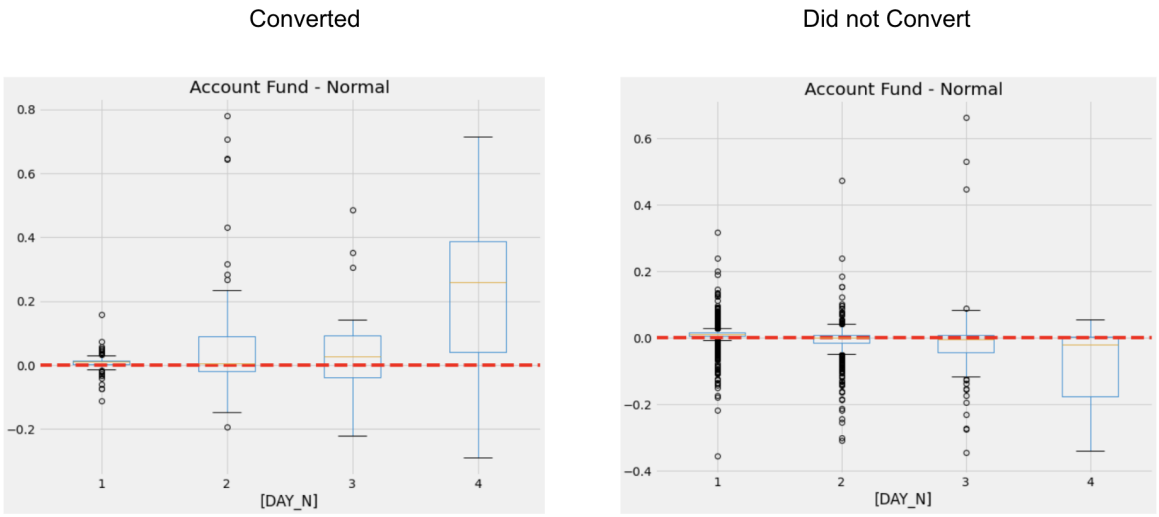


Figure 69. Account Funding operation

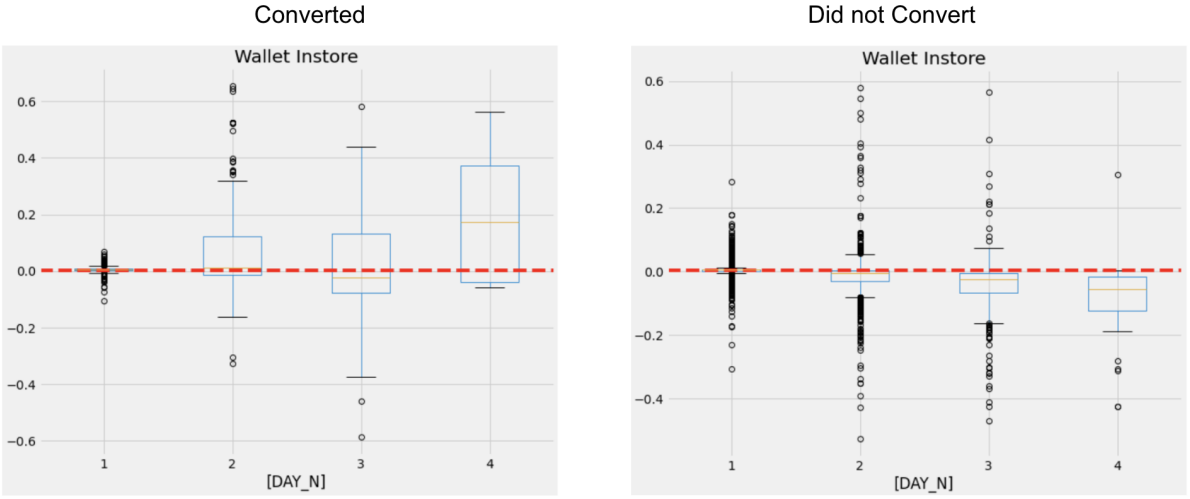


Figure 70. QR operation

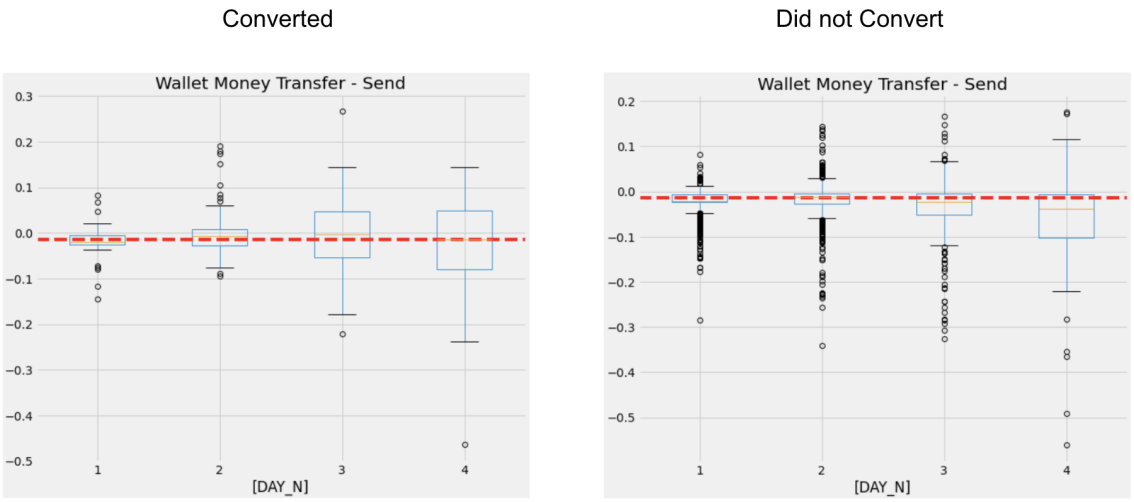


Figure 71. Transfer Money operation

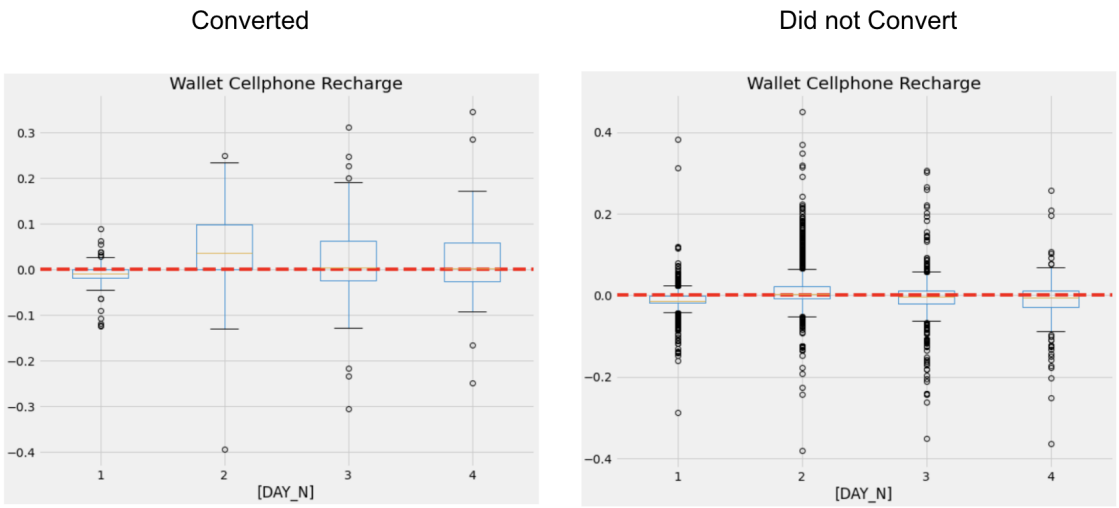


Figure 72. Recharge operation

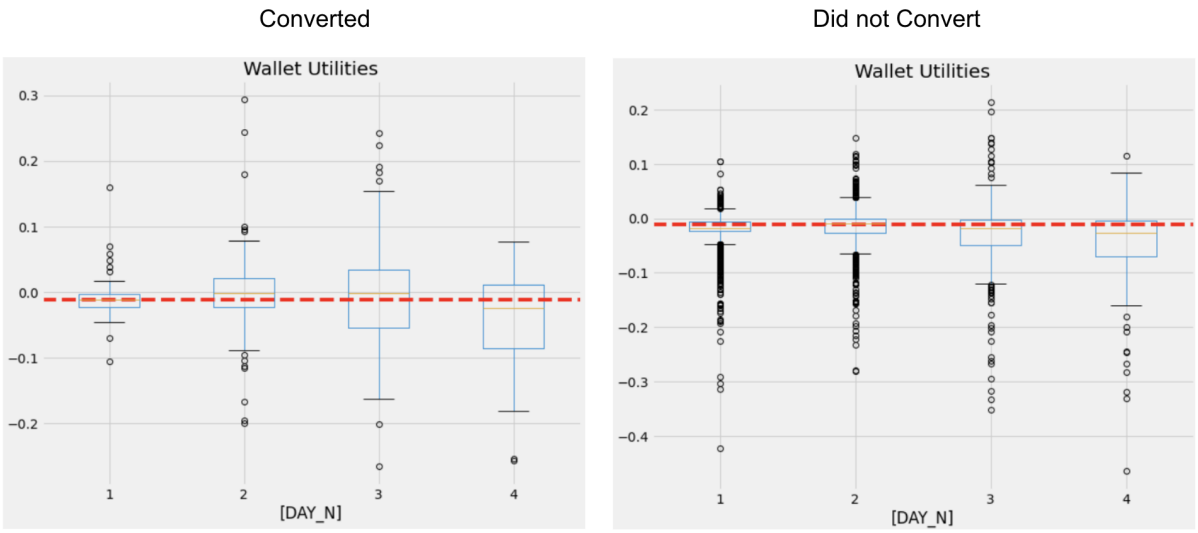


Figure 73. Utilities operation

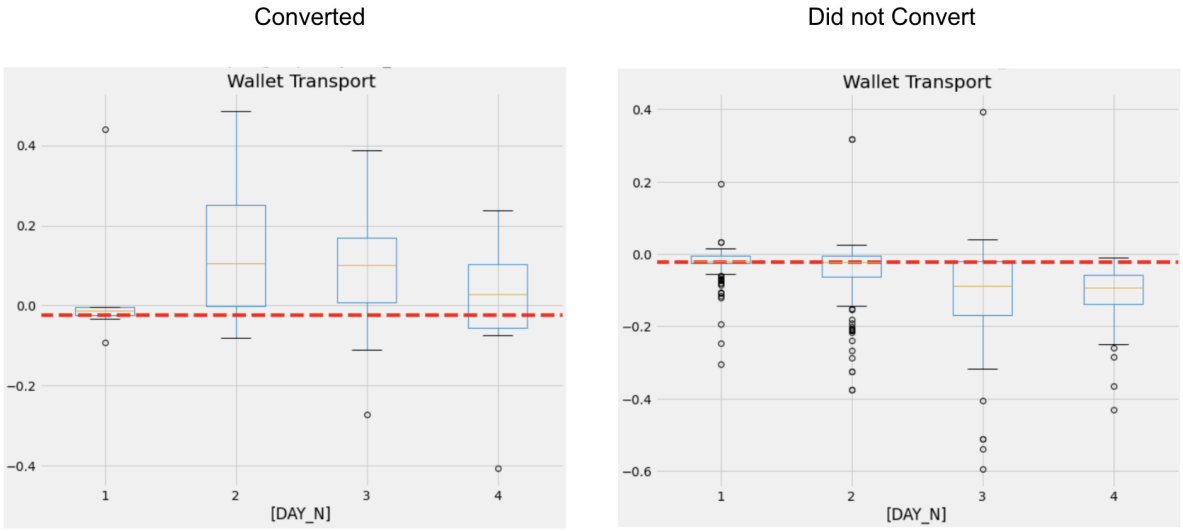


Figure 74. Transport operation