150 Successful Machine Learning Models: 6 Lessons Learned at Booking.com

Lucas Bernardi*
Booking.com
Amsterdam, Netherlands
lucas.bernardi@booking.com

Themis Mavridis*
Booking.com
Amsterdam, Netherlands
themistoklis.mavridis@booking.com

Pablo Estevez*
Booking.com
Amsterdam, Netherlands
pablo.estevez@booking.com

ABSTRACT

Booking.com is the world's largest online travel agent where millions of guests find their accommodation and millions of accommodation providers list their properties including hotels, apartments, bed and breakfasts, guest houses, and more. During the last years we have applied Machine Learning to improve the experience of our customers and our business. While most of the Machine Learning literature focuses on the algorithmic or mathematical aspects of the field, not much has been published about how Machine Learning can deliver meaningful impact in an industrial environment where commercial gains are paramount. We conducted an analysis on about 150 successful customer facing applications of Machine Learning, developed by dozens of teams in Booking.com, exposed to hundreds of millions of users worldwide and validated through rigorous Randomized Controlled Trials. Following the phases of a Machine Learning project we describe our approach, the many challenges we found, and the lessons we learned while scaling up such a complex technology across our organization. Our main conclusion is that an iterative, hypothesis driven process, integrated with other disciplines was fundamental to build 150 successful products enabled by Machine Learning.

CCS CONCEPTS

• **General and reference → Experimentation**; • **Computing methodologies → Machine learning**; **Model development and analysis**; • **Applied computing → Electronic commerce**; **Business IT alignment**;

KEYWORDS

Machine Learning; Data Science; Business Impact; Product Development; Experimentation; E-commerce

ACM Reference Format:
Lucas Bernardi, Themis Mavridis, and Pablo Estevez. 2019. 150 Successful Machine Learning Models: 6 Lessons Learned at Booking.com. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330744

*All authors contributed equally to this research.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08.
https://doi.org/10.1145/3292500.3330744

1 INTRODUCTION

Booking.com is the world's largest online travel agent where millions of guests find their accommodation and millions of accommodation providers list their properties, including hotels, apartments, bed and breakfasts, guest houses, etc. Our platform is developed by many different interdisciplinary teams working on different products, ranging from a large new application like our recent Booking Assistant, or an important page of the website with rich business value like the search results page, to a part of it, like the destinations recommendations displayed at the bottom. Teams have their own goals, and use different business metrics to quantify the value the product delivers and to test hypotheses, the core of our learning process. Several issues make our platform a unique challenge, we briefly describe them below:

High Stakes: Recommending the wrong movie, song, book, or product has relevant impact in the consumer experience. Nevertheless, in most cases there is a way to "undo" the selection: stop listening to a song or watching a movie, even returning a unsatisfactory product. But once you arrive to an accommodation that does not meet your expectations, there is no easy undo option, generating frustration and disengagement with the platform.

Infinitesimal Queries: Users searching for accommodations barely specify their destination, maybe the dates and number of guests. Providing satisfying shopping and accommodation experiences starting from this almost-zero-query scenario is one of the key challenges of our platform.

Complex Items: Booking an accommodation requires users to decide on several aspects like destination, dates, accommodation type, number of rooms, room types, refund policies, etc. These elements define a multi-dimensional space where bookable options are located, and since not all possible combinations exist, it is not trivial to navigate; users need help to find the best combination.

Constrained Supply: Accommodations have limited and dynamic availability. Its interaction with prices directly affects guest preferences and the behavior of accommodation providers. This aspect cannot be neglected when designing the shopping experience.

Continuous Cold Start: Guests are in a continuous cold start state [2]. Most people only travel once or twice every year. By the time they come back to our web site their preferences might have changed significantly; long in the past history of users is usually irrelevant. Furthermore, new accommodations and new accommodation types are added to the supply every day, their lack of reviews and general content, such as pictures and multilingual descriptions, make it difficult to give them visibility. Providing a personalized experience regardless of how often a guest interacts with Booking.com and being capable to find an audience for every

property from the very beginning of them joining Booking.com are difficult and important problems we face.

Content Overload: Accommodations have very rich content, e.g. descriptions, pictures, reviews and ratings. Destinations themselves also have rich content including visitor authored pictures, free text reviews, visitors endorsements and guides. This is a powerful advertising tool, but also very complex and difficult to be consumed by guests. Successfully exploiting such rich content by presenting it to users in an accessible and relevant way is another key challenge of our platform.

During the last years we have applied Machine Learning techniques to address these and other issues. We found that driving true business impact is amazingly hard, plus it is difficult to isolate and understand the connection between efforts on modeling and the observed impact. Similar issues have been highlighted by a few authors in the last years. Wagstaff position paper [13] mentions the lack of studies and lessons on how to exploit Machine Learning and achieve relevant impact in real world problems. In an example closer to our industry, Jannach et al. [5] explain how the field of recommender systems provides little guidance to impact metrics relevant to service providers, such as sales diversification, conversion rate or loyalty. Many other publications have described specific use cases of machine learning and their impact on business metrics (e.g. [10]) but no previous work to our knowledge has studied the overall process of developing and testing products to obtain business and user value through Machine Learning.

In this work we analyze 150 successful customer facing applications of Machine Learning techniques (plus many associated failures), and share the challenges we found, how we addressed some of them, lessons that we got along the way, and general recommendations. Our contributions are:

- A large scale study on the impact of Machine Learning in a commercial product, to our knowledge the first one in the field
- A collection of "lessons learned" covering all the phases of a machine learning project
- A set of techniques to address the challenges we found within each project phase

The rest of the paper is organized as a set of lessons associated to a specific phase of the development process of a Machine Learning project, namely Inception, Modeling, Deployment, Monitoring and Evaluation, and a final section where we present our conclusions.

2 INCEPTION: MACHINE LEARNING AS A SWISS KNIFE FOR PRODUCT DEVELOPMENT

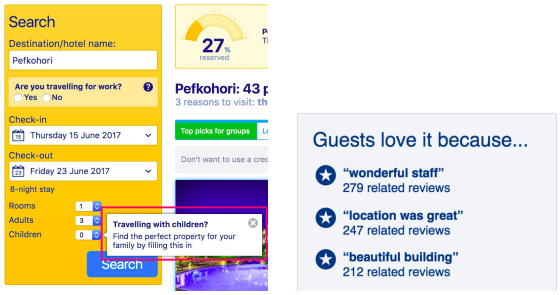
During the inception phase of a Machine Learning based project, a product team produces ideas, hypotheses, business cases, etc., where Machine Learning fits as part of the solution. One important lesson we have learned is that Machine Learning can be used for many and very different products in widely different contexts. In practice, our models are tools that help different teams improve their products and learn from their users. At one extreme, we create models which are very specific for a use case. For instance, they optimize the size of an specific element of the user interface, or provide recommendations tailored for one point on the funnel and one specific context. Because of their specificity, we can design and tune them to achieve good performance, hoping to create a strong business impact. The counterside is that their breadth of application is limited to a few use cases. At the opposite end of the spectrum we also create models which act as a meaningful *semantic layer*. They model understandable concepts, enabling everyone involved in product development to introduce new features, personalization, persuasion messages, etc., based on the output of the model. They could for instance indicate how flexible a user is with respect to the destination of their trip, giving product teams a concept of destination-flexibility that they can use to improve their products. Such models provide an interpretable signal, valid under all the contexts where the product teams would like to use them. This requirement limits the coupling between model prediction and specific target business metrics, but this is counteracted by the broad adoption such models have, generating often dozens of use cases all over the platform. Concretely, in our analysis we found that on average each semantic model generated twice as many use cases as the specialized ones.

2.1 Model Families

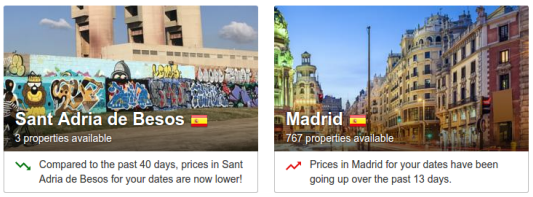
The following paragraphs explore different families of models currently deployed in our platform, focusing on how they can be used by product teams to influence our customers. This categorization works as a tool to generate ideas to exploit the capabilities of Machine Learning, forming the backbone of our strategy to address the issues described in the introduction.

2.1.1 Traveller Preference Models. Users display different levels of flexibility on different aspects, from no flexibility at all to complete indifference. We consider several trip aspects like destination, property price, property location, quality, dates, and facilities among others, and build several Machine Learning models that, in combination, construct a user preference profile assigning a flexibility level to each aspect. Models in this family work as a *semantic layer*. As an example the Dates Flexibility model gives a measure of how flexible a user is about traveling dates. If a user is flexible, dates recommendations might be relevant in some situations, but if the user is not flexible, date recommendations might turn out distracting and confusing, and are therefore not displayed. Another treatment could focus on inflexible users, re-enforcing the chosen dates with relevant information like price trends or availability.

2.1.2 Traveller Context Models. Travellers travel as couples, as families, with a group of friends or alone, either for leisure or for business. They might go by car to a close by city or by plane to the other side of the world, and visit one single city for a long stay or several cities one after the other for shorter periods. All of these are examples of what we call *Traveller Context*, which is a theme of the trip that defines constraints and requirements. Most of these contexts are not explicitly stated in our platform, and the ones that can be specified, are usually omitted by most users. Thus, predicting, or guessing the context of the current user, as early in the shopping experience as possible, is highly valuable. The Traveller Context Models also work as a semantic layer, in this case, enabling teams to create features for specific contexts. As an example consider the Family Traveller Model, that estimates how



(a) Traveller Context Model (b) Content Curation Model



(c) Content Augmentation Model

Figure 1: Examples of Application of Machine Learning

likely is that a user is shopping for a family trip. Usually, Family Travellers forget to fill in the number of children they travel with (see Figure 1(a)), going through a big part of the shopping process only to find out that the chosen property is out of availability for their children. The Family Traveller Model is used to remind the user to fill in the children information as early in the experience as possible, hopefully, removing frustration.

2.1.3 Item Space Navigation Models. Most users who browse our inventory navigate through several supplementary and complementary options and items, such as dates, properties, policies, locations, etc. In order to make a choice, they need to keep track of the options they have seen, while exploring neighbouring ones and trying to make a purchase decision. Our *item space navigation models* both feed from this process and try to guide it. They treat different actions, like scrolling, clicking, sorting, filtering etc., as implicit feedback about the user preferences. These signals can then be used to facilitate access to the most relevant items in the user history, as well as to surface other relevant items in our inventory.

2.1.4 User Interface Optimization Models. Font sizes, number of items in a list, background colors or images, the presence or absence of a visual element, etc., all have big impact in user behaviour as measured by business metrics. Models in this family directly optimize these parameters with respect to a specific target business metric. We found that it is hardly the case that one specific value is optimal across the board, so our models consider context and user information to decide the best user interface.

2.1.5 Content Curation. Content describing destinations, landmarks, accommodations, etc., comes in different formats like free text, structured surveys and photos; and from different sources like accommodation managers, guests, and public databases. It has huge potential since it can be used to attract and advertise guests to specific cities, dates or even properties, but it is also very complex, noisy and vast, making it hard to be consumed by users. *Content Curation* is the process of making content accessible to humans. For example, we have collected over 171M reviews in more than 1.5M properties, which contain highly valuable information about the service a particular accommodation provides and a very rich source of selling points. A Machine Learning model "curates" reviews, constructing brief and representative summaries of the outstanding aspects of an accommodation (Figure 1(b)).

2.1.6 Content Augmentation. The whole process of users browsing, selecting, booking, and reviewing accommodations, puts to our disposal implicit signals that allow us to construct deeper understanding of the services and the quality a particular property or destination can offer. Models in this family derive attributes of a property, destination or even specific dates, *augmenting* the explicit service offer. Content Augmentation differs from Content Curation in that curation is about making already existing content easily accessible by users whereas augmentation is about enriching an existing entity using data from many others. To illustrate this idea, we give two examples:

- *Great Value*: Booking.com provides a wide selection of properties, offering different levels of value in the form of amenities, location, quality of the service and facilities, policies, and many other dimensions. Users need to assess how the price asked for a room relates to the value they would obtain. "Great Value Today" icons simplify this process by highlighting properties offering an outstanding value for the price they are asking, as compared to other available options. A machine learning model analyses the value proposition of millions of properties and the transactions and ratings of millions of users and selects the subset of properties with a "Great Value" offer.
- *Price Trends*: Depending on the anticipation of the reservation, the specific travelling dates and the destination, among other aspects, prices display different dynamics. Since we have access to thousands of reservations in each city every day, we can build an accurate model of the price trend of a city for a given time and travelling dates. When the model finds a specific trend, we inform the users to help them make a better decision, either by encouraging them to choose a destination and dates that look like an opportunity, or discouraging particular options in favor of others. Note that in this case, the augmented item is not an accommodation but a destination (see Figure 1(c)).

2.2 All model families can provide value

Each family of Machine Learned Models provides business value. This is reflected in Figure 2 where each bar represents the relation between the median improvement on one of our core metrics by a model family and a baseline computed as the median improvement on the same metric for of all the successful projects (machine learning based or not), on a comparable period. Most of the families contribution are above the benchmark, one is below, but all of them make a significant contribution, and the collective effect is clearly positive.

The graph mentioned above shows the direct impact of Machine Learning based projects, measured at their introduction or when

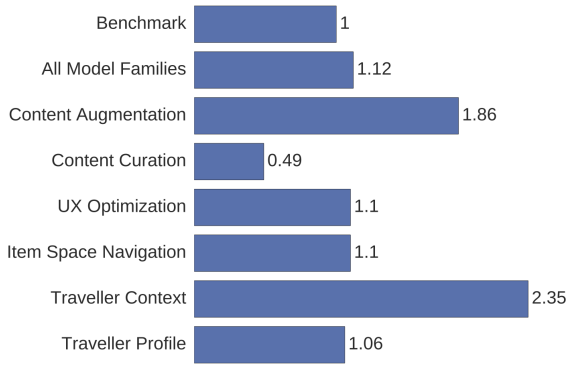


Figure 2: Model Families Business Impact relative to median impact.

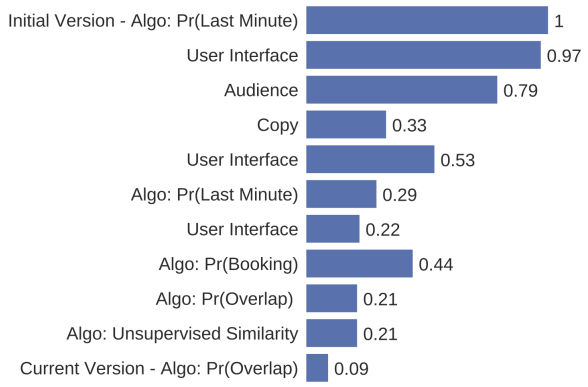


Figure 3: A sequence of experiments on a Recommendations Product. Each experiment tests a new version focusing on the indicated discipline or ML Problem Setup. The length of the bar is the observed impact relative to the first version (all statistically significant)

improving the model behind them. We have also observed models becoming the foundation of a new product, enabling value generation through other product development disciplines. Such indirect impact is hard to quantify, but the multiplying effect is clear and it is a concept that product teams exploit. As an example, Figure 3 illustrates the iterative process of the development of a destinations recommendations system. Each bar represents a successful iteration starting from the top and focusing on one aspect of the product: User Interface, Target Audience, Copy (captions, descriptions, messages, etc.), or the Algorithm itself. The length of the bar indicates the relative (all statistical significant) impact relative to the first iteration. All these improvements were enabled by the first algorithm, illustrating the indirect impact of Machine Learning projects through other disciplines.

3 MODELING: OFFLINE MODEL PERFORMANCE IS JUST A HEALTH CHECK

A common approach to quantify the quality of a model is to estimate or predict the performance the model will have when exposed to data it has never seen before. Different flavors of cross-validation are used to estimate the value of a specific metric that depends on the task (classification, regression, ranking). In Booking.com we are very much concerned with the value a model brings to our customers and our business. Such value is estimated through Randomized Controlled Trials (RCTs) and specific business metrics like conversion, customer service tickets or cancellations. A very interesting finding is that increasing the performance of a model, does not necessarily translates to a gain in value. Figure 4 illustrates this learning. Each point represents the comparison of a successful model that proved its value through a previous RCT, versus a new model. The horizontal coordinate is given by the relative difference between the new model and the current baseline according to an offline estimation of the performance of the models. This data is only about classifiers and rankers, evaluated by ROC AUC and Mean Reciprocal Rank respectively. The vertical coordinate is given by the relative difference in a business metric of interest as observed in a RCT where both models are compared (all models aim for the same business metric). We include a total of 23 comparisons (46 models). Visual inspection already shows a lack of correlation, deeper analysis shows that the Pearson correlation is -0.1 with 90% confidence interval (-0.45, 0.27), and Spearman correlation is -0.18 with 90% confidence interval (-0.5, 0.19). We stress that this lack of correlation is not between offline and online performance, but between offline performance gain and *business value* gain. At the same time we do not want to overstate the generality of this result, the external validity can be easily challenged by noting that these models work in a specific context, for a specific system, they are built in specific ways, they all target the same business metric, and furthermore they are all trying to improve it after a previous model already did it. Nevertheless we still find the lack of correlation a remarkable finding. In fact, such finding led us to investigate other areas of Booking.com and consistently found the same pattern. For example [8] highlights that the standard performance metric for Machine Translation (BLEU) exhibits a "rather tenuous" correlation with human metrics. Only where the offline metric is almost exactly the business metric, a correlation can be observed.

This phenomenon can be explained by different factors, we list the ones we found most relevant to share:

- *Value Performance Saturation*: It is clear that there are business problems for which it is not possible to drive value from model performance gains indefinitely, at some point the value vs performance curve saturates, and gains in performance produce no value gain, or too small gains, impossible to detect in an RCT in reasonable time.
- *Segment Saturation*: when testing a new model against a baseline we apply triggered analysis to make sure we only consider the users exposed to a change, that is, users for which the models disagree. As models improve on each other, this disagreement rate goes down, reducing the population of users that are actually exposed to a treatment, and with

that, the power to detect gains in value. More details about how we test competing models can be found in Section 7.4.

- *Uncanny Valley effect*: as models become better and better, they seem to know more and more about our users, or can predict very well what the user is about to do. This can be unsettling for some of our customers (see Figure 5 and [10]), which likely translates to a negative effect on value.
- *Proxy Over-optimization*: Usually, our Machine Learned models are supervised models that maximize certain observed variable, but not necessarily the specific objective business metric. For example, we might build a recommender system based on Click Through Rate because we know that CTR has a strong correlation or even causation with Conversion Rate, the business metric we really care about in this case. But as models get better and better, they might end up "just driving clicks", without any actual effect on conversion. An example of this is a model that learned to recommend very similar hotels to the one a user is looking at, encouraging the user to click (presumably to compare all the very similar hotels), eventually drowning them into the paradox of choice and hurting conversion. In general, over-optimizing proxies leads to distracting the user away from their goal.

It is challenging to address each of this issues on its own. Our approach relies on a fast cycle of developing hypotheses, building minimum models to test them in experiments, and using the results to keep iterating. Offline model performance metrics are only a health check, to make sure the algorithm does what we want to. This cycle drives us to focus on many aspects of the product development process besides the offline model performance, multiplying the iterative process along many dimensions. These include the Problem Construction Process described in the following section, qualitative aspects of a model (like diversity, transparency, adaptability, etc.), experiment design and latency. As an example consider a recommender system that predicts the rating a user would give to an accommodation. Minimizing RMSE looks like a reasonable approach. After a few successful iterations we hypothesize that the model is lacking diversity, so we create a challenger model that although still minimizes RMSE, somehow produces higher diversity. It is likely that this new model has a higher RMSE, but as long as it succeeds at increasing diversity and gives a reasonable RMSE, it will be used to test the hypothesis "diversity matters". Inconclusive results might point to adjustments of the model or experiment design, to make sure we give the hypothesis a fair chance. Negative results will likely reject the concept. Positive results on the other hand, will encourage diversity related changes, not only in the model but also in the User Interface, and even the product as a whole.

4 MODELING: BEFORE SOLVING A PROBLEM, DESIGN IT

The Modeling phase involves building a Machine Learning model that can contribute in solving the business case at hand. A basic first step is to "set up" a Machine Learning Problem, and we learned that focusing on this step is key. The Problem Construction Process takes as input a business case or concept and outputs a well defined modeling problem (usually a supervised machine learning problem), such that a good solution effectively models the given business case

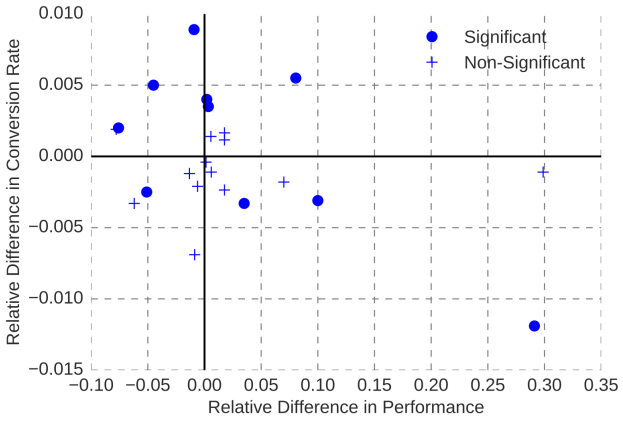


Figure 4: Relative difference in a business metric vs relative performance difference between a baseline model and a new one.

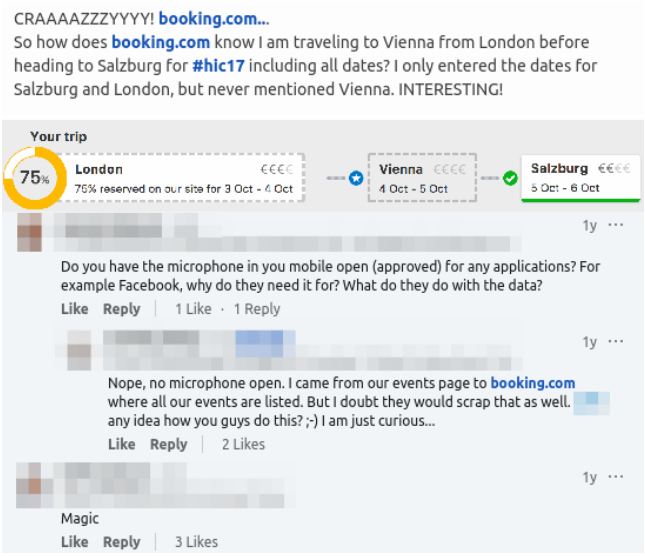


Figure 5: Uncanny valley: People not always react positively to accurate predictions (destination recommender using Markov chains).

or concept. The point(s) at which the prediction needs to be made are often given, which fixes the feature space universe, yet the target variable and the observation space are not always given and they need to be carefully constructed. As an example, consider the Dates Flexibility model mentioned before, where we want to know the dates flexibility of the users every time a search request is submitted. It is not obvious what flexibility means: does it mean that a user is considering more alternative dates than a typical user? or that the dates they will end up booking are different to the ones they are looking at right now?; or maybe it means that a visitor is willing to change dates but only for a much better deal, etc. For each of these definitions of flexibility a different learning setup can be used. For example, we could learn to predict how many different dates the user will consider applying regression to a specific dataset composed by users as observations, or to estimate the probability of changing dates by solving a classification problem, where the observations are searches, and so on. These are all constructed

machine learning problems, that, when solved, output a model of the Dates Flexibility of a user.

To compare alternative problems we follow simple heuristics, that consider among others, the following aspects:

- *Learning Difficulty*: when modeling these very subjective concepts, target variables are not given as ground truth, they are constructed. Therefore, some setups are harder than others from a learning perspective. Quantifying learnability is not straightforward. For classification problems the Bayes Error is a good estimate since it only depends on the data set, we apply methods from the work of Tumer & Ghosh [12]. Another popular approach that works well for ranking problems is to compare the performance of simple models against trivial baselines like random and popularity. Setups where simple models can do significantly better than trivial models are preferred.
- *Data to Concept Match*: some setups use data that is closer to the concept we want to model. For example, for the Dates Flexibility case we could create a data set asking users themselves if they know the dates they want to travel on, and then build a model to predict the answer. This would give a very straightforward classification problem, that, compared to the other options, sits much closer to the idea of Dates Flexibility. On the other hand, the data would suffer from severe selection Bias since labels are only available for respondents.
- *Selection Bias*: As just described, constructing label and observation spaces can easily introduce selection bias. An unbiased problem would be based on observations that map 1 to 1 to predictions made when serving, but this is not always possible or optimal. Diagnosing selection bias is straightforward: consider a sample of the natural observation space (users or sessions in the dates flexibility case), we can then construct a classification problem that classifies each observation into the class of the observations for which a target variable can be computed and the class of the observations for which a target variable cannot be computed. If this classification problem is easy (in the sense that a simple algorithm performs significantly better than random) then the bias is severe and must be addressed. Correcting for this type of bias is not obvious. Techniques like Inverse Propensity Weighting [11] and Doubly Robust [4] are helpful in some cases, but they require at least one extra model to build (the propensity model). Other approaches that have been applied successfully but not systematically are methods from the PU-Learning [9] and Semi Supervised Learning fields.

The Problem Construction process opens an iteration dimension. For a given business case and a chosen problem setup, improving the model is the most obvious way of generating value, but we can also change the setup itself. A simple example is a regression predicting the length of a stay, turned in to a multiclass classification problem; and a more involved example is a user preferences model based on click data switched to a Natural Language Processing problem on guest review data. Figure 3 shows a more concrete example. There are 6 successful algorithm iterations and 4 different setups: Pr(Last Minute) classifies users into Last Minute or not, Pr(Booking) is a conversion model, Pr(Overlap) models the probability of a user making 2 reservations with overlapping stay dates and Unsupervised Similarity models the similarity of destinations.

In general we found that often the best problem is not the one that comes to mind immediately and that changing the set up is a very effective way to unlock value.

5 DEPLOYMENT: TIME IS MONEY

In the context of Information Retrieval and Recommender Systems, it is well known that high latency has a negative impact on user behavior [1]. We quantified the business impact that latency has in our platform by running a multiple-armed RCT where users assigned to each arm were exposed to synthetic latency. Results are depicted in Figure 6 (bottom right quadrant). Each point is one arm of the experiment, the horizontal coordinate is the relative difference in observed (mean) latency between the arm and the control group, and the vertical coordinate is the relative difference in conversion rate. Crosses correspond to arms that did not show statistical significance and circles to arms that did. This is a single experiment with 4 arms (plus a control group), so we use Šidák correction to account for multiple testing. Visual inspection shows a clear trend, in which an increase of about 30% in latency costs more than 0.5% in conversion rate (a relevant cost for our business). This finding led us to hypothesize that decreasing latency can produce a gain in conversion. On the top left quadrant of Figure 6 we can see the effect of decreasing the latency, in 4 individual experiments in different devices and different pages of the site. All results are statistically significant, supporting the hypothesis.

This is particularly relevant for machine learned models since they require significant computational resources when making predictions. Even mathematically simple models have the potential of introducing relevant latency. A linear model, for example, might require to compute many (hundreds of thousands), and complicated features, or require to be evaluated on thousands of accommodations. Many pages in our site contain several Machine Learned models, some of them computed at the user level, others at the item level (destination, accommodation, attraction, etc.) or even at UI widget level. Even if each model is fast enough, the overall effect must be considered carefully.

To minimize the latency introduced by our models we use several techniques:

- *Model Redundancy*: Copies of our models are distributed across a cluster to make sure we can respond to as many predictions as requested, scale horizontally and deal with large traffic.
- *In-house developed Linear Prediction engine*: We developed our own implementation of linear predictions, highly tuned to minimize prediction time. It can serve all models reducible to inner products, such as Naive Bayes, Generalized Linear Models, k-NN with cosine or euclidean distance, Matrix Factorization models and more.
- *Sparse models*: The less parameters a model has, the less computation is needed at prediction time.
- *Precomputation and caching*: When the feature space is small we can simply store all predictions in a distributed key-value

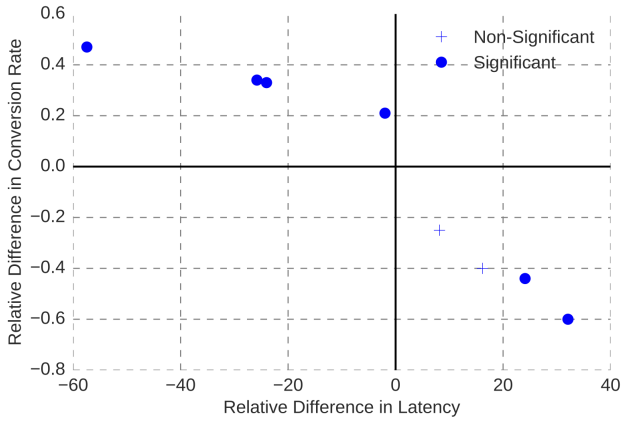


Figure 6: Impact of latency on conversion rate

store. When the feature space is too big we can still cache frequent requests in memory.

- *Bulking*: Some products require many requests per prediction. To minimize network load, we bulk them together in one single request.
- *Minimum Feature Transformations*: Sometimes features require transformations that introduce more computation, for example, we might cluster destinations geographically, and then learn parameters of a linear model for each cluster. At prediction time one could compute the cluster given the destination, and then invoke the model. We save one step by just expressing the model in terms of the destination, mapping the weight of a cluster to all the cities in it.

Most of these techniques are implemented by our Machine Learning Production service, which provides a simple interface to deploy and consume models in a variety of formats. This service abstracts away many challenging aspects of model deployment, including prediction latency, but also high availability, fault tolerance, monitoring, etc. Although these techniques are usually very successful at achieving low latency on an individual model level, there could always be the case where adding a fast model is "the last straw" that breaks our system. To detect this situation we use a method described in detail in section 7.3. The idea is to disentangle the effects of latency and the model itself on the business metric, so that we can decide whether there is a need to improve the latency or the model itself in one single RCT.

6 MONITORING: UNSUPERVISED RED FLAGS

When models are serving requests, it is crucial to monitor the quality of their output but this poses at least two challenges:

Incomplete feedback: In many situations true labels cannot be observed. For example, consider a model that predicts whether a customer will ask for a "special request". Its predictions are used while the user shops (search results page and hotel page), but we can only assign a true label to predictions that were made for users that actually booked, since it is at booking time when the special request can be filled in. Predictions that were made for users that did not book, will not have an associated true label.

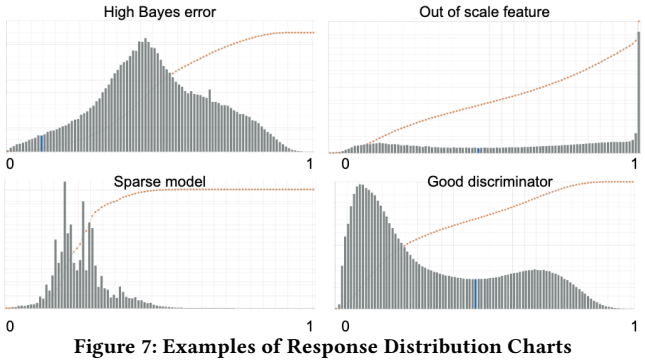


Figure 7: Examples of Response Distribution Charts

Delayed feedback: In other cases the true label is only observed many days or even weeks after the prediction is made. Consider a model that predicts whether a user will submit a review or not. We might make use of this model at shopping time, but the true label will be only observed after the guest completes the stay, which can be months later.

Therefore, in these situations, label-dependent metrics like precision, recall, etc, are inappropriate, which led us to the following question: what can we say about the quality of a model by just looking at the predictions it makes when serving? To answer this question for the case of binary classifiers we apply what we call Response Distribution Analysis, which is a set of heuristics that point to potential pathologies in the model. The method is based on the Response Distribution Chart (RDC), which is simply a histogram of the output of the model. The simple observation that the RDC of an ideal model should have one peak at 0 and one peak at 1 (with heights given by the class proportion) allows us to characterize typical patterns that signal potential issues in the model, a few examples are:

- A smooth unimodal distribution with a central mode might indicate high bias in the model or high Bayes error in the data
- An extreme, high frequency mode might indicate defects in the feature layer like wrong scaling or false outliers in the training data
- Non-smooth, very noisy distributions point to too excessively sparse models
- Difference in distributions between training and serving data may indicate concept drift, feature drift, bias in the training set, or other forms of training-serving skew.
- Smooth bimodal distributions with one clear stable point are signs of a model that successfully distinguishes two classes

Figure 7 illustrate these heuristics. The rationale behind these heuristic is that if a model cannot assign different scores to different classes then it is most likely failing at discriminating one from another, small changes in the score should not change the predicted class. It is not important where the stable point is (which could indicate calibration problems), it only matters that there is one, since the goal is to clearly separate two classes, one that will receive a treatment and one that will not. These are the advantages this method offers:

- It can be applied to any scoring classifier

- It is robust to class distribution. In extreme cases, the logarithm of the frequency in the RDC is used to make the cues more obvious
- It addresses the Incomplete Feedback issue providing Global Feedback since the RDC is computed considering all predictions
- It addresses the Delayed Feedback issue providing Immediate Feedback, since the RDC can be constructed as soon as a few predictions are made
- It is sensitive to both class distribution and feature space changes, since it requires very few data points to be constructed
- It can be used for multi-class classification when the number of classes is small by just constructing one binary classifier per class that discriminates between one class and the others (one vs all or one vs rest)
- It offers a label-free criterion to choose a threshold for turning a score into a binary output. The criterion is to simply use a minimum of the RDC in between the 2 class-representative modes. If the region is large, then one can choose to maximize recall or precision using the lower and upper bound of that region respectively. This is very useful when the same model is used in various points of the system like hotel page or search results page, since they have different populations with different class distributions.

The main drawbacks are:

- It is a heuristic method, it cannot prove or disprove a model has high quality
- It does not work for estimators or rankers

In practice, Response Distribution Analysis has proven to be a very useful tool that allows us to detect defects in the models very early.

7 EVALUATION: EXPERIMENT DESIGN SOPHISTICATION PAYS OFF

Experimentation through Randomized Controlled Trials is ingrained into Booking.com culture. We have built our own experimentation platform which democratizes experimentation, enabling everybody to run experiments to test hypotheses and assess the impact of our ideas [6]. Machine Learning products are also tested through experiments. The large majority of the successful use cases of machine learning studied in this work have been enabled by sophisticated experiment designs, either to guide the development process or in order to detect their impact. In this section we show examples of how we use a combination of triggered analysis with treatments design to isolate the causal effect of specific modeling and implementation choices on business metrics.

7.1 Selective triggering

In a standard RCT, the population is divided into control and treatment groups, all subjects in the treatment group are exposed to the change, and all subjects in the control group are exposed to no change. However, in many cases, not all subjects are eligible to be treated, and the eligibility criteria are unknown at assignment time. In the case of machine learning models, this is often the case since

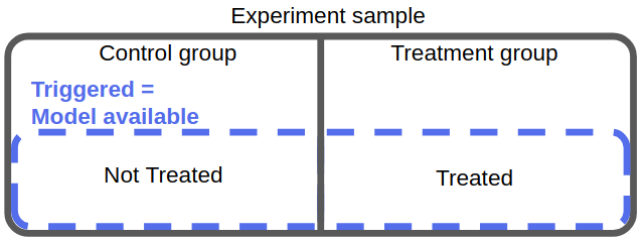


Figure 8: Experiment design for selective triggering.

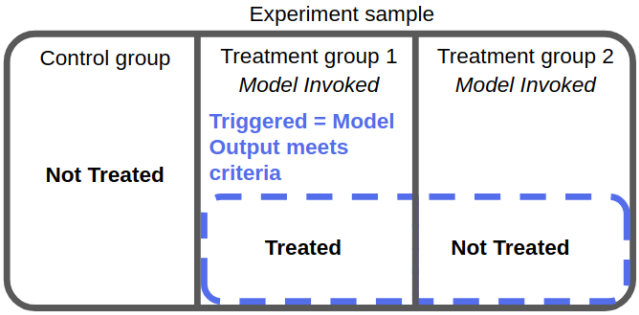


Figure 9: Experiment design for model-output dependent triggering and control for performance impact.

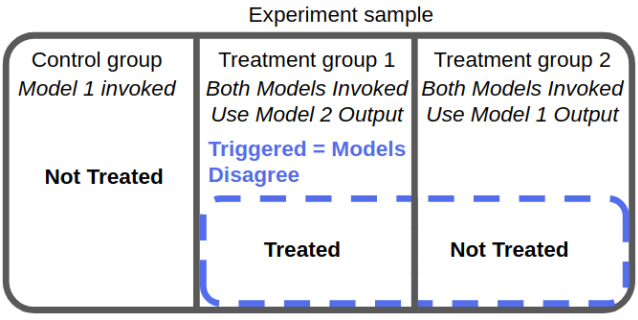


Figure 10: Experiment design for comparing models.

models may require specific features to be available. The subjects assigned to a group but not treated add noise to the sample, diluting the observed effect, reducing statistical power and inflating the False Discovery Rate. To deal with this situation, we apply Triggered Analysis [3], where only the treatable (or triggered) subjects in both groups are analyzed. Figure 8 illustrates this setup.

7.2 Model-output dependent triggering

Even when all the model requirements are met, the treatment criteria might depend on the model output. This happens for instance when we show a block with alternative destinations only to users identified as destination-flexible by the model. It may also be the case that subsequent steps fail or succeed depending on the model output, like fetching relevant items which may not be available. In such cases, some users are not exposed to any treatment, once more diluting the observed effect. Nevertheless, the setup of Figure 8 cannot be used since in the control group the output of the model is not known and therefore cannot condition the triggering. Modifying the control group to call the model is not advised, since

we also use this group as a safety net to detect problems with the experiment setup, and in such cases all the traffic can be directed to the control group while studying the issue. The setup for model output dependent triggering requires an experiment with 3 groups as shown on Figure 9. The control group C is exposed to no change at all, the two treatment groups $T1$ and $T2$ invoke the model and check the triggering criteria (e.g. output > 0) but only in $T1$ triggered users are exposed to a change. In $T2$ users are not exposed to any change regardless of the model output. The statistical analysis is conducted using only triggered subjects from both $T1$ and $T2$.

7.3 Controlling performance impact

The setup described in the previous section serves also as a way to disentangle the causal effect of the functionality on the user experience - for instance, new recommendations - from that of any slow down due to model computation. A comparison of metrics between C and $T1$ measures the overall effect of the new functionality, including any performance degradation. A positive result endorses the current implementation. Otherwise, we can still learn from two more comparisons. With C and $T2$ we can isolate and measure both the slowdown and its impact on the metrics of interest, since there is no change on the functionality between these variants. Conversely, $T1$ and $T2$ share the same computational load due to model invocation and are only different on the exposure to the new functionality, allowing to measure its effect regardless of the computational cost associated to the model. A positive result in this last comparison supports the new functionality, independently of the effect of the model on latency. More details on this topic can be found in [7].

7.4 Comparing Models

When comparing treatments based on models which improve on one another, there are often high correlations. As a toy example, consider a binary classification problem and consider model x, a successful solution with 80% accuracy. Model y improves on this result by correcting half of the mistakes of model x, while introducing only 5% new mistakes. These two models disagree only on at most 15% of the cases. For the other (at least) 85% of the cases, being on control or treatment does not result on a different experience, differences in metrics cannot be caused by difference in the model outputs, and therefore this traffic only adds noise. Figure 10 shows the setup for this situation, which is very similar to the previous one. In this case the triggering condition is *models disagree*, which means that the outputs from both models are required in $T1$ and $T2$. The control group invokes and uses the output from model 1, the current baseline, and also works as a safety net. As an additional gain, both $T1$ and $T2$ perform the same model associated computations, removing any difference due to performance between the models, isolating the causal effect of the difference between the model outputs on the target metric.

8 CONCLUSION

In this work we shared 6 lessons we have learned while developing 150 successful applications of Machine Learning in a large scale e-commerce. We covered all the phases of a Machine Learning project from the perspective of commercial impact delivery. All our lessons are about improving the hypothesis-model-experiment cycle: the semantic layer and model families help us to initiate as many cycles as possible; the finding that offline metrics are poorly correlated to business gains led us to focus on other aspects, like for example, Problem Construction, which adds a very rich iteration dimension; the finding that latency has commercial value led us to implement methods to keep it low giving each model the best chance to be impactful, and led to experiment design techniques to isolate its effects on business metrics; Response Distribution Analysis improved our ability to detect model issues right after deployed; and finally, experiment sophistication improved the iteration cycle by giving fast, reliable and fine-grained estimations of the effect of our choices and the validity of our hypotheses. In order to turn these lessons into actions we integrated ideas from various disciplines like Product Development, User Experience, Computer Science, Software Engineering, Causal Inference among others. Hypothesis driven iteration and interdisciplinary integration are the core of our approach to deliver value with Machine Learning, and we wish this work can serve as a guidance to other Machine Learning practitioners and sparkle further investigations on the topic.

ACKNOWLEDGMENTS

The authors would like to acknowledge the colleagues who built the models referenced in this work, as well as the UX designers, software developers, copy writers and project managers who transformed them into useful products.

REFERENCES

[1] Ioannis Arapakis, Xiao Bai, and B. Barla Cambazoglu. 2014. Impact of Response Latency on User Behavior in Web Search. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '14)*. ACM, New York, NY, USA, 103–112. https://doi.org/10.1145/2600428.2609627

[2] L Bernardi, J Kamps, Y Kiseleva, MJI Mueller, T Bogers, and M Koolen. 2015. The continuous cold start problem in e-commerce recommender systems. In *CEUR Workshop Proceedings*, Vol. 1448. CEUR-WS. org.

[3] Alex Deng and Victor Hu. 2015. Diluted treatment effect estimation for trigger analysis in online controlled experiments. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 349–358.

[4] Miroslav Dudík, John Langford, and Lihong Li. 2011. Doubly Robust Policy Evaluation and Learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML'11)*. Omnipress, USA, 1097–1104. http://dl.acm.org/citation.cfm?id=3104482.3104620

[5] Dietmar Jannach, Paul Resnick, Alexander Tuzhilin, and Markus Zanker. 2016. Recommender systems—beyond matrix completion. *Commun. ACM* 59, 11 (2016), 94–102.

[6] Raphael Lopez Kaufman, Jegar Pitchforth, and Lukas Vermeer. 2017. Democratizing online controlled experiments at Booking. com. *arXiv preprint arXiv:1710.08217* (2017).

[7] Timo Kluck. 2016. Using multivariant tests to determine performance impact. (2016). Retrieved Dec 04, 2017 from https://booking.ai/using-multivariant-tests-to-determine-performance-impact-c249ab9bfc16

[8] Pavel Levin, Nishikant Dhanuka, Talaat Khalil, Fedor Kovalev, and Maxim Khalilov. 2017. Toward a full-scale neural machine translation in production: the Booking. com use case. *arXiv preprint arXiv:1709.05820* (2017).

[9] Xiao-Li Li and Bing Liu. 2005. Learning from positive and unlabeled examples with different data distributions. *Machine Learning: ECML 2005* (2005), 218–229.

[10] Athanasios Noulas and M Stafseng Einarsen. 2014. User engagement through topic modelling in travel. In *Proceeding of the Second Workshop on User Engagement Optimization*. 2–7.

[11] James M Robins, Andrea Rotnitzky, and Lue Ping Zhao. 1994. Estimation of regression coefficients when some regressors are not always observed. *Journal of the American statistical Association* 89, 427 (1994), 846–866.

[12] Kagan Tumer and Joydeep Ghosh. 2003. Bayes error rate estimation using classifier ensembles. *International Journal of Smart Engineering System Design* 5, 2 (2003), 95–109.

[13] Kiri Wagstaff. 2012. Machine learning that matters. *arXiv preprint arXiv:1206.4656* (2012).