LSTM Response Models for Direct Marketing Analytics: Replacing Feature Engineering with Deep Learning

Mainak Sarkar, Arnaud de Bruyn

▶ **To cite this version:**

Mainak Sarkar, Arnaud de Bruyn. LSTM Response Models for Direct Marketing Analytics: Replacing Feature Engineering with Deep Learning. Journal of Interactive Marketing, 2021, 53, pp.80 - 95. 10.1016/j.intmar.2020.07.002 . hal-03491302

HAL Id: hal-03491302
https://hal.science/hal-03491302

Submitted on 22 Sep 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Version of Record: https://www.sciencedirect.com/science/article/pii/S1094996820301080
Manuscript_bfd1bf762951e89682761c74c37dae05

LSTM Response Models for Direct Marketing Analytics:

Replacing Feature Engineering with Deep Learning

Mainak Sarkar
ESSEC Business School

Arnaud De Bruyn♣
ESSEC Business School

Forthcoming in Journal of Interactive Marketing

Abstract – In predictive modeling, firms often deal with high-dimensional data that span multiple channels, websites, demographics, purchase types, and product categories. Traditional customer response models rely heavily on feature engineering, and their performance depends on the analyst's domain knowledge and expertise to craft relevant predictors. As the complexity of data increases, however, traditional models grow exponentially complicated. In this paper, we demonstrate that long-short term memory (LSTM) neural networks, which rely exclusively on raw data as input, can predict customer behaviors with great accuracy. In our first application, a model outperforms standard benchmarks. In a second, more realistic application, an LSTM model competes against 271 hand-crafted models that use a wide variety of features and modeling approaches. It beats 269 of them, most by a wide margin. LSTM neural networks are excellent candidates for modeling customer behavior using panel data in complex environments (e.g., direct marketing, brand choices, clickstream data, churn prediction).

Keywords – Long-short term memory neural network (LSTM); Recurrent neural network (RNN); Feature engineering; Response model; Panel data; Direct marketing.

♣ Corresponding author. Arnaud De Bruyn, Professor of Marketing, ESSEC Business School, Avenue Bernard Hirsch, 95000 Cergy, France. Email: debruyn@essec.edu.

© 2020 published by Elsevier. This manuscript is made available under the CC BY NC user license
https://creativecommons.org/licenses/by-nc/4.0/

INTRODUCTION

In direct marketing, a firm targets a customer with a marketing solicitation such as a catalog, a direct solicitation, or a coupon, and the customer decides whether or not to respond. Since soliciting a customer unlikely to respond is unprofitable, and not soliciting a potentially profitable customer leaves money on the table, the ability to predict customers' responses has long been a crucial endeavor for both practitioners and academics (e.g., Roberts and Berger 1999; Malthouse 1999).

Response models in direct marketing predict customer responses from past customer behavior and marketing activity. These models often summarize past events using features such as recency or frequency[1] (e.g., Blattberg et al. 2008; Van Diepen, Donkers and Franses 2009; Malthouse 1999), and the process of feature engineering has received significant attention (Kuhn and Johnson, 2019; Zheng, 2018).

In machine learning, a *feature* refers to a variable that describes some aspect of individual data objects (Dong et al. 2018). Feature engineering has been used broadly to refer to multiple aspects of feature creation, extraction, and transformation. Essentially, it refers to the process of using domain knowledge to create useful features that can be fed as predictors into a model.

However, feature engineering presents its own set of challenges.

First, the same features might identically summarize widely different behavior sequences (Blattberg et al. 2008; Fader, Hardie, and Lee 2005). Consider the customer behavior pattern depicted in Figure 1. All four customers in the figure have the same seniority (date of first

[1] Though formulation of optimal mailing strategies is beyond the scope of our research, research on optimal mailing strategies has extensively relied on the use of recency and frequency for model development. Some examples include Bult and Wansbeek (1995); Bitran and Mondschein (1996); Gonul and Shi (1998); Elsner et al. (2004); Gonul and Hofstede (2006); Simester et al. (2006); George et al. (2013).

purchase), recency (date of last purchase), and frequency (number of purchases). However, each of them has a visibly different transaction pattern. A response model relying exclusively on seniority, recency, and frequency would not be able to distinguish between customers who have similar features, but different behavioral sequences.

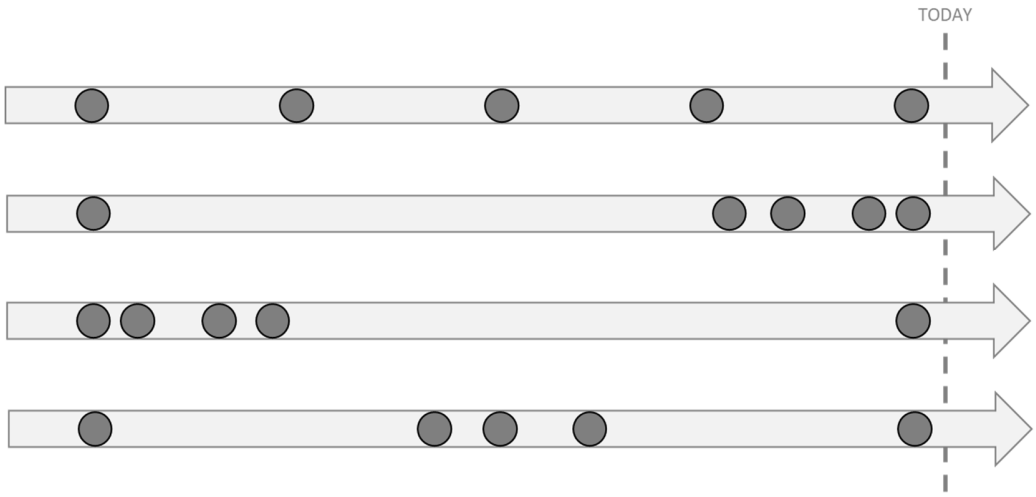

Figure 1 – Four customers with markedly different purchase patterns but identical features in terms of recency (last purchase), frequency (number of purchases), and seniority (first purchase).

Second, in a complex environment where there are multiple streams of data, such as in a data-rich environment where the analyst has access to historical marketing activity of various sorts (e.g., multiple types of solicitations sent through various marketing channels) and diverse customer behaviors (e.g., purchase histories across various product categories and sales channels) observed across different contexts (e.g., multiple business units or websites, see Park and Fader, 2004), the vast number and exponential complexity of inter-sequence and inter-temporal interactions (e.g., sequences of marketing actions, such as email-phone-catalog vs. catalog-email-phone) will make the data analyst's job arduous.

Let us reflect for a moment on one of the simplest and most commonly used features in direct marketing: *recency*, or the time elapsed since the last customer's purchase. How should the

analyst hand-craft relevant recency features in an environment spanning multiple product categories? Should she take into account the last absolute recency, regardless of the product category purchased (hence losing richness and granularity, and potentially hurting the model's predictive power)? Should she include in the model as many recency indicators as there are product categories in the data set (hence creating excruciating multicollinearity issues if customers buy from multiple product categories at each purchase occasion)? Should she combine individual and aggregate recency indicators? When crafting relevant recency indicators, should the analyst consider purchases in brick-and-mortar stores and purchases on the firm's website jointly, or should she treat these indicators separately?

When an analyst uses feature engineering to predict behavior, the performance of the model will depend greatly on the analyst's domain knowledge, and in particular, her ability to translate that domain knowledge into relevant features for the model. In complex environments, such as in the presence of multiple channels or multiple product categories, it can be quite challenging indeed for an analyst to capture all useful inter-sequence and inter-temporal interactions.

In this paper, we explore whether Long-Short Term Memory neural networks (LSTM), a special kind of Recurrent Neural Networks (RNN), which rely on raw sequential data and do away with feature engineering, can offer the promise of a solution to this general class of modeling problems in marketing.

In customer response models, the data is often in the form of panel data, where the firm's actions (e.g., solicitations) and customers' behavior (e.g., purchases) are observed repeatedly over time and along multiple dimensions (e.g., multiple channels or product categories).

Surprisingly, while RNN models are common in natural language processing, their applications to panel data –let alone marketing panel data– have been scarce, and even close to

nonexistent. In their seminal book, Goodfellow et al. (2016) cite applications of RNN in the domains of machine translation, prediction of text sequences, handwriting recognition, and speech recognition. Pointer (2019, p.70) mentions in passing that RNNs are particularly suited for "data that has a temporal domain (e.g., text, speech, video, and time-series data)," but dedicate the chapter to text analysis. Saleh (2018) dedicates an entire section to the numerous applications of RNN (pp.153-157), but exclusively cites natural language processing, speech recognition, machine translation, unidimensional time-series forecasting, and image recognition. However, as we will demonstrate, RNN models in general, and LSTM models in particular, seem particularly suited for panel data analysis.

We organize the paper as follows. In §1, we introduce the LSTM model as a special class of recurrent neural networks. Given the newness of the method to social scientists in general, and to marketing analysts in particular, we dedicate significant space to explain its inner working. While LSTM models take raw behavioral data as input and therefore do not rely on feature engineering or domain knowledge, our experience taught us that some fine-tuning is required to achieve optimal LSTM performance; in §2, we pay special attention to the proper calibration of an LSTM model, including parameter and hyperparameter tuning, which can be fully automated and do not require domain knowledge either. In §3, we demonstrate the superior performance of the LSTM model in a relatively simple, direct marketing setting with only donations (yes/no) and solicitations (yes/no). We show that the LSTM model, relying on raw data, achieves a better average fit and performance than the feature-based, benchmark models. In §4, we benchmark a vanilla LSTM model in a much more complex environment (e.g., multiple channels and donation types) against 271 hand-crafted models developed by about as many human analysts. The LSTM outperforms 269 of them. In §5, we discuss the marketing applications in which we expect LSTM

neural networks to prove valuable, and important technical considerations in the fast-moving field of deep learning in §6. We conclude in §7.

MODEL DESCRIPTION

Recurrent Neural Network (RNN)

In a traditional feedforward neural network, a vector x is processed through propagation in a neural network and produces an output vector y, as depicted in Figure 2 (A). Recurrent neural network (RNN) is a kind of artificial neural network (ANN) that is adapted to model sequential tasks. Rather than relying exclusively on the vector x to make its predictions, an RNN will also use part of the output of the previous iteration (the *hidden* state) as input for the next prediction (see Figure 2 (B)). By "unrolling" an RNN, as depicted in Figure 2 (C), it becomes apparent that this neural network architecture is particularly suited to model sequence data.

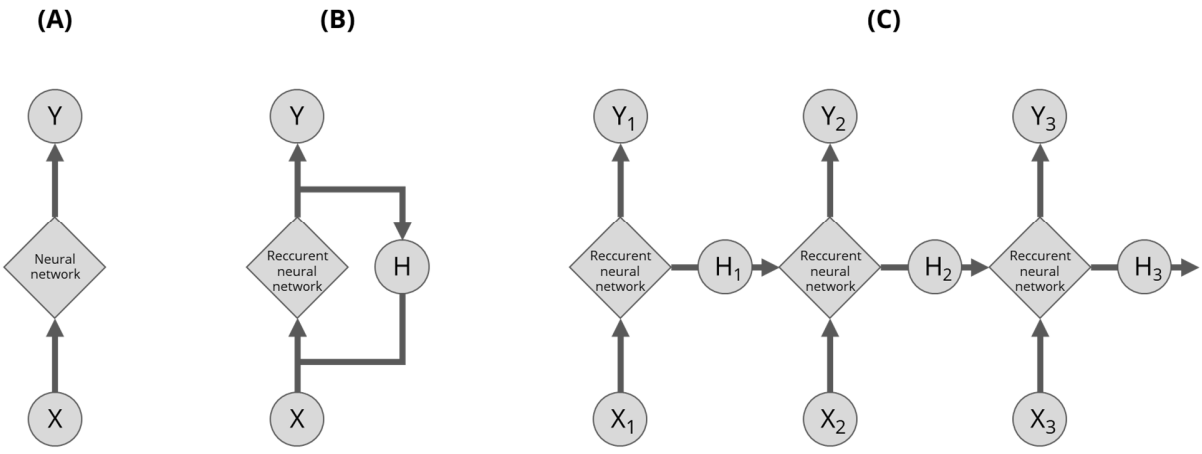

Figure 2 – Classic feedforward neural network (A), recurrent neural network (B), and "unrolled" graphical representation of a recurrent neural network (C) where we use sequence data (x_1, x_2, x_3) to make sequence predictions (y_1, y_2, y_3) while preserving information through the hidden states h_1, h_2, h_3.

The sequence-based specialization allows the RNN to process longer sequences than what would be practical by other forms of neural network architectures (Goodfellow et al., 2016). The RNN has the form of a chain of repeating modules, each passing a message to its successor module (Olah, 2015), and is, therefore, ideal for modeling sequential predictive tasks (Rumelhart et al., 1986). Each module in the sequence is sometimes referred to as timesteps based on their position in the sequence. The RNN processes a sequence of input vectors ($x_1, x_2, x_3, \ldots, x_T$), with each vector being input into the RNN model at its corresponding timestep or position in the sequence. The RNN has a multidimensional hidden state, which summarizes task-relevant information from the entire history, and is updated at each timestep as well.

Because of their typical high-dimensionality, the hidden states of RNN models are usually more potent than that of hidden Markov models (e.g., Netzer et al. 2008), which are commonly used in marketing to capture customer dynamics. The HMM has N discrete hidden states (where N is typically small) and, therefore, has only $\log_2(N)$ bits of information available to capture the sequence history (Brown & Hinton, 2001). On the other hand, the RNN has distributed hidden states, which means that each input generally results in changes across all the hidden units of the RNN (Ming et al., 2017). RNNs combine a large number of distributed hidden states with nonlinear dynamics to update these hidden states, thereby allowing it to have a more substantial representational capacity when compared with an HMM (Brown & Hinton, 2001; Hinton, 2013).

The RNN follows the following equations:

$$h_t = \tanh\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right) \quad (1)$$

$$\hat{y}_t = W_y h_t + b_y \quad (2)$$

Where:

W_{hh} Represents the recurrent weight matrix.

W_{hx} Is the input-to-hidden weight matrix.

W_y Is the hidden-to-output weight matrix.

b_h, b_y Represent the bias parameters.

Some researchers present Equation 1 in the following manner to simplify its notation:

$$h_t = \tanh (W_h[x_t, h_{t-1}] + b_h) \quad (3)$$

The above equations show that every position in the sequence has its corresponding input vector, hidden state, and output vector. This architecture leads to the formation of a recursive/recurrent function that makes the RNN share its parameters across the different positions in the sequence, thereby letting the model learn and generalize across timesteps.

The cross-entropy loss between the true y of the training data and the predicted $\hat{y}$ is used to create the cost function, in a step known as forward propagation. Then, backpropagation uses information from the cost function to calculate gradients with respect to the parameters used in the RNN. Since, in an RNN architecture, the network modules are arranged sequentially, forward propagation involves moving from left to right across timesteps. In contrast, backpropagation involves moving from right to left (as if moving backward in time). The backpropagation in the RNN is thus called backpropagation-though-time (BPTT) (Rumelhart et al., 1986; Goodfellow et al., 2016).

The gradients calculated by BPTT are used by a learning algorithm such as gradient descent with momentum, RMSProp (Tieleman and Hinton, 2012), or the Adam optimizer (Kingma and Ba, 2014) to tune the parameters towards the minimum of the cost function.

The learning mechanism of the recurrent neural network thus involves (1) the forward propagation step where the cross-entropy loss is calculated; (2) the backpropagation step where the gradient of the parameters with respect to the loss is calculated; and finally, (3) the optimization algorithm, that changes the parameters of the RNN based on the gradient.

The strengths and shortcomings of RNN

For natural language processing, an RNN would encode the sentence "A black cat jumped on the table" as a sequence of seven vectors (x_1, x_2, … x_7), where each word would be represented as a single non-zero value in a sparse vector[2] (Goodfellow et al., 2016). For instance, if we train a model with a vocabulary of 100,000 words, the first word "A" in the sentence would be encoded as a sparse vector of 100,000 numerical values, all equal to 0, except the first (corresponding to the word "A"), which would be equal to 1. The word "black" would be encoded as a sparse vector of 100,000 zero's, except the 12,853rd element (corresponding to the word "black") equal to 1, etc.

The RNN processes the entire sequence of available data without having to summarize it into features. Since customer transactions occur sequentially, they can be modeled as a sequence prediction task using an RNN as well, where all firm actions and customer responses are represented by elements in a vector.

For instance, suppose a firm solicits customers either through phone, mail or email (three channels), and customers may purchase across 17 product categories. All the analyst has to do is to encode each observation period (e.g., a day, a week, a month) as a vector of size 20, where all the values are equal to 0, except when a solicitation is sent, or a purchase is observed. If purchase seasonality is significant, e.g., if peaks in sales occur around Christmas, the analyst can also encode the current month using a one-hot vector of size 12, for a total vector length of 32 raw inputs.

[2] The dimensionality of the vector is often reduced through word embedding, a technique used in natural language processing, and with little applicability to panel data analysis. We skip this discussion in the interest of space.

While an RNN can carry forward useful information from one timestep to the next, however, it is much less effective at capturing long-term dependencies (Pascanu et al. 2013; Bengio et al., 1994). This limitation turns out to be a crucial problem in marketing analytics.

The effect of a direct mailing does not end after the campaign is over, and the customer has made her decision to respond or not. An advertising campaign or customer retention program can impact customers' behaviors for several weeks, even months. Customers tend to remember past events, at least partially. Hence, the effects of marketing actions tend to carry-over into numerous subsequent periods (Van Diepen et al., 2009; Lilien et al., 2013; Schweidel and Knox, 2013).

The LSTM neural network, which we introduce next, is a kind of RNN that has been modified to effectively capture long-term dependencies in the data (Hochreiter and Schmidhuber 1997; Gers et al. 1999).

The Long-Short Term Memory (LSTM)

In many real-world applications, such as in natural language processing, machine translation, or customer modeling, it is crucial to capture long-term dependencies in the data. However, during BPTT, the gradient of the vanilla RNN, when propagated over multiple steps, tends to explode or vanish, leading to difficulties in capturing long-term dependencies (Pascanu et al. 2013; Bengio et al. 1994).

The LSTM model is a kind of RNN designed explicitly to capture long-term dependencies and resolve the vanishing/exploding gradient problem (Hochreiter and Schmidhuber, 1997; Gers et al. 1999). The LSTM network forms a chain of repeating modules, like any RNN, but the modules, apart from the external recurrent function of the RNN, possess an internal recurrence (or self-loop), which lets the gradients flow for long durations without exploding or vanishing.

An LSTM model is governed by the following equations[3]:

$$\tilde{c}^{\langle t \rangle} = \tanh\left(W_c\,[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c\right) \tag{4}$$

$$\tau_u = \sigma\left(W_u\,[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u\right) \tag{5}$$

$$\tau_f = \sigma\left(W_f\,[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f\right) \tag{6}$$

$$\tau_o = \sigma\left(W_o\,[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o\right) \tag{7}$$

$$c^{\langle t \rangle} = \tau_u * \tilde{c}^{\langle t \rangle} + \tau_f * c^{\langle t-1 \rangle} \tag{8}$$

$$a^{\langle t \rangle} = \tau_o * \tanh\left(c^{\langle t \rangle}\right) \tag{9}$$

The LSTM, when compared with the vanilla RNN, apart from the original hidden state (referred to as $a^{\langle t \rangle}$ here), has an additional hidden state $c^{\langle t \rangle}$ (also referred to as the cell state) which specially acts as a memory cell. In Equation (4), the weight matrix W_c represents the combination of input-to-hidden and the recurrent weight matrices as was shown in the RNN Equation 3 with W_h.

The LSTM has three gates: the *update* gate ("u"), the *forget* gate ("f"), and the *output* gate ("o"). As shown in Equations 5 through 7, these gates are opened or closed based on the weight matrices (W_u, W_f, and W_o, respectively) of the corresponding gates and a sigmoid function. The cell state remembers relevant information from the past timestep through the gating mechanism of the update and the forget gates. As depicted in Equation 8, the internal recurrence which calculates the values of $c^{\langle t \rangle}$ uses the update and the forget gates to calculate a weighted average of the candidate $\tilde{c}^{\langle t \rangle}$ (i.e., the possible new value of $c^{\langle t \rangle}$) and $c^{\langle t-1 \rangle}$ from the last timestep.

For intuition, let us consider that the update gate has a value close to 1, while the forget gate is close to 0. The candidate $\tilde{c}^{\langle t \rangle}$ becomes the new value for $c^{\langle t \rangle}$, and all prior information is "forgotten." On the other hand, when the update gate is close to 0, and the forget gate is close to

[3] The sign * represents the Hadamard element-wise product.

1, the past value of $c^{<t-1>}$ is carried forward as $c^{<t>}$; the cell state is fully carried over, unchanged, and without loss of information, to the next timestep. Using this internal recurrence along with the gating mechanism, the LSTM can selectively carry forward relevant information across numerous timesteps.

Finally, at each timestep, the output gate controls how much of the current cell state $c^{<t>}$ is to be revealed to the hidden state $a^{<t>}$. The hidden state and the hidden-to-output weight matrix, along with the activation function (as shown in Equation 10), produces the predicted output at that timestep. For our transaction incidence model, since we are predicting customer responses as binary outcomes, we use a sigmoid function σ:

$$\hat{y}^{<t>} = \sigma\,(W_y\, a^{<t>} + b_y) \tag{10}$$

It is worth noting that though our study focuses on LSTM neural networks, there are other variants of the RNN as well such as the Gated Recurrent Unit (GRU) which use internal recurrence and gating mechanism along with the external recurrence of the RNN (Cho et al. 2014; Chung et al. 2014). However, research seems to suggest that none of the existing variants of the LSTM may significantly improve on the vanilla LSTM neural network (Greff et al. 2016). For this paper, these results encouraged us to consider the standard LSTM neural network instead of its other variants. We note, however, that deep learning is a fast-moving and rich research area, and other RNN variants may prove more suitable in the future.

LSTM Model Calibration

Bias, variance, and model capacity

As discussed in the LSTM model section, the parameters of the LSTM module/cell are W_u, W_f, W_o, W_c, b_u, b_f, b_o, and b_c. We use the parameters W_y and b_y to generate the predictions of

$\hat{y}^{<t>}$ from the hidden state of the LSTM. The dimension of the LSTM weight matrices depends on the dimension of the hidden state (referred to as hidden units) and the number of input features[4] in x.

At each timestep, we submit relevant variables x, such as marketing actions (e.g., solicitations), customer behavior (e.g., purchase occurrences), and seasonality indicators (e.g., month), in the form of a vector of dummy variables. In our illustration, the y variable is a vector of size one that indicates whether the customer has purchased during the following period, although the dependent variable can easily include multiple indicators.

In the first empirical illustration, we will predict the likelihood an individual will donate to a charity over a specific period based on past donation and solicitation histories. The calibration and prediction setup for our LSTM is displayed in Figure 3.

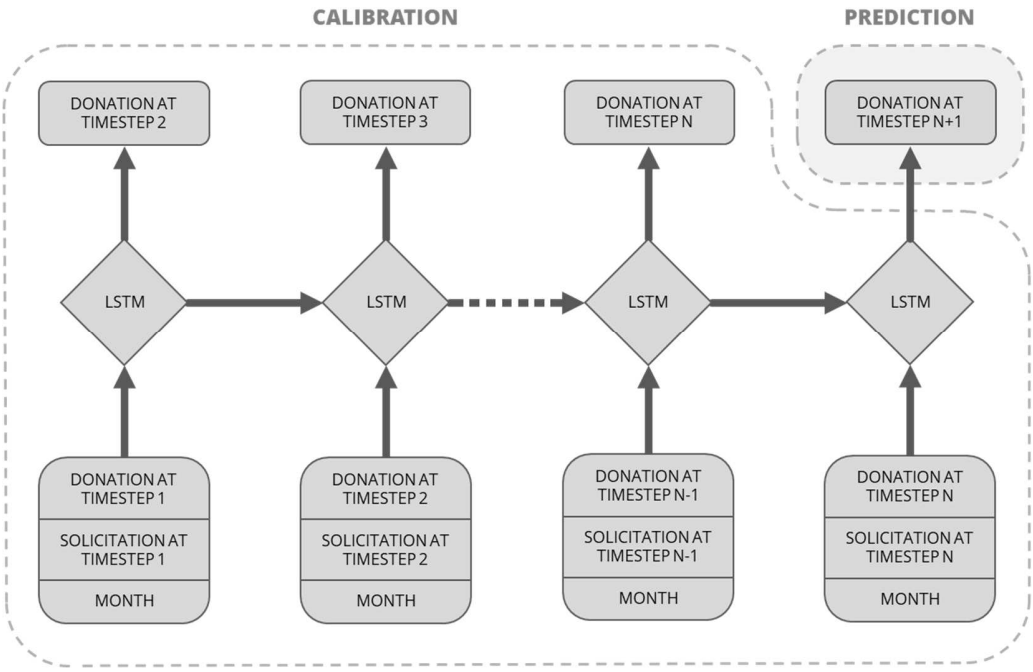

Figure 3 – LSTM network architecture for customer response prediction.

[4] The term 'feature' here is used to refer to the dimension of the x-variables that we input to the LSTM model at each timestep. These features are not based on using domain knowledge to extract relevant predictors, but rather uses un-summarized data.

The analyst sets the number of hidden units in the LSTM exogenously. Settings that are used to control the learning of the algorithm, but are not parameters tuned by the learning algorithm itself, are referred to as hyperparameters (Kuhn and Johnson 2013). Apart from the number of hidden units, the LSTM neural network has other hyperparameters such as the learning rate, batch size (how many observations are submitted to the LSTM per learning iteration), or the relative contribution of norm penalties to the cost function (for regularization purpose).

While training a model, the analyst aims at setting the parameters and hyperparameters such that the model reaches optimal capacity (Goodfellow et al., 2016) and therefore maximizes the chances that the model will generalize well to unseen data. Models with low capacity would underfit the training set and hence have a high bias. However, models with high capacity may overfit the training set and exhibit high variance. *Representational* capacity is the ability of the model to fit a wide range of functions. However, the *effective* capacity of a model might be lower than its representational capacity because of limitations and shortcomings, such as imperfect optimization or suboptimal hyperparameters (Goodfellow et al., 2016).

To increase the match of the model's effective capacity and the complexity of the task at hand, the analyst needs to tune both the parameters and the hyperparameters of the model. Given how sensitive LSTM models are to hyperparameter tuning, this area requires particular attention.

Hyperparameter tuning

To achieve the model's optimal capacity, we start with a wide range of possible values for hyperparameters, each having the potential to increase or decrease model capacity, and modify them in the hyperparameter tuning process.

In our application, the hyperparameters are:

1. The *number of hidden units*. Too few hidden units would hinder the capacity of the model to identify complex yet meaningful relationships in the data, though too many hidden units would increase the chances of model overfitting.
2. The *norm penalty* for the recurrent weights of the LSTM model. It is recommended to use either L1 (linear) and/or L2 (quadratic) regularization[5], each adding a penalty to the cost function based on the amplitude of the model parameters. A too-small penalty might not act effectively against model overfitting, while a too-large penalty would decrease model capacity.
3. The *learning rate* (the step size) of the learning algorithm is crucial for optimization purposes. When the learning rate is too high, learning becomes unstable, and errors might increase uncontrollably. When the learning rate is too low, training becomes extremely slow and might get stuck in regions of high error (Goodfellow et al. 2016). Although the learning rate does not control the representational capacity of the model, it may significantly hinder its effective capacity. The step size of the learning algorithm is often considered the most important hyperparameter of the LSTM model (Greff et al. 2016).
4. The *batch* size, or the number of examples provided to the algorithm at each learning iteration. Bengio (2012) mentions that the mini-batch size is usually chosen to be between 1 and a few-hundreds. While batch size may impact the speed of learning, in our experience, it does not seem to affect model performance much.

[5] Dropout is another form of regularization which can be used against overfitting, particularly in sequence networks (Srivastava et al. 2014; Gal et al. 2016).

We, therefore, use hyperparameter tuning to set (1) the number of hidden units, (2) the norm penalty, and (3) the learning rate. The hyperparameter tuning process involves the following steps:

1. Set a combination of hyperparameters;
2. Train the model on a portion of the data set reserved for that purpose (the training set);
3. Observe the loss (i.e., error) on the remainder of the data set (the validation set). Therefore, we test the model capacity on a portion of the data not used to train it;
4. Repeat steps 1 through 3 with different combinations of hyperparameters, and keep the combination that leads to the lowest loss in the validation set.

As the number of hyperparameters and their range grow, the search space becomes exponentially complex, and tuning the models manually or by grid-search becomes impractical. Bayesian optimization for hyperparameter tuning provides hyperparameters (step 1) iteratively based on previous performance (Shahriari, Swersky, Wang, Adams, and De Freitas, 2015). We use Bayesian optimization to search the hyperparameter space for our model extensively. In our experience, this approach proves far superior to the alternatives (i.e., manual or grid search). We provide more details on the Bayesian optimization approach for hyperparameter tuning in Web Appendix 1 and report computational considerations in Web Appendix 2.

Next, we present two empirical applications where we compare the performance of the LSTM model against the performance of more traditional –and sometimes much more complex– benchmark models.

EMPIRICAL APPLICATION #1

Introduction

LSTM neural networks do not rely on feature engineering. In complex settings (e.g., multidimensional data with inter-sequence and inter-temporal interactions), this is a significant advantage over hand-crafted, traditional models.

In contrast to the second application, which will offer a more realistic context, the first application presents the most straightforward proof of concept imaginable. The data tracks only two variables over time: whether a charity has solicited its contacts during a specific period (0/1), and whether they donated (0/1). The study follows the classic firm solicitation–customer reaction paradigm used extensively in prior research in direct marketing (Bult and Wansbeek, 1995; Colombo and Jiang, 1999; Malthouse 1999; Donkers et al., 2006; Van Diepen et al., 2009).

The data consist of the donation and solicitation histories of 6,134 donors over nine years. These donors were acquired by the charitable organization over the first seven years, followed by an additional year of observations so that even the most recent donors have at least one year of historical data on which we base our model predictions. We depict the data structure in Figure 4.

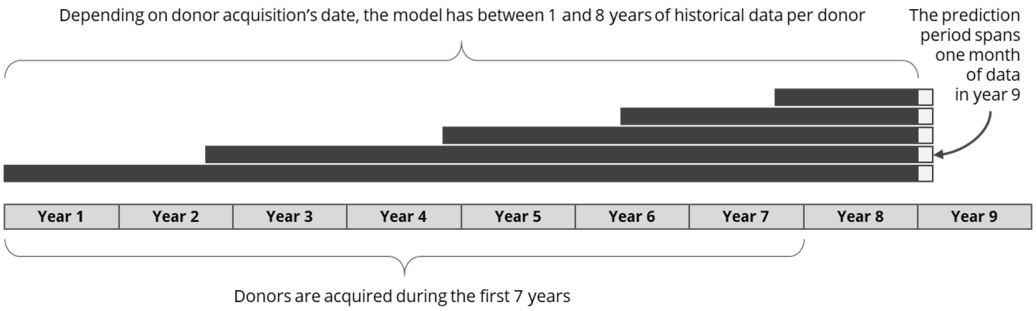

Figure 4 – The data set consists of 6,134 donors acquired over seven years, followed by an additional 8th year of observations, and a 9th year to validate the predictions. Each donor has between 1 and 8 years of data available, plus one year reserved for the holdout sample.

We hypothesize that the firm is interested in predicting donors' monthly activity (i.e., will this donor be donating during the upcoming month?). Donations are highly cyclical with some months receiving more donations (e.g., December) and some months receiving less (e.g., February), and comparing model performance on any specific month may be misleading. Consequently, we repeat the exercise for 12 consecutive months, each representing a stand-alone prediction that incorporates all prior data available up to that point in time. Predictions are not sequential; we do not incorporate past predictions into the model calibration at any stage. In essence, we simulate the firm's modeling efforts to predict its donors' behaviors, where the firm repeats the model calibration twelve times per year, each time one month apart.

We first test the performance of the model for January of the 9th year of data. Next, we incorporate the actual solicitations and donations in January into the calibration data to predict donations in February. We predict responses for February of the 9th year and continue this process for the remaining months of the 9th year. We test the performance of our model over 12 successive monthly predictions to demonstrate that the performance is not happenstance to the particular month under consideration, such as predicting behaviors during an active month (e.g., December) or a much less active one (e.g., August). Therefore, we test the model on a total of 12 independent predictive tasks over 6,134 donors, for a total of 73,608 predictions. Note that this setting does not rely on any look-ahead, and each prediction is made independently of the previous ones. We illustrate the rolling process in Figure 5.

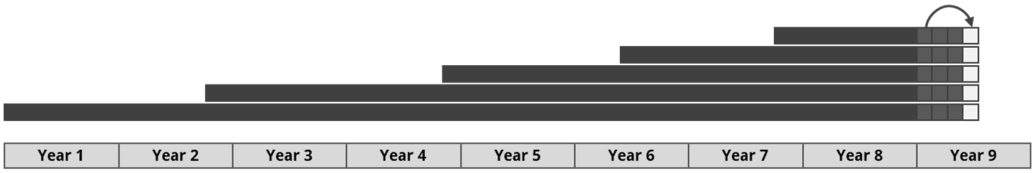

Figure 5 – Once we obtain the predictions for one month, we incorporate the actual solicitations and donations observed for that month in the calibration data set, we re-calibrate the model, and obtain predictions for the next month. We continue this process for an entire year.

Benchmark Models

To choose our benchmark models, we consulted publications in direct marketing (e.g., Roberts and Berger 1999; Malthouse 1999), spoke with experts and executives of charity organizations that use direct mailings, and referred to discussions in marketing academia on the industry standards for direct mailing response models (Gonul, Kim and Shi 2000). The industry typically uses a penalized logit/probit model with a binary dependent variable (donation/no donation) and RFM variables along with seasonality predictors as independent variables. We follow a similar approach and use a penalized logistic regression model with hand-crafted features as the benchmark model, against which we will compare the performance of the LSTM model. To limit the risk of overfitting, we also add regularization to the logistic regression benchmark model, as we do for the LSTM model. The elastic-net regularization combines the L1 and L2 regularization in an effective way to prevent overfitting (Zou and Hastie, 2005). For the penalized logit benchmark model, we use elastic-net regularization, with the contribution of the norm penalty set by cross-validation using *glmnet* (Friedman, Hastie, and Tibshirani, 2009).

As an additional benchmark, we test the random forest model, an ensemble learning method (Breiman 2001). Here we keep the number of trees to grow at 128 and tune the number of variables randomly sampled as candidates at each split (Oshiro et al. 2012).

The features are listed in Table 1. The model has a total of 24 features, plus an intercept, for a total of 25 parameters to estimate.

[Table 1]

As illustrated in Figure 3, an LSTM model makes as many predictions as there are timesteps in the calibration data: the data from period 1 to N is entered sequentially in the model as predictors, and the model predicts donations for periods 2 to N+1. The predictions for periods 2 to N are used to compute the prediction errors and calibrate the model, whereas the prediction for period N+1 is the metric of interest that we use *ex-post* to measure the actual predictive accuracy of the model.

In the interest of comparison fairness, we use a similar approach with the benchmark model. Therefore, if an individual has 30 months of observations, 29 observations (one for each month) are encoded −with their corresponding features− in the calibration data and estimated to fit the benchmark model, each with one more month of data than the preceding.

Results

We compare the performance of the LSTM and the benchmark models for 12 successive months. One of the most managerially-relevant success metrics is the *lift* of the model. Analysts routinely use lift charts to assess the performance of response models in direct marketing (Ling and Li 1998; Malthouse 1999; Kuhn and Johnson 2013). For example, if a direct marketer wants to target *m* of *n* possible customers, there are $\binom{n}{m}$ possible ways to pick the customers to solicit. In direct marketing campaigns, where the organization is interested in sending mailings to a selected fraction of its customer base, the lift is a useful and popular metric, since it assesses the performance of a response model in its capacity to select the most responsive customers.

To evaluate the performance of a model on the lift chart for a particular mailing occasion, we arrange the donors in descending order by the predicted donation probability. For a non-informative baseline, X% of donors would contain X% of the donations, i.e., 10% of donors selected randomly would contain 10% of all donations. However, if the model is informative and ranks donors effectively, the top 10% of the donors may account for, say, 36% of the total donations, thereby giving a lift of 3.6. If a direct marketer wants to target 10% of the customer base, he/she would pick the top decile of the customer base (ranked on model prediction) and would predict the performance of the said campaign by calculating the lift at 10%.

Note that the null model, with totally random recommendations, achieves a lift of 1. We take this into account while calculating the improvement in lift for the LSTM model over the benchmark models.

To evaluate the performance of the LSTM model in the direct marketing context, we report the lifts of the LSTM and benchmark models (a logistic regression model with elastic-net regularization, and a random forest model) at 1%, 5%, 10%, and 20% in Table 2.

[Table 2]

The logit (resp., random forest) model provides an average lift at 5% of 5.33 (resp., 5.42) over 12 independent predictive tasks. In comparison, the LSTM model, which does not rely on domain knowledge or feature engineering, and uses the raw data as predictors, achieves an average lift at 5% of 6.48. The LSTM model, therefore, provides a +26.60% (resp., +24.05%) improvement[6] over the logit model.

The logit model performs remarkably well at high lift values (i.e., 20%), whereas the random forest model shines at lower lift values (lift at 1%). This result might suggest that the best

[6] Since, by definition, the null model (i.e., random predictions) achieves a lift of 1, the improvement in lift between two models is computed as (liftA - 1) / (liftB - 1).

traditional model to deploy depends on the degree of targeting the analyst seeks. Random forest models are particularly good at identifying tiny niche of super-responsive donors, and therefore are well suited for ultra-precise targeting. Logit models generalize well. Hence, they appear more appropriate to target wider portions of the donor base.

Interestingly, though, the LSTM model beats both benchmark model across the board, and perform well at all lift levels.

[Table 3]

For completeness, we report additional metrics, such as RMSE, LogLoss, Precision, Recall, and F-measure in Table 3. The LSTM model achieves superior results on these metrics as well. On average, the logit (resp., random forest) model suffers from an RMSE +1.33% (resp., +1.07%) higher than the LSTM model, as well as a LogLoss +2.11% (resp. +74.88%) higher. The detailed results for each of the 12 independent tasks are available in Web Appendix 3.

EMPIRICAL APPLICATION #2

Objective

While an LSTM model does not depend on the analyst's ability to craft meaningful model features, traditional benchmarks do heavily rely on human expertise. Consequently, when an LSTM model shows superior results over a traditional response model –as we have shown in the previous illustration–, we cannot ascertain whether it is due to the superiority of the LSTM model, or to the poor performance of the analyst who designed the benchmark model.

To alleviate that concern, we asked 297 graduate students in data science and business analytics from one of the top-ranked specialized masters in the world[7] to compete in a marketing analytics prediction contest. Each author participated and submitted multiple models as well, for a total of 816 submissions. With the LSTM model competing against such a wide variety of human expertise and modeling approaches, it becomes easier to disentangle the model performance from its human component.

Problem description

Participants' goal was to predict who was likely to donate to a charity for one of its specific fundraising campaigns, and how much money they would likely give. By combining these two predictions (likelihood × amount), contestants would obtain an expected revenue from each solicited individual. Since every solicitation cost 2 € (a fake, unrealistic figure used for the purpose of the exercise), soliciting an individual with expected revenue of less than 2 € (e.g., someone who had 5% chances of giving 17 €, for an expected revenue of 85 cents) for that campaign was, therefore, deemed unprofitable.

The dataset came from a direct marketing campaign of 123,672 solicited donors. The participants had access to a calibration data of 61,928 individuals for whom we provided both the responses (yes/no) and donation amount to the campaign. The holdout data consisted of the remaining 61,744 individuals who had been solicited, but for whom the responses were unknown to the data analysts.

Participants were asked to submit a file of 61,744 individual decisions indicating whether the charity should solicit or not each contact in the holdout data. If the participant indicated a

[7] At the time of this writing, the Master in Data Sciences & Business Analytics, jointly organized by the ESSEC Business School and CentraleSupélec, was ranked #3 worldwide by QS World University Rankings.

solicitation for a particular individual, the charity endured a solicitation cost of 2 €, and we added the donation made by that individual to the gross revenue of the campaign, if any. If the participant indicated the individual should *not* be solicited (because his expected revenue was inferior to the threshold of 2 €), no solicitation cost was endured. However, donations made by the unsolicited individuals on that fundraising campaign were deemed to have never happened. For the purpose of this exercise, we assumed that no individual was going to donate on that campaign if the charity did not solicit him or her. We illustrate this process in Table 4.

[Table 4]

The contestants' objective was to maximize the net financial performance of the campaign. The instructor informed them that finding the right model features, and avoiding model overfitting, were vital for this assignment. All students followed several advanced classes prior, and were trained in marketing analytics, feature engineering, SQL queries, and predictive modeling techniques. They also had completed another assignment on the same charity database and were, therefore, familiarized with the organizational context of the exercise.

Each participant was given up to three trials. They submitted their recommendations (a text file of 61,744 yes/no decisions) through an online interface, and the website reported their net financial results in real-time (gross revenue, minus solicitation costs). The website provided no information about individual donations.

At the end, only contestants' top performance was retained and compared to the financial results of their cohort. The best performing contestant received 20/20 for the assignment, which constituted a significant fraction of the course's overall grade. The contestant with the worst performance received 8/20 (a failing grade). All others were graded linearly based on their rank in the cohort, with a grade of 14/20 for the students in the 50th percentile. Students who managed to achieve a higher performance than the instructor −who followed the same instructions, and

submitted recommendations of his own– received a bonus point on the entire course. The exercise is, therefore, strongly incentive-aligned.

Database

The database included the complete donation history of the 123,672 individuals originally solicited for the fundraising campaign, for a total of 1,066,376 one-off donations, and 1,364,500 automatic deductions (i.e., automated monthly payments by credit card or bank wire). Data specified to which campaign the donations were linked to, if any, as well as the mean of payment (e.g., check, credit card, bank wire, cash) and the channel (Web donation or not). The most recent donor was acquired two weeks before the launch of the campaign; the most ancient made her first donation 26 years prior.

The database also contained the most recent ten years of solicitation history, for a total of 4,365,405 solicitations (earlier solicitations were not recorded in the database). It specified whether it was an online (i.e., email) or an offline campaign. Notice that a donor could be solicited by email (online) but decide to donate by check (offline).

Besides, each contact's ZIP code, first name, and prefix (Mr, Mrs, Dr., etc.) was known.

Benchmark models

We assigned this exercise to 6 classes (i.e., cohorts) over three years, for a total of 299 graduate students. Two never submitted their assignment. Thirty participants made errors that prevented them from competing fairly[8], and we do not report their results. For the remaining 267 graduate students, we only report their best trials.

[8] For instance, students (a) made typo in the reference dates used to compute recency, (b) predicted log(amount), but forgot to exponentiate their predictions to obtain the actual amount, (c) made coding errors and mixed up the contacts order in their output file, resulting in what were essentially random recommendations, etc.

The two authors competed as well and submitted three recommendations each. Following experts' recommendations, we additionally retained the features of the authors' best-performing model and fed them into random forests and XG Boost models as well. The authors, therefore, submitted a total of 4 models to the competition, for a total of 271 entries.

The net financial results of the contestants ranged from 190,239 € to 225,063 €, with an average of 210,144 €, and a median of 211,649 €. We report the distribution in Figure 6.

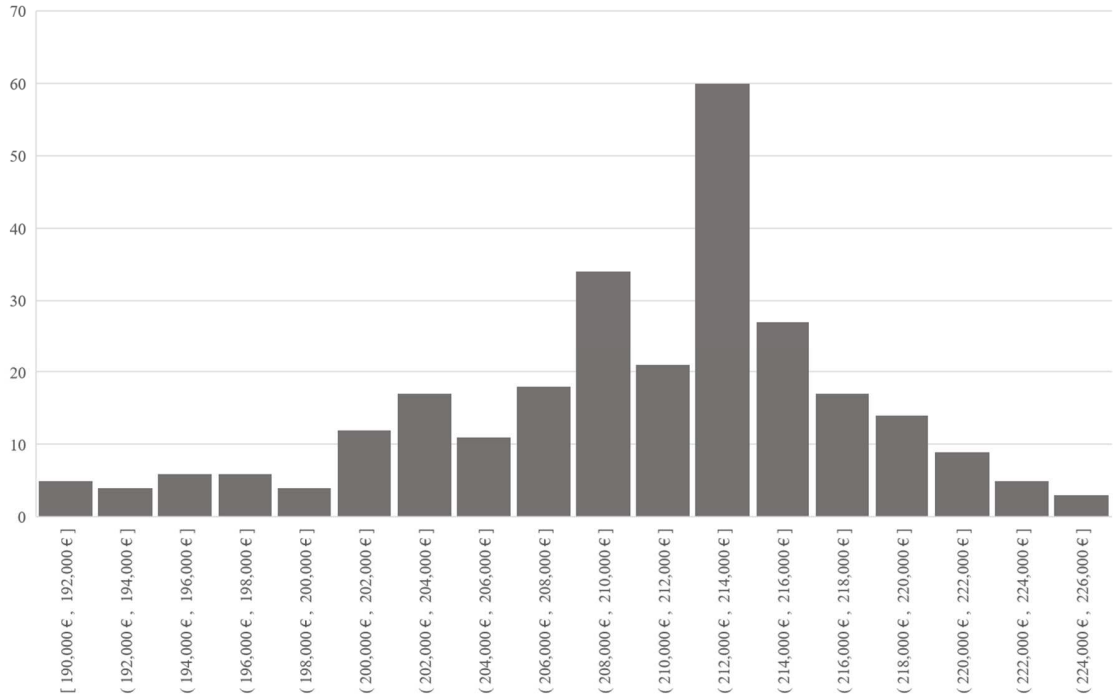

Figure 6 – Performance distribution (best trials) of the 271 entries in the predictive modeling competition. Results ranged from 190,239 € to 225,063 €, with an average of 210,144 €. Sixty entries felt between 212,000 € and 214,00 € (mode of the distribution.)

The authors achieved a performance of 218,030 € (author #1), 220,679 € (author #2 and course instructor), 217,378 € (random forest) and 215,093 € (XG Boost). Out of the 267 students, 11 managed to outperform the instructor, who had 20 years of experience in fundraising analytics and had developed professional scoring models for the WWF, the Salvation Army, and the Red

Cross, among others. We take that result as a strong signal that the students took the assignment seriously, and a testimony of the quality of their work.

Although several contestants explored advanced models such as feedforward neural networks, random forests, or support vector machines, these more elaborate models appeared to suffer from overfitting. Among the top 20 results, all used simple logistic (for the response model) and linear regressions (for the amount model) with lasso (and/or ridge) penalties. All top performers developed a wide variety of advanced features.

Table 5 reports the most common features designed and used by the contestants in their modeling efforts. The number of features per model ranged from six to several hundred.

[Table 5]

LSTM Model

For this exercise, the authors developed two separate LSTM models. The first one predicted the likelihood that each donor was going to respond favorably to the solicitation (0/1), and we calibrated it on the entire calibration data (N=61,928). The second LSTM model predicted the donation amount in case of donation, and we calibrated it on the individuals who donated in the calibration data (N=6,456). Both models were then applied to the holdout data and combined to obtain expected revenue from a solicitation.

For the purpose of sequence generation, we grouped the data in bimonthly increments, for a total of 24 steps per calendar year (e.g., January 1st to 15th is period 1, January 16th to 31st is period 2). Both LSTM models used the same sequences of raw indicators as inputs, namely:

1. Online solicitation (0/1)
2. Offline solicitation (0/1)
3. Online, one-off donation (0/1)

4. Online, automatic deduction (0/1)
5. Offline, one-off donation (0/1)
6. Offline, automatic deduction (0/1)
7. One-off donation amount (0/log(amount))
8. Automatic deduction amount (0/log(amount))

For instance, if a contact donates 50 € by check on February 4th, and is solicited by email on February 11th, the sequence data for that period (3rd period of the year) indicates "online solicitation = 1," "offline, one-off donation = 1," and "one-off donation amount = log(50)," with all the other indicators equal to zero.

The only differences between the two independent LSTM models are (a) the data we use to train the models and (b) the output functions. For the response model, the output is processed through a sigmoid function to ensure a probability between 0 and 1; for the amount model, the output is exponentiated to guarantee an amount prediction in the positive domain.

The authors' first trial mixed an LSTM model for response prediction with a simple linear regression for the amount model and achieved a net financial performance of 223,004 €. The second trial implemented an LSTM model for the amount prediction as well. Combined with the previous LSTM response model, it achieved a net performance of 224,233 €. While the first two trials used a simple grid search for hyperparameter turning, we deployed a full-blown Bayesian optimization search for the third trial. Still, results did not improve, demonstrating that a simple grid search was sufficient to achieve optimal results, at least in this application.

Competing against 271 benchmark models (4 of which were designed by the authors), the full LSTM model achieved third place. A net result of 224,233 € is 2.01 standard deviations above the contestants' average performance and 0.37% (0.118 s.d.) below the top-performing entry.

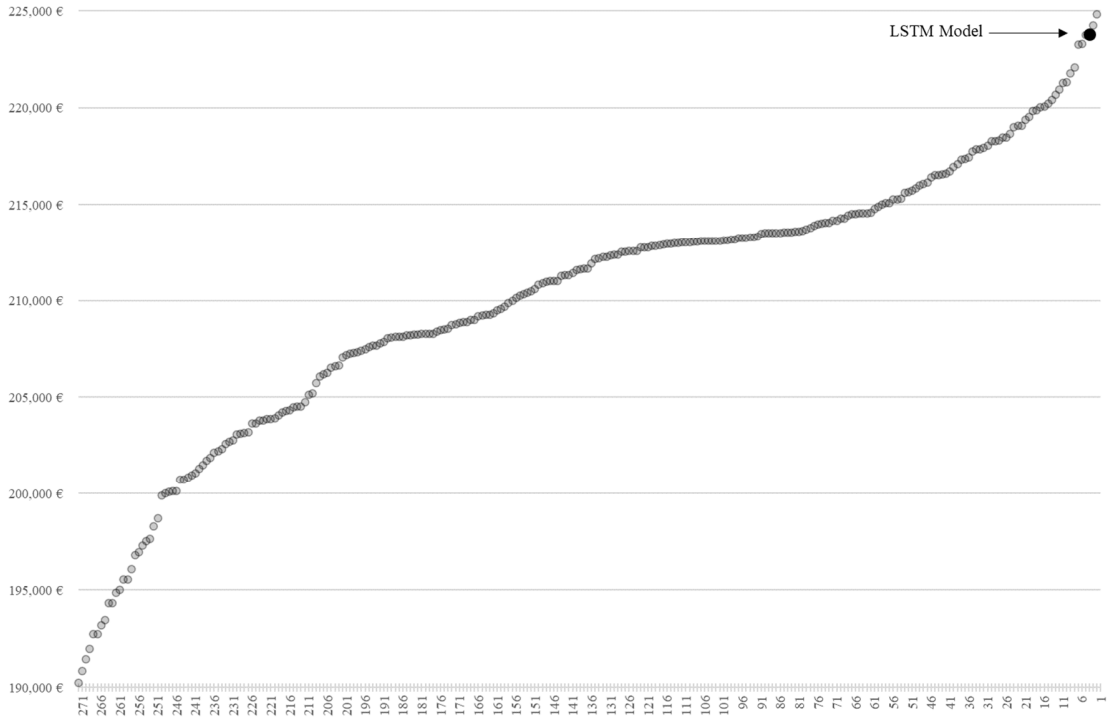

Figure 7 – Performance distribution (best trials) of the 271 entries in the predictive modeling competition, from worst (left) to best (right) performance. The LSTM model, which does not rely on advanced feature engineering, and instead uses raw data to make its predictions, achieved third place.

APPLICATIONS OF LSTM NEURAL NETWORKS IN MARKETING

Though we set our studies in a direct marketing context, LSTM neural networks can provide a solution to the general class of prediction tasks that involve panel data. We foresee that, since panel data is ubiquitous in marketing, LSTM neural networks can find widespread applications in marketing academia and practice. We discuss some possible applications below.

Brand choice and market share forecasting using scanner data

Demand forecasting for products within a category is a critical task for retailers and brand managers alike. The multinomial logit model (MNL) is commonly used to predict brand choice

and market share using marketing-mix and loyalty variables (Guadagni & Little 1983). Artificial feedforward neural networks (ANN) have also been shown to effectively predict household brand choices, as well as brand market shares (Agarwal & Schorling 1996). Since brand choices can be modeled as sequential choices, and data complexity increases exponentially with the number of brands (with interaction effects), LSTM neural networks offer suitable alternatives.

Similar to our studies, we could encode brand choices and the decision environment as we encoded solicitations and donations: as a multidimensional vector. We conjecture that testing the performance of LSTM neural networks in the context of brand choices would constitute an exciting replication area.

Churn prediction

Customer retention is crucial in numerous industries, such as telecom, credit cards, or online gambling. A comprehensive stream in the literature focuses on predicting customers who are likely to defect/churn (e.g., Coussement & De Bock, 2013; Ascarza et al., 2016). Churn prediction models can be 'single future period' or 'time-series' (Blattberg et al. 2008), where churn is predicted either for a specific period of interest (Lemmens & Croux, 2006; Neslin et al., 2006) or over an extended period and multiple steps (Blattberg et al. 2008). The LSTM neural network typology is well suited for modeling churn, especially in time-series format. However, its performance against standard churn prediction models remains an avenue for further research.

Clickstream data

Online retailers routinely use clickstream data to predict online customer behavior. These retailers observe the clickstream data from a panel of customers and use the history of customers' browsing behavior to make predictions about browsing behaviors, purchasing propensities, or consumer interests. Marketing academics have leveraged the clickstream data of a single website

to model the evolution of website-visit behavior (Moe and Fader 2004a) and purchase-conversion behavior (Moe and Fader 2004b). However, observing the clickstream data from a single website usually does not give a complete picture as customers often visit multiple websites while shopping. Park and Fader (2004) leveraged internet clickstream data from multiple websites, such that relevant information from one website could be used to explain behavior on the other. The LSTM neural network would be well suited for modeling online customer behavior across multiple websites since it can naturally capture inter-sequence and inter-temporal interactions from multiple streams of clickstream data without growing exponentially in complexity.

TECHNICAL CONSIDERATIONS

It would be presumptuous to claim that LSTM models offer an ideal, one-fit-all solution to panel data analytics. In particular, the analyst is invited to be mindful of the following challenges.

First, hyperparameter tuning is not a trivial task. While a simple grid search may be sufficient to achieve optimal performance, Bayesian optimization may be required on occasion.

Second, as in all deep learning models, overfitting is a constant concern. Many solutions have been proposed, and can even be combined together, such as early stopping, L1 regularization, L2 regularization, and dropout (Michelucci, 2018). Unfortunately, it is not clear that one approach will systematically provide optimal results, irrespective of the data structure or network architecture.

Third, while LSTM models offer a markedly improved solution to the problem of exploding gradients (over vanilla RNN models), they are not guaranteed to be shielded from it

entirely. Facing such an issue, the analyst might need to rely on computational tricks, such as gradient clipping (Bengio, 2012), gradient scaling, or batch normalization (Bjorck et al., 2018).

Finally, the field of deep learning in general, and recurrent neural networks, in particular, is evolving rapidly. Many alternative model specifications and network architectures offer the promises of improvements over vanilla LSTM models. They have already been proven superior in some domains. Such alternative specifications include Gated Recurrent Units, BiLSTM (Siami-Namini et al., 2019), Multi-Dimensional LSTM (Graves and Schmidhuber 2009), Neural Turing Machines (Graves et al. 2014), Attention-Based RNN and its various implementations (e.g., Luong et al. 2015, Bahdanau et al. 2014), or Transformers (Vaswani et al. 2017). It is not clear that one architecture will lead systematically to the best possible performance. Lacking benchmarking studies, the analyst may be required to experiment with several models (although, as demonstrated in this paper, a simple LSTM model already provides excellent performance).

CONCLUSIONS

Ben Weber (2019) stated that "One of the biggest challenges in machine learning workflows is identifying which inputs in your data will provide the best signals [i.e., features] for training predictive models. For image data and other unstructured formats, deep learning models are showing large improvements over prior approaches, but for data already in structured formats, *the benefits are less obvious*" [italics added].

In this paper, we have shown that recent neural network architectures, traditionally used in natural language processing and machine translation, could effectively do away with the complicated and time-consuming step of feature engineering, even when applied to highly-structured problems such as predicting the future behaviors of a panel of customers. We apply the

LSTM neural networks to predict customer responses in direct marketing and discuss its possible application in other contexts within marketing, such as market-share forecasting using scanner data, churn prediction, or predictions using clickstream data.

Martinez et al. (2020) used 274 features to predict customer behaviors in a non-contractual setting. One of the authors, who has extensive industry experience, has built predictive models with 600 features and more. Feature engineering is not only a time-consuming process; it is also error-prone, complex, and highly dependent on the analyst's domain knowledge (or, sometimes, lack thereof). On the other hand, LSTM neural networks rely on raw un-summarized data to predict customer behaviors and can be scaled easily to very complex settings involving multiple streams of data.

Feature engineering is not an obsolete skill. When model explainability and controllability are important, a simpler model with well-crafted features may be best (De Bruyn et al., 2020), even at the expense of slightly-reduced model accuracy (Rudin, 2019).

We believe nonetheless that the ability of Recurrent Neural Networks (RNN), and specifically Long-Short Term Memory neural networks (LSTM), to dispense largely with this step, while achieving superior performance, is a noteworthy achievement that should resonate well with practitioners. This finding is especially relevant, knowing that data scientists spend about three-quarters of their time doing data-janitorial work – collecting, transforming, and cleaning data.

TABLES

Category	Features
Recency of donation (2)	Donors who gave recently are more likely to support further the organization than individuals who have lapsed for an extensive period. However, donors who donated barely a few weeks ago might not be ready to donate again in the immediate future. This inverted u-shape relationship between recency and loyalty is captured tentatively by a linear feature and its corresponding log-transform.
Frequency of donation (2)	Individuals who donated many times in the past are more likely to remain loyal in the future. The relationship between donation frequency and loyalty is, however, complex and nonlinear. Fundraising managers confirmed that going from one to two donations is a giant leap, whereas going from ten to eleven is less meaningful. The benchmark models capture the nonlinearity between frequency and loyalty by a linear feature and its corresponding log-transform
Recency of solicitation (2)	Solicitations tend to quickly generate a peak in donations, followed by a long tail (Basu et al., 1995). To capture this nonlinear relationship between the sending of a solicitation and its responses, we include the recency of the latest solicitation sent to the donor, as well as its log-transform, as model features.
Frequency of solicitations (2)	The number of solicitations sent to a donor is a strong indicator of how confident the organization is in the generosity potential of that individual. Many accumulated solicitations also communicate the organization's pressing needs to the donor base, and therefore may positively influence their decisions to support the organization in the future. Therefore, we introduce both the absolute number of solicitations sent since the beginning –and its log-transform– as model features
Month (11)	Some months (e.g., November, December, January) are more favorable to fundraising activities than others (e.g., February, August). We encode each month (minus one for identification purposes) as dummy variables
Donation same month last year (1)	One of the strongest predictors available in this context is whether a donor has donated during the same period a year prior. Some donors have idiosyncratic donation patterns that this feature captures well.
Solicitation same month last year (1)	For completeness, we also capture whether the donor has been solicited in the same period a year prior
Average time between donations (1)	The average time between donations can be used to capture changes in the frequency of donation over time.
Average time between solicitations (1)	The average time between solicitations can be used to capture changes in the frequency of solicitation over time.
Dummy for missing solicitation recency (1)	When the customer is newly acquired and has not received any solicitations, the recency of solicitation cannot be calculated; the average time between solicitations either. We replace these invalid values with a dummy to avoid any confounding effects

Table 1 – Description of the features used in the first empirical application. The numbers in brackets indicate the exact number of features per indicator.

	Lift 1%	Lift 5%	Lift 10%	Lift 20%
Logit	11.460	5.328	3.951	2.731
Random Forest (RF)	13.053	5.417	3.626	2.434
LSTM	13.903	6.479	4.305	2.778
LSTM vs. Logit	+23.36%	+26.60%	+12.00%	+2.75%
LSTM vs. RF	+7.06%	+24.05%	+25.88%	+24.03%

Table 2 – Average lift at 1%, 5%, 10%, and 20% for the LSTM model and the benchmark models (logit and random forest) over 12 independent predictive tasks (one for each month in the holdout dataset). On average, the LSTM model achieves a +26.60% (resp., +24.05%) improvement in the lift at 5% compared to the logit (resp., random forest) model.

	RMSE	LogLoss	Precision	Recall	F-measure
Logit	0.198	0.167	0.632	0.068	0.122
Random Forest (RF)	0.197	0.286	0.667	0.095	0.165
LSTM	0.195	0.164	0.710	0.103	0.175
Logit vs. LSTM	+1.3%	+2.1%	*A positive value indicates that RMSE, Logloss errors are more substantial with the benchmark models*		
RF vs. LSTM	+1.1%	+74.9%			
LSTM vs. Logit	*A positive value indicates that LSTM achieves superior performance*		+12.3%	+51.5%	+43.4%
LSTM vs. RF			+6.4%	+8.4%	+6.1%

Table 3 – Average root Mean Squared Error (RMSE), LogLoss, Precision, Recall, and F-measure for the LSTM model and the benchmark models (logit and random forest) over 12 predictive tasks (one for each month). To calculate the measures of Precision, Recall, and F-measure, we label predictions < 0.5 as 'negative' and ≥ 0.5 as 'positive'. The LSTM model outperforms the benchmark models across the board.

Individual	**Observed donation** (actual data)	**Solicitation** (contestant's decisions)	**Marketing cost**	**Gross revenue**
1	0 €	0	-	-
2	0 €	0	-	-
3	10 €	0	-	-
4	0 €	0	-	-
5	0 €	0	-	-
6	0 €	1	2 €	0 €
7	50 €	1	2 €	50 €
8	0 €	0	-	-
9	150 €	1	2 €	150 €
10	0 €	1	2 €	0 €

Table 4 – Observed donations for 10 individuals in the holdout sample (2nd column), hypothetical contestant's decisions to solicit these individuals or not (3rd column), and simulated marketing costs and gross revenue for that campaign (4th and 5th columns). Note that the third individual (in gray) gave 10 € in the actual campaign, but her donation is not added to the contestant's financial results due to the absence of solicitation.

Category	Features
Recency	Date of last donation. Common variants included mathematical transformations (log-transform, square root, power) and channel-specific recency (last online donation, last offline donation.)
Frequency	Number of donations. Common variants included mathematical transformations (log-transform, square root, power), channel-specific frequencies (number of online donations, number of offline donations), and time frames (e.g., number of donations over the last *N* years.)
Seniority	Date of first donation, i.e., date of donor acquisition. Common variants included mathematical transformations (log-transform, square root, power) and binary indicators indicating the channel (online, offline) and the type of first donation (one-off donation, automatic deduction.)
Interactions	Interaction terms between the aforementionned indicators, most commonly recency × frequency.
Amount	Donation amount. Common variants included moments (minimum, maximum, average, median amounts), channel (online vs. offline), mode of payment (e.g., check, credit card), and period under consideration (e.g., whole history of donations, *N* most recent donations, *N* most recent years). Some students experimented with exponentially-weighted moving averages.
Time gaps	Gaps between observed donations (or solicitations). Common variants include moments (minimum, maximum, average, median time gaps) and channel (online vs. offline.)
Time gap × recency	Several students (and the instructor) included a relative measure of time gap compared to contact's recency. For instance, an average time gap between donations of 300 days, and recency of 150 days, gives a ratio of 0.5. A ratio close to 1 indicates perfect timing for a solicitation. A value above 1 indicates the donor might have churned. Variants included the introduction of standard deviations in the computations (confidence interval) and mathematical transformations (log-transform, square root.)
Seasonality	The campaign of interest was launched in late June, which is an unusual timing. Additional features included the likelihood of making a donation during the summer, the likelihood of responding to past campaigns launched in June, etc.
Response rate	Ratios of donations vs. solicitations, either over the entire contact history, over a recent period (e.g., three years), or over specific solicitations (e.g., offline solicitations, solicitations sent in June, etc.)
Demographics	Prefix, ZIP codes (e.g., binary indicators for each of the most common ZIP codes in the contact list), departments (the equivalent of States, inferred from ZIP codes), etc.
Advanced demographics	Some contestants linked the donors' ZIP codes to publicly-available Census bureau data to infer education attainment, income, number of children, age, etc., or have linked donors' first names to the average age pyramid of said first names in the population to infer donors' age.
Automatic deductions	While the fundraising campaign was targeting one-off donations, the fact that some solicited contacts were already under automatic deductions (i.e., monthly donations) was informative. Common features included binary indicators (e.g., contact currently or previously under automatic deductions), as well as automatic deduction recency, frequency, seniority, and amounts.

Table 5 – Description of the most common features used by the 269 contestants. The number of features ranged from six to several hundred.

REFERENCES

Agrawal, D., & Schorling, C. (1996). Market share forecasting: An empirical comparison of artificial neural networks and multinomial logit model. *Journal of Retailing*, *72*(4), 383-408.

Ascarza, E., Iyengar, R., & Schleicher, M. (2016). The perils of proactive churn prevention using plan recommendations: Evidence from a field experiment. *Journal of Marketing Research*, *53*(1), 46-60.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Basu, Amiya K., Atasi Basu, and Rajeev Batra (1995). Modeling the response pattern to direct marketing campaigns. Journal of Marketing Research, 32(2), 204-212.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2), 157-166.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. Neural networks: Tricks of the trade (pp. 437-478). Springer, Berlin, Heidelberg.

Bengio Y., and R. Pascanu (2012). On the difficulty of training recurrent neural networks. https://arxiv.org/abs/1211.5063.

Bitran, G. R., and Mondschein, S. V. (1996). Mailing decisions in the catalog sales industry. Management Science, 42(9), 1364-1381.

Bjorck, Johan, Carla Gomes, Bart Selman, and Kilian Q. Weinberger (2018). Understanding Batch Normalization. 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

Blattberg, R. C., Byung-Do Kim, and Scott A. Neslin (2008), Database Marketing: Analyzing and Managing Customers. International Series in Quantitative Marketing.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5-32.

Brown, A. D., & Hinton, G. E. (2001). Products of Hidden Markov Models. In *AISTATS*.

Bull, A. D. (2011). Convergence rates of efficient global optimization algorithms. Journal of Machine Learning Research, 12(Oct), 2879-2904.

Bult, J. R., and Wansbeek, T. (1995). Optimal selection for direct mail. Marketing Science, 14(4), 378-394.

Colombo, R., and Jiang, W. (1999). A stochastic RFM model. Journal of Interactive Marketing, 13(3), 2-12.

Cotter, A., Shamir, O., Srebro, N., and Sridharan, K. (2011). Better mini-batch algorithms via accelerated gradient methods. In Advances in neural information processing systems (pp. 1647-1655).

Coussement, K., & De Bock, K. W. (2013). Customer churn prediction in the online gambling industry: The beneficial effect of ensemble learning. *Journal of Business Research*, *66*(9), 1629-1636.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Chollet, F. (2018). Deep Learning with Python. Manning Publications

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

De Bruyn, Arnaud, Vijay Viswanathan, Yean Shan Beh, Jürgen Kai-Uwe Brock, and Florian von Wangenheim (2020), "Artificial Intelligence and Marketing: Pitfalls and Opportunities," Journal of Interactive Marketing.

Dong, G., and Liu, H. (2018). Feature engineering for machine learning and data analytics. CRC Press.

Donkers, B., Paap, R., Jonker, J. J., and Franses, P. H. (2006). Deriving target selection rules from endogenously selected samples. Journal of Applied Econometrics, 21(5), 549-562.

Elsner, R., Krafft, M., and Huchzermeier, A. (2004). Optimizing Rhenania's direct marketing business through dynamic multilevel modeling (DMLM) in a multicatalog-brand environment. Marketing Science, 192-206.

Olah, C. (2015). Understanding LSTM networks.

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012). How many trees in a random forest?. In *International workshop on machine learning and data mining in pattern recognition* (pp. 154-168). Springer, Berlin, Heidelberg.

Fader, P. S., Hardie, B. G., and Lee, K. L. (2005). RFM and CLV: Using iso-value curves for customer base analysis. Journal of marketing research, 42(4), 415-430.

Friedman, J., Hastie, T., and Tibshirani, R. (2009). glmnet: Lasso and elastic-net regularized generalized linear models. R package version, 1(4).

Gal, Y., & Ghahramani, Z. (2016, June). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In international conference on machine learning (pp. 1050-1059).

George, M., Kumar, V., and Grewal, D. (2013). Maximizing profits for a multi-category catalog retailer. Journal of Retailing, 89(4), 374-396.

Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). Learning to forget: Continual prediction with LSTM.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. MIT press.

Gönül, F. F., and Hofstede, F. T. (2006). How to compute optimal catalog mailing decisions. Marketing Science, 25(1), 65-74.

Gönül, F. F., Kim, B. D., and Shi, M. (2000). Mailing smarter to catalog customers. Journal of Interactive Marketing, 14(2), 2-16.

Gönül, F., and Shi, M. Z. (1998). Optimal mailing of catalogs: A new methodology using estimable structural dynamic programming models. Management Science, 44(9), 1249-1262.

Graves, A., Wayne, G., & Danihelka, I. (2014). Neural turing machines. arXiv preprint arXiv:1410.5401.

Graves, A., and Schmidhuber J. (2008). "Offline handwriting recognition with multidimensional recurrent neural networks". Advances in Neural Information Processing Systems 21 (NIPS 2008).

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2016). LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10), 2222-2232.

Guadagni, P. M., & Little, J. D. (1983). A logit model of brand choice calibrated on scanner data. *Marketing science*, *2*(3), 203-238.

Hinton, Geoffrey (2013), "Advanced Machine Learning: Recurrent neural networks," available at https://www.cs.toronto.edu/~hinton/csc2535/notes/lec10new.pdf

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Kingma, D. P., and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kuhn, M., and Johnson, K. (2013). Applied predictive modeling (Vol. 26). New York: Springer.

Kuhn, M., and Johnson, K. (2019). Feature Engineering and Selection: A Practical Approach for Predictive Models. Chapman & Hall/CRC Data Science Series.

Lemmens, A., & Croux, C. (2006). Bagging and boosting classification trees to predict churn. *Journal of Marketing Research*, *43*(2), 276-286.

Lilien, G. L., A. Rangaswamy, and A. De Bruyn (2013). Principles of Marketing Engineering and Analytics, 3rd edition. DecisionPro.

Ling, C. X., and Li, C. (1998, August). Data mining for direct marketing: Problems and solutions. In KDD (Vol. 98, pp. 73-79).

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

Malthouse, E. C. (1999). Ridge regression and direct marketing scoring models. *Journal of interactive marketing*, *13*(4), 10-23.

Martínez, Andrés, Claudia Schmuck, Sergiy Pereverzyev Jr., Clemens Pirker, and Markus Haltmeier (2020), "A machine learning framework for customer purchase prediction in the non-contractual setting," European Journal of Operations Research, 281, pp.588-596.

Michelucci, Umberto (2018). Applied Deep Learning: A Case-Based Approach to Understanding Deep Neural Networks. Apress.

Ming, Y., Cao, S., Zhang, R., Li, Z., Chen, Y., Song, Y., & Qu, H. (2017, October). Understanding hidden memories of recurrent neural networks. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)* (pp. 13-24). IEEE.

Moe, W. W., & Fader, P. S. (2004a). Capturing evolving visit behavior in clickstream data. *Journal of Interactive Marketing*, *18*(1), 5-19.

Moe, W. W., & Fader, P. S. (2004b). Dynamic conversion behavior at e-commerce sites. *Management Science*, *50*(3), 326-335.

Neslin, S. A., Gupta, S., Kamakura, W., Lu, J., & Mason, C. H. (2006). Defection detection: Measuring and understanding the predictive accuracy of customer churn models. *Journal of marketing research*, *43*(2), 204-211.

Netzer, O., Lattin, J. M., & Srinivasan, V. (2008). A hidden Markov model of customer relationship dynamics. *Marketing science*, *27*(2), 185-204.

Park, Y. H., and Fader, Peter S. (2004), "Modeling browsing behavior at multiple websites," *Marketing Science*, 23(3), 280-303.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013), "On the difficulty of training recurrent neural networks. In International conference on machine learning," Proceedings of the 30th

International Conference on International Conference on Machine Learning, Volume 28, June 2013 1310-1318.

Pointer, Ian (2019), "Programming PyTorch for Deep Learning: Creating and Deploying Deep Learning Applications," O'Reilly Media.

Roberts, M. L., and Berger, P. D. (1999). Direct marketing management. Prentice Hall International (UK).

Rudin, Cynthia (2019), "Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead," Nature Machine Intelligence, Vol 1, 206-215.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.

Saleh, H. (2018). Machine Learning Fundamentals: Use Python and scikit-learn to get up and running with the hottest developments in machine learning. Packt Publishing Ltd.

Schweidel, D. A., and Knox, G. (2013). Incorporating direct marketing activity into latent attrition models. Marketing Science, 32(3), 471-487.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. (2015). Taking the human out of the loop: A review of Bayesian optimization. Proceedings of the IEEE, 104(1), 148-175.

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2019, December). The performance of LSTM and BiLSTM in forecasting time series. In 2019 IEEE International Conference on Big Data (Big Data) (pp. 3285-3292). IEEE.

Simester, D. I., Sun, P., and Tsitsiklis, J. N. (2006). Dynamic catalog mailing policies. Management science, 52(5), 683-696.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In Advances in neural information processing systems (pp. 2951-2959).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

Stein, M. L. (2012). Interpolation of spatial data: some theory for kriging. Springer Science & Business Media.

Tieleman, T., and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning, 4(2), 26-31.

Van Diepen, M., Donkers, B., and Franses, P. H. (2009). Dynamic and competitive effects of direct mailings: A charitable giving application. Journal of Marketing Research, 46(1), 120-133.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

Williams, C. K., and Rasmussen, C. E. (2006). Gaussian processes for machine learning (Vol. 2, No. 3, p. 4). Cambridge, MA: MIT Press.

Zheng, Alice, and Amanda Casari (2018). Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists. O'Reilly Media.

Zou, H., and Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the royal statistical society: series B (statistical methodology), 67(2), 301-320.