Master Thesis

Ecole Polytechnique Fédérale de Lausanne
School of Computer and Communication Sciences

# Forecasting Model for Digital Promotions

Electronic Arts

*Author:*
Oussama Abouzaid

*Supervisors:*
Pr. Rüdiger Urbanke
Adrien de Castro

March 6, 2020

**Acknowledgement**

I would like to thank Professor Rüdiger Urbanke for supervising me and granting me the opportunity to work on such an interesting topic, within a company that I have never thought I would be interning at. I would also like to thank my supervisor Adrien for the constant support and laughs during my time at EA; it has been a blast! Special thanks also to Matteo, Amin, Saeed & the Digital Team for their support and valuable feedback to improve my work.

Last but not least, I would like to express my deepest gratitude to my parents, my brother and my sisters for always supporting me and encouraging me throughout my journey at EPFL. This would not have been possible without your constant love and support.

*To my mother...*

# Table of contents

## List of Figures

## List of Tables

## Abstract

When promoting a product, it is crucial to be able to accurately estimate the volume of the sales generated by this promotion, at a specific time, which can allow the Marketing department to take actions accordingly. During the past 6 months, I have been working within the Data Science team at EA, in Marketing Analytics. In this thesis, we propose a forecasting model that predicts the uplift in sales during a promotion for a particular game and simulate the outcomes of multiple case scenarios of promotions using different discount rates, to recommend the best one.

# 1 Introduction

Although marketing strategies differ across companies around the world, the ultimate objective remains to have a step ahead of competitors. In 1960, Edmund Jerome McCarthy introduced the 4 Ps of the marketing mix: *product, price, place & promotion.* They cover the set of marketing tools used by a firm to reach its marketing objectives [1]. Throughout this work, we shall tackle price discounts, which will be referred to as **promotions**, being one of the main strategies aiming to increase sales. In our context, we will be interested in digital promotions in the gaming industry, which are promotions placed on digital channels, such as the PlayStation Store, the Microsoft Store and Origin (EA's digital distribution platform). Figure 1 shows an example for the Xbox One games catalog during sales.

The primary motivation behind this work is to be able to quantify the incremental response of the promotion compared to a non-promotion scenario, which is called the **uplift**. The goal of uplift modeling is to measure the incremental impact of promotions on games sales directly. Quantifying the uplift for specific promotion periods and games, and uncovering the KPIs behind would help in understanding previous promotion successes/failures. Furthermore, a good predictive model would allow for an automated and fast way to simulate multiple promotion scenarios and help in taking promotion strategy decisions.

In this thesis, we first introduce the dataset of promotions, present some predictive methods for uplift modeling along with results, interpretation, and output examples. We will afterward pick the best performing model and build a recommender system on top of it, which will allow for fine-tuning the discount of the promotion and measuring the response of the sales.

## 1.1 Related work

Even though uplift modeling is extensively applied in the context of marketing (amongst many other fields), it has not seen exceptional popularity in the literature. It is commonly categorized into 3 approaches. Firstly, the Two-Model approach which consists of training 2 models separately (for the treatment and the control groups). Secondly, the Class-transformation approach (introduced by Jaskowski and Jaroszewicz [2]). And lastly, the direct modeling of the uplift using Machine Learning approaches. Gutierrez & Gérardy compare the three approaches in their review of the uplift literature [3].

In this thesis, we tackle the last approach that consists of using a modified ML technique for uplift prediction. There have been a few different ML approaches in the past: In 2002, Victor S. Y. Lo [4] proposed a standard logistic regression to model the effect of the treatment group, which is, in our case, the promotion event. Su (2012) [5] and Guelman (2014) [6] proposed a k-nearest neighbor model. In 2013, Jaroszewicz and Zaniewicz [7] presented a SVM model.

However, the tree-based models remain the best performing models and the most popular in the literature. Radcliffe and Surry (2011) [8] presented a summary of some of the results that have been delivered using tree-based uplift modeling in practice, along with some examples from the industry.

## 1.2 Existing work

The model currently in production for promotion uplift prediction uses a traditional Random Forest. The error of the model in WMAPE[1] is  35% in terms of units predicted, using a 10-fold cross-validation. We propose in this thesis a novel approach for uplift prediction, that not only is more accurate than the existing model but also adds interpretability to the model. This new approach allows for recommending the best discount rate that leads to suggesting the uplift that maximizes the number of units sold. This approach uses a modified version of the known vanilla Random Forest and overcomes its known extrapolation limitations.

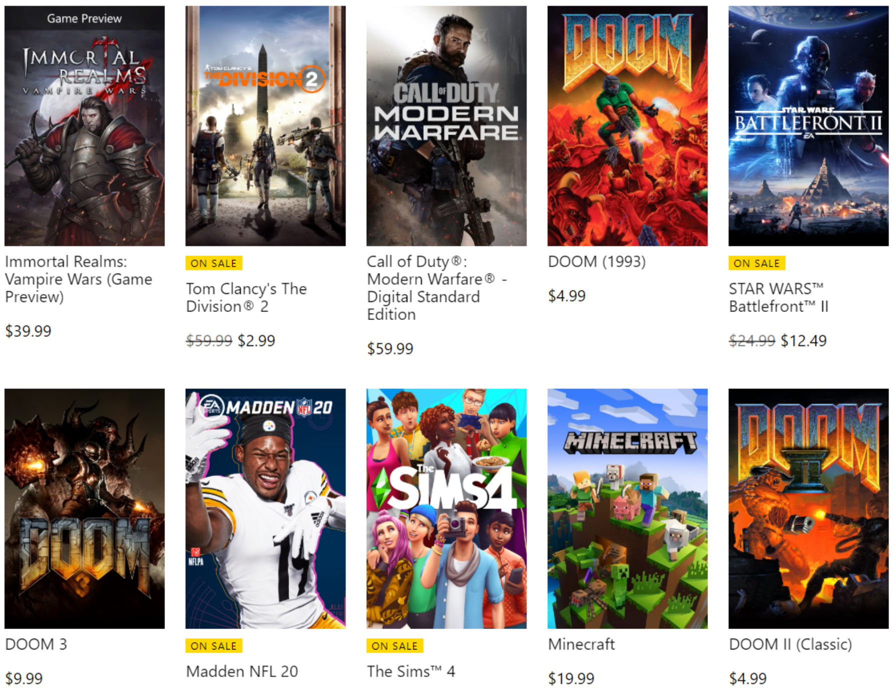


Figure 1: Games on sale in the Microsoft Store

[1]Weighted Mean Absolute Percentage Error, defined in Section 4.1

# 2 Data Definition

## 2.1 Dataset overview

The initial dataset contains 32'000+ records, where each record represents a promotion for a specific platform (Console or PC), country, and type of game (full game or an extension to the game). Promotions range from October 2018 until January 2020.

Below is a sample record from the data:

| startdate | enddate | platform | mastertitle | genre | itemtype | promotiontype | countrygroup | discount | promo_price | baseprice | **uplift** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2018-12-21 | 2019-01-03 | Xbox One | Sports game 2019 | Sports | Full Game | Countdown Sale | Nordic | 0.6 | 27.6 | 69.99 | **1.43** |

Table 1: Sample record in the dataset

This example shows a promotion of NHL 20 (Hockey game) in the Nordics (Sweden, Noway, and Finland) at 60% discount on Xbox One. Ultimately, the goal of the forecasting model is to predict the number of units sold.

## 2.2 Dataset description

Here is en exhaustive list of the columns present in the dataset, along with definitions:

- **startdate**: Start date of the promotion
- **enddate**: End date of the promotion
- **platform**: Console (PS4, Xbox One), or PC
- **mastertitle**: Name of the game, e.g. FIFA 20. Include both standard editions and deluxe editions (contains additional game features)
- **genre**: Genre category of the game (e.g. Sports, Adventure, Shooter, etc.)
- **itemtype**: Full Game or Extra-Content (adds-on/extensions to the game)
- **promotiontype**: Type of the promotion (e.g. Black Friday, Easter Sale, etc.)
- **countrygroup**: Country group, e.g. Americas (USA + Canada), UKI (UK + Ireland), France, etc.
- **discount**: Discount rate of the promotion, between 0 and 1
- **promotionprice**: Price during the promotion ($= baseprice \times (1 - discount)$)
- **baseprice**: Base price of the game, without discount
- **units** or **ST Units** (sell-through): Number of units sold, between **startdate** and **enddate**, both included.

# 3 Data augmentation and cleaning

In this section, we will first highlight some key assumptions that were taken in the process of cleaning and augmenting the dataset. We will then introduce some newly created features, in order to enrich the dataset, including the introduction of the response variable, that will be used later on for the prediction model.

## 3.1 Key assumptions

- ⇒ We reduce these features to contain only the **top 15 mastertitles** (80% of all sales), and **top 9 promotion types** (90% of all sales) in terms of overall total ST Units, and assign all the rest into a category **"other"**. This step is key to help reduce overfitting. (The initial dataset is very heterogeneous by nature, and some features such as **promotiontype** and **mastertitle** contain more than 175 and 37 unique categories, respectively)
- ⇒ There is a presence of 5000+ promotions where there is no sales ($units = 0$), which is noisy data and induces bias. These promotions are discarded.
- ⇒ Round the discount rate to the closest 5% to reduce the parameter space
- ⇒ Recompute the release date for some games since it is wrongly inputted in the past. The release date of a game is set to the first date at which the daily sales exceed the 15th percentile of the life-to-date sales, in order to capture the peak sales at the release date. The choice of this percentile is entirely empirical and showed good results in extracting actual release dates.

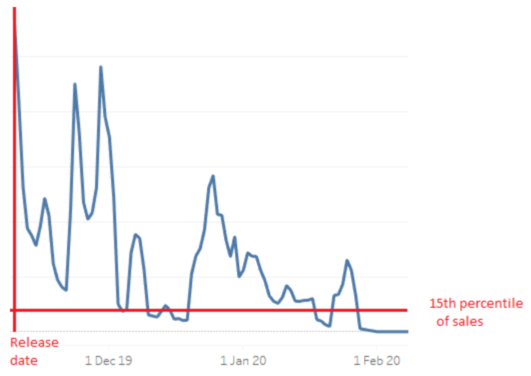


Figure 2: Release date extraction using 15th percentile of ST Units

## 3.2 Feature creation

After the data has been processed using the key assumptions, we create and added the following features to the dataset:

- **startdate_month** and **startdate_year**: month and year of the promotion
- **promotion_length**: the duration of the promotion in days
- **content**: distinguish standard editions from premium/deluxe editions.
- **n_promoted**: number of times the title has been promoted before in the past, in the current `countrygroup`
- **purchase_velocity**: number in days between the release_date of the title until the `startdate` of the promotion
- **pricedrop**: The price drop of the game, difference between base price and promotion price
- **sq_discount**: square of the discount
- **sq_pricedrop**: square of the price drop

## 3.3 Response variable creation

In order to quantify the impact that a promotion has, it is interesting to measure the **uplift** in sales, which measures the incremental impact of the sales during the promotion (treatment), as opposed to a period when the promotion is not active (control).

Before formally defining the uplift, let us define 2 more important features that will help us quantify it.

- **promo_daily**: average ST Units during a promotion
- **perc_25_baseline**: 25th percentile of daily ST units by (promotion, mastertitle,country_group), on a 30-day window prior to the start date of the promotion

The 25th percentile was chosen to discard all eventual peaks present in the data, which may occur due to external factors (other promotions at the same time, sport event, etc.)

We now define the uplift to be:

$$\textbf{uplift} = \frac{promo\_daily}{perc\_25\_baseline}$$

To give an intuition, the uplift measures the amplitude of the jump of sales that occurs on a promotion. In Figure 3, we can see a time Series of sales showing an uplift in sales during a promotion period (in red), compared to a period with no promotion

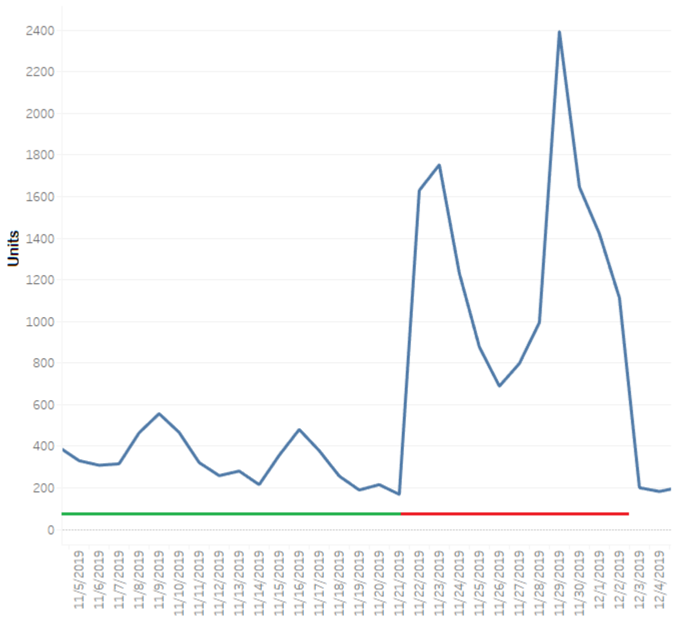


Figure 3: Time series showing the uplift during a promotion

(in green) The uplift will be the dependent variable to be predicted later on in the learning part.

Experience showed that predicting the uplift outperformed by far predicting the number of units, and this is due to the fact that information about sales before the promotion is also captured in our definition (green part).

It is important to note that for promotions in the future, **promo_daily** is calculated from the forecast provided by another department, as there we do not have the actuals yet.

## 4 Uplift Modeling

Once our dataset is cleaned and augmented with the features listed in Section 3, we proceed to predict the uplift. In this section, we will present some predictive approaches, along with their performances. We will also discuss the choice of the loss function to assess different models. Finally, uplift predictions from the best model will be illustrated for some key promotions and compared with actual uplifts.

Most of the implementation is done in Python using the `sklearn` module. Some code snippets are included at the end of this work.

### 4.1 Loss function

All the models presented in this work predict the uplift of a promotion. However, it is crucial for interpretability to measure the performance of our models using a more explicit quantity, such as the number of units sold (ST Units). **For this matter, the predicted uplift will be translated into units, and compared with the actual units to assess the performance.**

- Let $y_i$ be the predicted uplift for a promotion $i$, for $i \in \{1...n\}$, $n \in \mathbb{N}$, $n > 1$ We can extract the units predicted $u_i$ as:

$$u_i = \frac{exp(y_i) * perc\_25\_baseline}{promotion\_length}$$

- Let $\hat{u}_i$ be the actual units sold during promotion $i$

It is **very crucial** to chose an error metric that weights the amplitude of the error when computing the loss. **Example:** a 10% prediction error on a 60.000 unit promotion should have twice more weight than a 10% prediction error on a 30.000 unit promotion

$\Longrightarrow$ To address this issue, we use the **Weighted Mean Absolute Percentage Error (WMAPE)** to address this issue. The WMAPE is defined as follows:

$$WMAPE = \frac{\sum_{i=1}^{n} |u_i - \hat{u}_i|}{\sum_{i=1}^{n} \hat{u}_i}$$

## 4.2 Linear Regression

The first approach to model the uplift consisted of a simple Linear Regression. Let $X$ denote the predictor created in Section 3, including the augmented features. We use a one-hot-encoding for categorical variables, and we standardize the data by rescaling each feature to have a zero-mean and a unit variance distribution.

Let $\tilde{X}$ denote the resulting predictor, and $y$ denote the response variable, and $\beta$ the vector of the estimated coefficients.

### 4.2.1 Ridge regression

We use an ridge regression at first, with a penalty parameter $\lambda$. Let us recall that we aim to minimize the sum of squared errors, penalized by the sum of squares of the estimated coefficients:

$$\min_{\lambda}(y - \tilde{X}\beta)^T(y - \tilde{X}\beta) + \lambda\beta^T\beta$$

$\Rightarrow$ Grid search to find the best $\lambda^* = 3.2$

$\Rightarrow$ We use a 10-fold cross validation to measure the WMAPE = 63.9% (+/- 5.2% std. dev.)

### 4.2.2 Lasso regression

We now try the Lasso regression with a penalty parameter $\lambda$. The Lasso aims to minimize the sum of squared errors, penalized by the sum of absolute estimated coefficients:

$$\min_{\lambda}(y - \tilde{X}\beta)^T(y - \tilde{X}\beta) + \lambda\sum_{i}|\beta_i|$$

$\Rightarrow$ Again, we perform an exhaustive grid search to find the best $\lambda^* = 2.6$

$\Rightarrow$ We use a 10-fold cross validation to measure the WMAPE = 70.5% (+/- 2.8% std. dev.)

Both the errors obtained by the Ridge and the Lasso regressions are far from being good results, as the current tree-based model has WMAPE of  35%. In the next subsection, we will present a different uplift modeling approach based on decision trees, report their loss, and compare it to that of the linear regression.

## 4.3 Random Forest

### 4.3.1 Definition

Random forests were first introduced by Leo Breiman from the University of Berkeley in the early 2000s. They are learners that are a collection of decision tree predictors where each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest [11]. They have excellent performance in practice. One of their powerful strengths it that one can inspect features and their values corresponding to nodes and therefore make sense of what is going on. For regression, as it is the case for predicting the uplift, we fit the same regression tree many times to bootstrap-sampled versions of the training data, and average the result.[9][10][11]

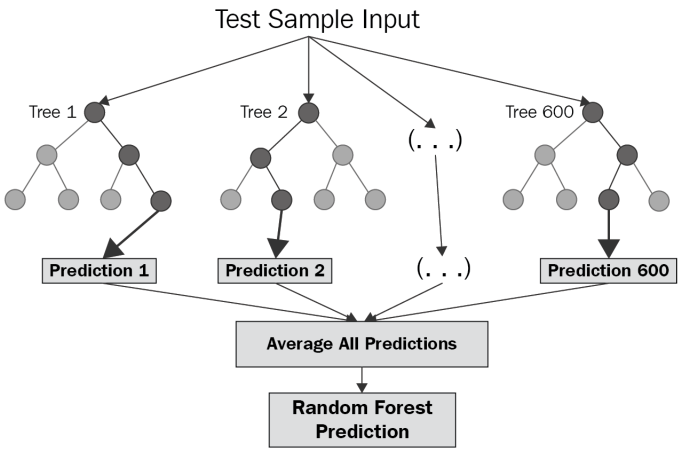


Figure 4: Random Forest regressor with $n = 600$ decision trees [12]

### 4.3.2 Vanilla Random Forest

In this subsection, we present a simple Random Forest approach for uplift prediction.

⇒ We use all the original features (Section 2.2) in addition to the newly created features (Section 3.2)

⇒ We use a 10-fold cross validation to measure the WMAPE

⇒ Parameters such as **max_depth** (maximum tree depth), **max_features** (Number of features to consider at every split), **min_samples_split** (The minimum number of samples required to split an internal node), **n_estimators** (The number of trees in the forest) are tuned using a grid search

⇒ We chose the set of parameters that minimize the mean of the 10 validation errors

## Results and model exploration

- **Best loss achieved:** Average WMAPE (10 folds) = 37% (+/− 2.7% std. dev.)
- **Best hyperparameters:** `max_depth` = 10, `n_estimators` = 500, `max_features` = 8, `min_samples_split` = 4
- **Feature importance:** `sklearn` computes a feature importance coefficient, which is the Gini coefficient. It is the number times a feature is used to split a node, weighted by the number of samples it split. Here are the most influential features in predicting the uplift:

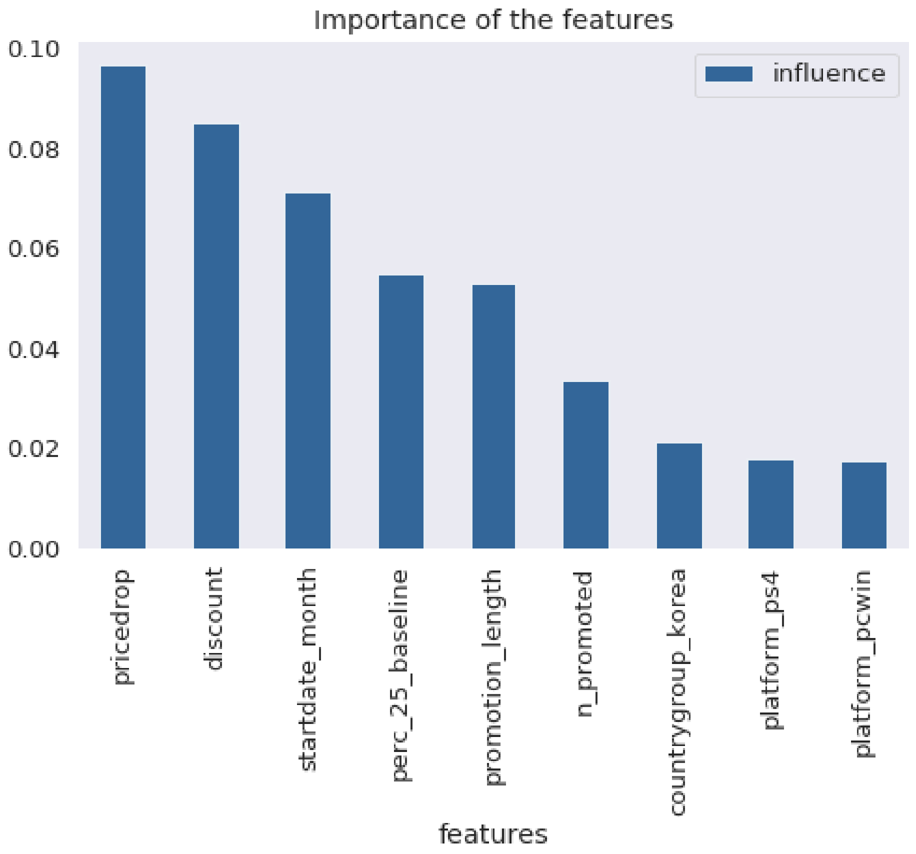


Figure 5: Importance of the features in predicting the uplift

We can see from Figure 5 that the most essential features that drive the uplift are the price drop of the game, the discount rate, and the month of the year, which is a desired result. In a theoretical world, the number of units sold during a promotion should rapidly increase with the discount rate. To get an intuition behind this, one can think of a discount of 5%, which would have a minimal impact as the price drop is very small. However, a 90% discount would generate much more sales as the promotion price significantly drops, which motivates the players to buy the games.

- **Prediction sample:** Below are some predicted uplifts (and units) for promotions of Black Friday 2018, for various games and platforms

| startdate | enddate | mastertitle | platform | countrygroup | discount | price | lift | pred_lift |
|---|---|---|---|---|---|---|---|---|
| 2018-12-18 | 2019-01-08 | sports game 2 | Console | america | 0.6 | 24 | 0.33 | **0.19** |
| 2018-11-20 | 2018-11-27 | role game | PC | america | 0.50 | 20 | 1.32 | **0.67** |
| 2018-11-16 | 2018-11-27 | shooter game | Console | america | 0.65 | 10 | 0.07 | **0.13** |
| 2018-11-16 | 2018-11-26 | sports game 1 | Console | uki | 0.45 | 44 | -0.45 | **-0.44** |
| 2018-11-20 | 2018-11-27 | role game | PC | america | 0.50 | 20 | 0.74 | **0.41** |
| 2018-11-20 | 2018-11-27 | shooter game 2 | PC | russia | 0.75 | 6 | -0.25 | **0.68** |
| 2018-11-16 | 2018-11-26 | sports game 1 | Console | mena & india | 0.45 | 44 | -0.25 | **-0.48** |
| 2018-11-16 | 2018-11-26 | sports game 1 | Console | pem | 0.45 | 44 | 0.70 | **-0.28** |
| 2018-11-16 | 2018-11-27 | fighting game | Console | america | 0.75 | 20 | 1.60 | **0.80** |

Table 2: Prediction examples of the RF model

- Table 2 shows output example of the RF model, with predicted lift and units in **bold**. We notice that in most of the cases, the predicted uplift is smaller than the actual uplift; hence the model tends to undershoot the target units sold. Let us take a closer at the predicted value and investigate what is causing this effect.

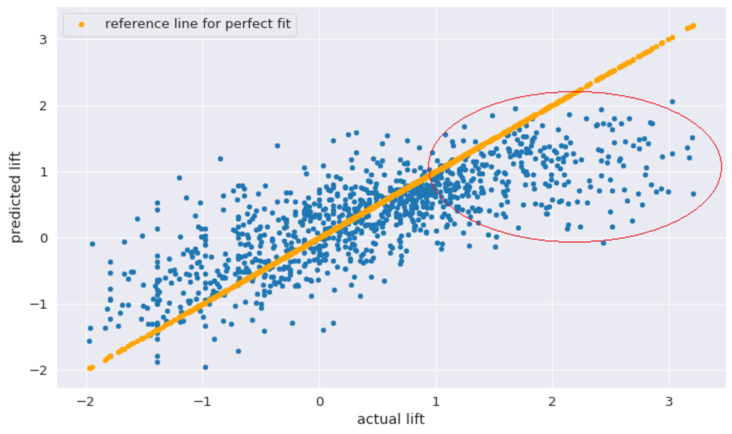


Figure 6: RF Model: Predicted uplifts against actual uplifts

We can observe from Figure 6 that the model is having trouble predicting uplifts greater than 1. That is mainly due to the fact that promotions with extreme uplifts are not very frequent in the data, and the model has a **week point** in extrapolation, which is a known pitfall of Random Forests. As was already mentioned in the definition 4.3.1, predictions from a Random Forest are obtained by averaging the results of each one of the underlying decision trees. Therefore, these predictions are always within the range of the uplift values present in the training dataset [13].

### 4.3.3 Regression Enhanced Random Forest

Results from the Vanilla Random Forest model showed that RFs suffered tremendously from the lack of extrapolation. In particular, the model was never able to predict an uplift outside of the range of that on the training set. To address this issue, an alternative model, Regression-Enhanced Random Forest[13] (RERF) is used.

RERF combines a penalized parametric regression and with a Random Forests, and gets the best out of both worlds. Parametric regressions such as Ridge can capture the linear relationship between the features and the response variable which allows for extrapolation outside of the training set boundaries.

The RERF algorithm [13] runs using the following steps:

- **Step 1:** Extend the predictor **X** to **X*** by adding the created features mentioned in Section 3.2.
- **Step 2:** Run Ridge of $Y$ (uplift) on $X^*$. Define the error $\mathbf{e} = \mathbf{Y}$ - $\mathbf{X^*}\beta_\lambda$ , where $\beta_\lambda$ is the estimated coefficient from Ridge, and $\lambda$ is the penalty parameter.
- **Step 3:** Build a Random Forest $\mathbf{T}_{m,s}$, of **e** on **X**. Final prediction of the uplift at value $X_0$ is then given by:
$$\hat{Y}(X_0) = X_0\hat{\beta}_\lambda + T_{m,s}(X_0)$$
- **Step 4:** Tune the parameters ($\lambda$, $s$, $m$) through K-fold cross validation by repeating Step 2 and Step 3.
  - $\lambda$ = Ridge penalty
  - $s$ = Minimum size of terminal nodes
  - $m$ = Number of variables randomly sampled as candidates at each split
- **Step 5:** Select the set of parameters that minimizes the WMAPE of the final predictions over the 10 folds.

⇒ Let $X$ be the predictor with the original features as defined in Section 2.2, and $X^*$ the augmented predictor using the features created in Section 3.2

⇒ As before, we use a 10-fold cross validation to measure the WMAPE

⇒ Grid search to tune the hyperparameters ($\lambda$, $s$, $m$). Values tried:
$\lambda \in \{0, 0.001, 0.05, 0.01, 0.05, 0.1, 0.5, 1, 2, 5\}$
$s \in \{1, 2, 3, 4, 5, 10, 15\}$
$m \in \{5, 10, 15, 20, 30, \text{None}\}$

**Results and model exploration**

- **Best loss achieved:** Average WMAPE (10 folds) = 36% (+/− 2.3% std. dev.)
- **Best hyperparameters:** $(\lambda, s, m) = (0.01, 20, 5)$
- **Feature importance:** The RERF model consists of a Ridge regression that extracts linear relationships between the features and the uplift, as well as a Random Forest that predicts the error of the Ridge model in predicting the uplift. Below are the coefficients of the Ridge regression and the Random Forest feature importance in predicting the error.

| **Feature i** | Coeff $\beta_i$ |
|---|---|
| discount | 0.97 |
| sq_discount | 0.86 |
| countrygroup_uki | 0.55 |
| promotiontype_black friday | 0.51 |
| platform_ps4 | 0.39 |
| franchise_fifa | 0.25 |
| ... | ... |
| itemtype_extra_content | -0.33 |
| countrygroup_china | -1.38 |

Table 3: Estimated coefficients of the Ridge regression (Step 2)

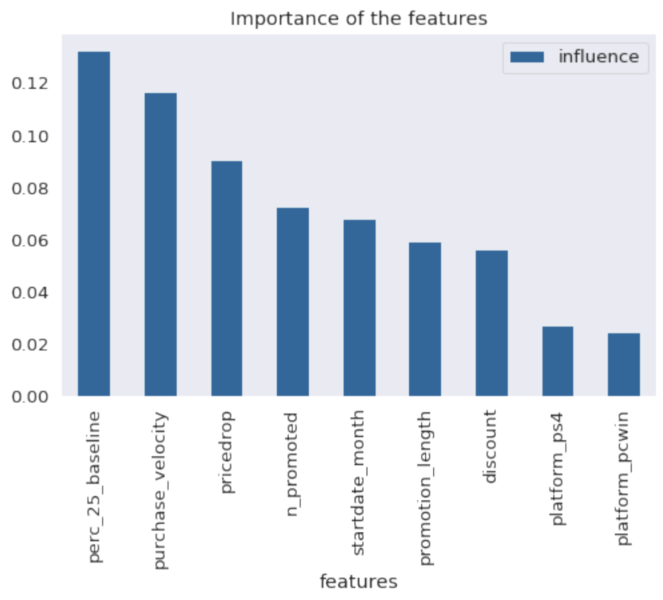


Figure 7: Importance of the features in predicting the uplift error (Step 3)

We can see from Table 3 that the discount and its square are the highest positive Ridge coefficients, which is again a desired result. Let us recall that the uplift should be strictly increasing as a function of the discount; the cheaper the game is, the more sales are generated, provided that all the other variables are fixed. We also notice that Ridge captured more information about the countries: UK promotions tend to have a higher uplift, as opposed to promotions in China that are usually characterized by a low uplift, which is confirmed by the coefficients. Also, we notice that the model assigned an important weight to Black Friday promotions, during which the highest uplifts are indeed achieved.

In Figure 7, we show the important features of the RF in predicting the uplift error, generated by the Ridge model. `perc_25_baseline` [2] and `purchase_velocity` [3] are important in predicting (and hence correcting) the Ridge regression error. An interesting thing to notice is that these features had an insignificant weight in the Ridge coefficient, which suggests that the RF is reincorporating these "overlooked" features to correct for the bias.

[2]Defined in Section 3.3: sales baseline for per promotion, game, platform and country. It is the 25th percentile of the daily sales on a 30-day window before the promotion

[3]Number of days since the release date of the game, at the start date of the promotion

- **Prediction sample:** Same as before, we show below some predicted uplifts (and units) for promotions during Black Friday 2019, for various platforms.

| startdate | enddate | mastertitle | platform | countrygroup | discount | price | lift | pred |
|---|---|---|---|---|---|---|---|---|
| 2019-11-22 | 2019-12-03 | sports game 1 | Console | america | 0.50 | 30 | 3.61 | **3.69** |
| 2019-11-21 | 2019-12-02 | sports game 2 | Console | america | 0.40 | 36 | 2.14 | **2.13** |
| 2019-11-22 | 2019-12-03 | sports game 2 | Console | america | 0.40 | 36 | 2.70 | **2.75** |
| 2019-11-21 | 2019-12-02 | racing game | Console | america | 0.35 | 39 | 0.78 | **1.06** |
| 2019-11-21 | 2019-12-02 | sports game 1 | PC | latam | 0.35 | 50 | 3.05 | **1.89** |
| 2019-11-21 | 2019-12-02 | sports game 1 | PC | america | 0.50 | 30 | 2.33 | **2.24** |
| 2019-11-22 | 2019-12-03 | role game | Console | america | 0.75 | 10 | 3.61 | **3.32** |
| 2019-11-26 | 2019-12-03 | role game | PC | america | 0.50 | 20 | 2.92 | **2.52** |

Table 4: Prediction examples of the RERF model

- Table 4 shows output example of the RERF model, with predicted lift and units in **bold**. The model seems to have overcome the problem of undershooting the target uplift, as the predicted uplifts are significantly close to the actuals. Let us take a closer look at the predicted uplifts output by the RERF model.

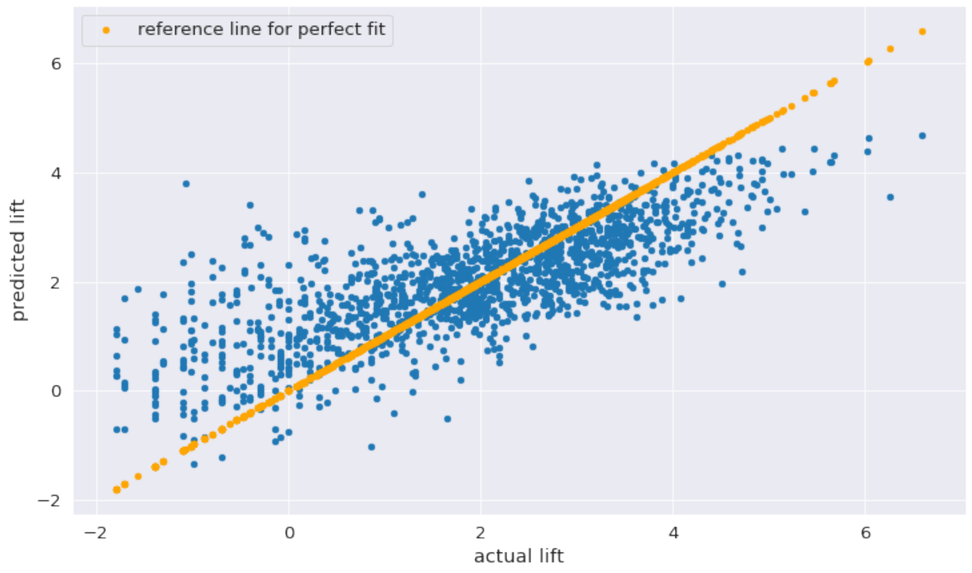


Figure 8: RERF Model: Predicted uplifts against actual uplifts

We can see from Figure 8 that there has been a significant improvement in forecasting the high lifts compared to the last RF model. The model still undershoots the target uplift for high values, but considerably less than the previous model. This suggests that this new RERF model has a better capacity to extrapolate values for the uplift outside of the range of the training set.

## 4.4 Results summary

| Model | WMAPE | Extrapolation |
|---|---|---|
| Ridge | 63.9% | Yes |
| Lasso | 70.5% | Yes |
| Vanilla RF | 37.4% | No |
| **RERF** | **36.1%** | **Yes** |

Table 5: Summary table of the WMAPEs of different models using a 10-fold CV

From what has been shown above, not only the RERF model has the lowest WMAPE among the 4 models, but it also bypasses the extrapolation limitation that the Vanilla RF suffers from. For the rest of the work, we will only consider the 2 RF-based models as they are the best-performing ones by far.

## 5 Recommender System

In this section, we build a recommender system on top of the model previously introduced. The idea behind this recommender system is to forecast for each promotion, different case scenarios by varying the discount rate from 10% to 80% with an increment of 5% and predicting the uplift (hence the units) generated by the promotion. For each promotion, this generates a curve of sales as a function of the discount rate. This curve will help the marketeer when deciding on the promotion strategy they shall opt for while taking into account the objective they are targeting (for instance, it could be the net revenue generated by the sales). In Table 6, the upper part above shows an existing promotion, which is extended by varying the discount rate and forecasting the uplift for each scenario, as shows the lower part of the table. Ultimately, we would like to be able to have a good estimate of the sales for a given (`promotion`, `mastertitle`, `platform`, `revenue_model`) tuple.[4] It is important to note that we are more interested in how a promotion performs globally rather than in a specific country. For this matter, we aggregate our predictions over the mentioned tuple (across countries) and measure the WMAPE of the aggregated predictions.

| promotion | mastertitle | platform | countrygroup | **discount** | pred_units | pred_lift |
|---|---|---|---|---|---|---|
| easter sale | Shooter game | PS4 | america | **0.3** | 120 | 0.4 |

$\downarrow$

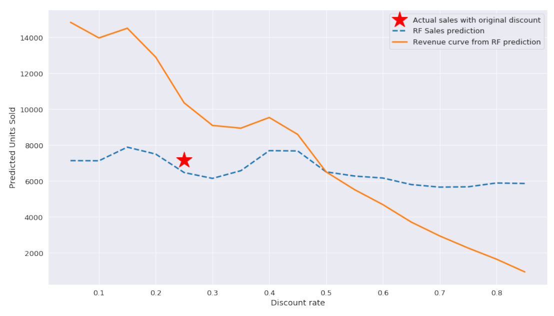
| promotion | mastertitle | platform | countrygroup | discount | pred_units | pred_lift |
|---|---|---|---|---|---|---|
| — ” — | —— ” —— | – ” – | —— ” —— | **0.1** | ??? | ??? |
| — ” — | —— ” —— | – ” – | —— ” —— | **0.15** | ??? | ??? |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| easter sale | Shooter game | PS4 | america | **0.3** | 120 | 0.4 |
| ⋮ | ⋮ | ⋮ | ⋮ | | ⋮ | ⋮ |
| — ” — | —— ” —— | – ” – | —— ” —— | **0.75** | ??? | ??? |
| — ” — | —— ” —— | – ” – | —— ” —— | **0.8** | ??? | ??? |

Table 6: Extending an existing promotion to multiple-scenario promotions

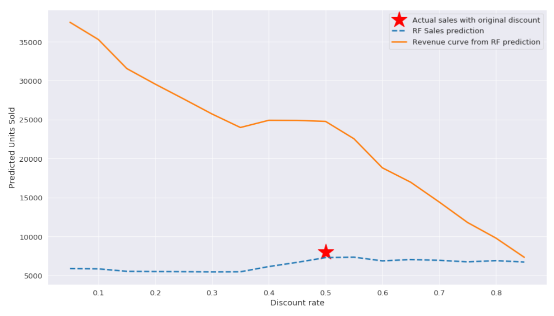
For both the Vanilla Random Forest, and the Regression-Enhanced Random Forest, we visualize predictions from tuning the discount rate for some key promotions, and inspect how sensitive is the model to the discount rate.

[4]See Section 2.2 for feature definitions.

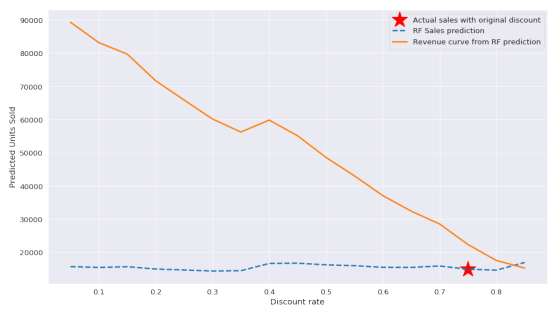
## 5.1 Vanilla Random Forest



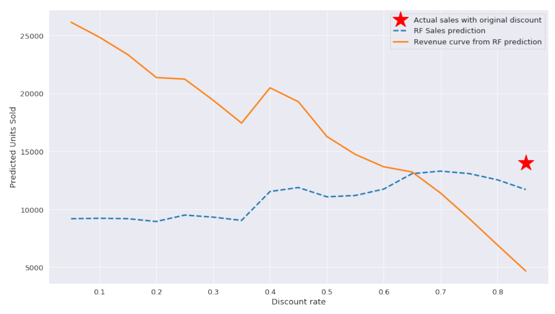
(a) RF output 1



(b) RF output 2



(c) RF output 3



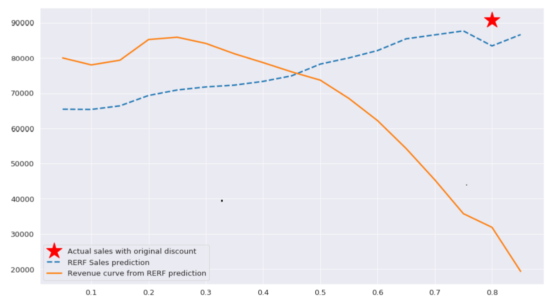
(d) RF output 4

Figure 9: RF sales prediction response to discount tuning for various promotions

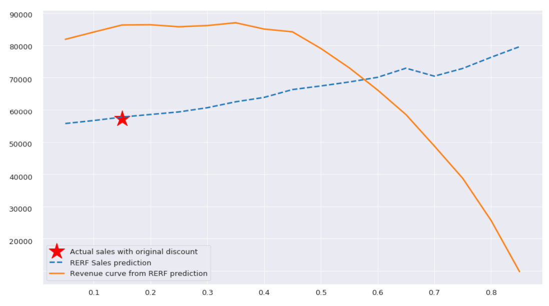
We can see from the output examples in Figure 9 that the RF predictions are rather flat and not very sensitive to the discount rate. However, one should expect the units sold to be an increasing function of the discount rate, as mentioned in Section 4. The RF model has never seen a discount lower than 30% hence always produces a flat prediction below this discount rate, which again an undesired result.

These examples reveal a major pitfall of the random forest, which is the lack of extrapolation when it comes to predicting values not already seen in training (synthesized promotions with hypothetical discount rates, except the discount in red star). On the other hand, as promotions of the same type tend to have more or less the same discount rates over time, the model outputs good predictions **only** for the actual discount rate (red star). As the predictions are flat, the revenue curve has a decreasing shape, meaning that we would expect more revenue when there is no promotion (discount rate = 0%) compared to a promotion period, which contradicts the reality. *Therefore, this model would tremendously fail as a recommender system for new discount rates.*

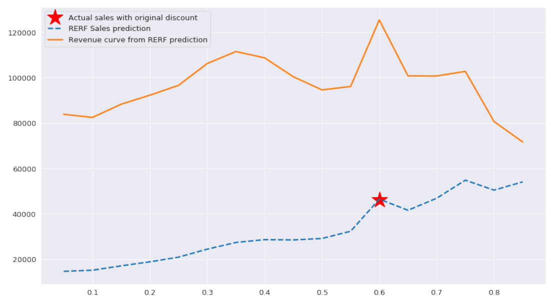
## 5.2 Regression Enhanced Random Forest



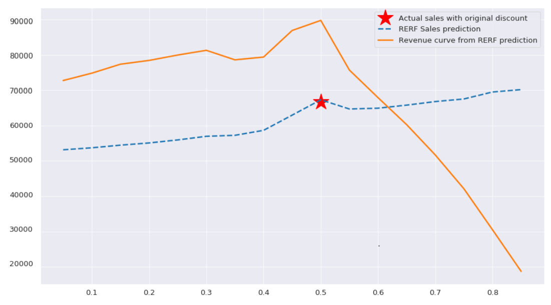
(a) RERF output 1



(b) RERF output 2



(c) RERF output 3



(d) RERF output 4

Figure 10: RERF response to discount tuning for various promotions

The predictions of the RERF above show more sensitivity to the discount rate than the previous Vanilla RF model. The response of the sales is an (almost) increasing function with the discount, which is a natural behavior. The model not only predicts well the promotions at their original discount but also "builds" a response curve whose slope dictates the sensitivity to the discount, which depends mainly on the game and the period of time, as seen earlier in Section 4.3.

Once again, the RERF model overcomes the Random Forest extrapolation weakness. Even for extreme discount values (10-20%, 70-80%), the response more reactive than the previous model, where the response for these extreme values was flat. Also, the maximum revenue is now achieved for different discount rates across the 4 examples. *Therefore, this model would be a good candidate for a discount rate recommender system.*

$\Longrightarrow$ The RERF model will, from now on, be carried out for the rest of the recommender system as it is the most suitable.

## 5.3 Aggregation and smoothing

In the previous section, we showed that the RERF model was able to produce some good predictions and meaningful extrapolation of the uplift (hence the sales). Let us recall that the recommender suggests discount rates for specific games and promotions globally, meaning that instead of looking at promotion uplift in a specific country, we are more interested in the global picture. For this matter, all the predictions output by the RERF are summed over the countries.

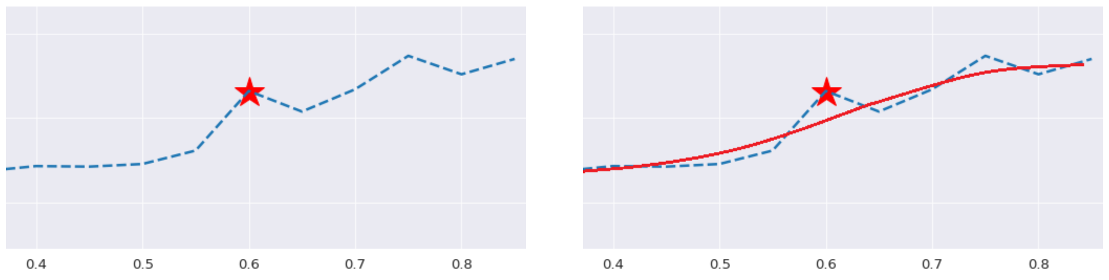
However, the response curves from Figure 10 show there is some noise in the prediction as the curves are not strictly increasing. For example, the predicted sales at a 60% discount rate are higher than at a 65% rate, which is implausible. To address this issue, a polynomial fit is applied to the raw aggregated RERF output and results in a monotonic function, as shown in Figure 11. More details about the smoothing function can be found in the appendix.

As a result of aggregation and smoothing function, a revenue curve is obtained by multiplying the number of units by the discounted price of the unit. As a summary, here are the steps to build the recommender:

1. Use the RERF model to forecast the uplift (hence the units) at a fine granularity (by country)

2. Group the prediction by game, type and promotion and sum over the units to obtain a coarser granularity

3. Apply the smoothing function to the resulting aggregation, gives an immediate optimal discount rate that optimizes a user-defined objective function $f(x)$, where $x$ is the discount rate



Raw units curve Smooth units curve

Figure 11: Smoothing the RERF units response curve as a function of the discount

## 5.4 Results and output

Let us recall from Section 4.3.3 that the RERF model had a WMAPE of 36% using a 10-fold cross-validation.

$\Longrightarrow$ After the country aggregation, the WMAPE drops to **22%** as errors of overshooting and undershooting of the uplift prediction cancel out. It is important to note that the prediction was made a fine granularity due to data scarcity, which easily leads to overfitting. The size of the aggregated dataset is roughly 10 times smaller than the original one.

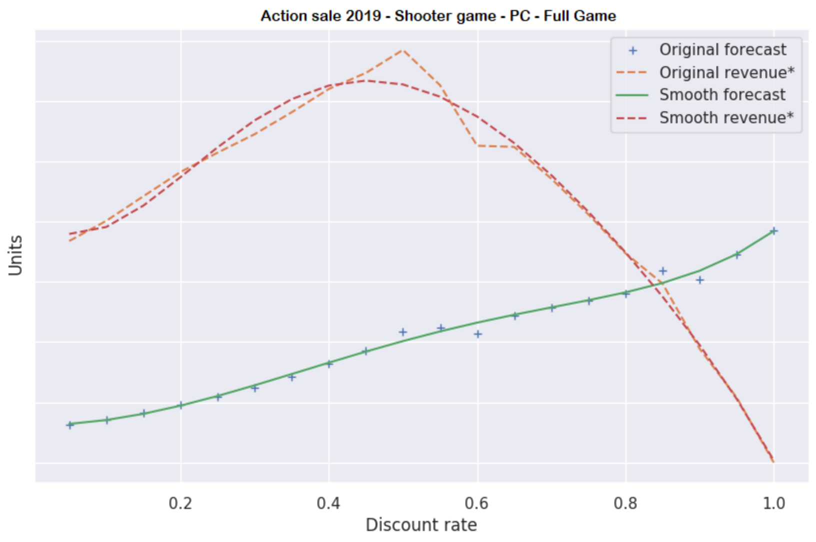


Figure 12: Shape of raw and smooth RERF example on a PC promotion

Figure 12 shows the shape of both raw and smooth predicted units and respective revenue, obtained using the smoothing function is depicted in Figure 11. The recommender displays the smooth green (units) and red (revenue) curves, from which the user can immediately read the optimal discount rate, which is achieved at 45% in this example. One of the main features of this recommender is that it adds interpretability of the discount rate. It measures its sensibility for a specific game on a specific platform and specific promotion.

To evaluate the quality of the recommended promotions, we sampled 600 promotions and took a closer look at the optimal recommended discounts. We can see in Figure 13 that the discounts recommended are often in the range 40-60%, and decrease towards the tails (<40% & >60%), which is a desired result as discounts become less and less efficient the smaller they get, and unprofitable if too high.

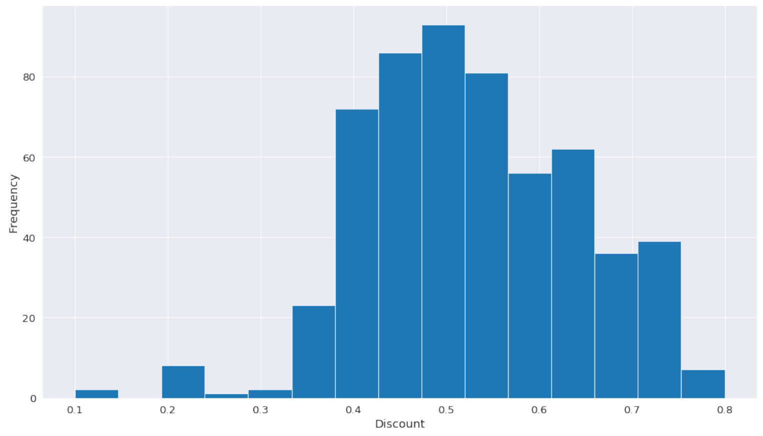


Figure 13: Distribution of the recommended optimal discount

## 6 Production

### 6.1 Pipeline automation

Once the recommender system is built, we automatize the whole data pipeline from the data collection to visualization through a dashboard. As shown in the diagram in Figure 14, here is a summary of the major steps taken along the process:

1. A script that refreshes the collections of the promotions by pulling data up to the current day is run at a daily basis.

2. The promotions collected are cleaned, the response variable, i.e. the uplift is created.

3. Prediction is made either using the last trained model or after retraining the model on the extended dataset, which is the user's choice through a flagged Python script. Usually, retraining should only be considered when the distribution of the data changes.

4. Predictions are aggregated at a global level, and the smoothing function is applied. The resulting curves are then exported and displayed on a dashboard.

We use **Jenkins** [17] for automation and continuous integration. Once the predictions are output by the script, they are exported to an internal folder using FTP. The dashboard queries these predictions to be displayed to the stakeholders (A prediction example is shown in Figure 15). The dashboard is coded in `R` using the `Shiny` package which is easy to use and offers flexibility and reactivity for integrating widgets.

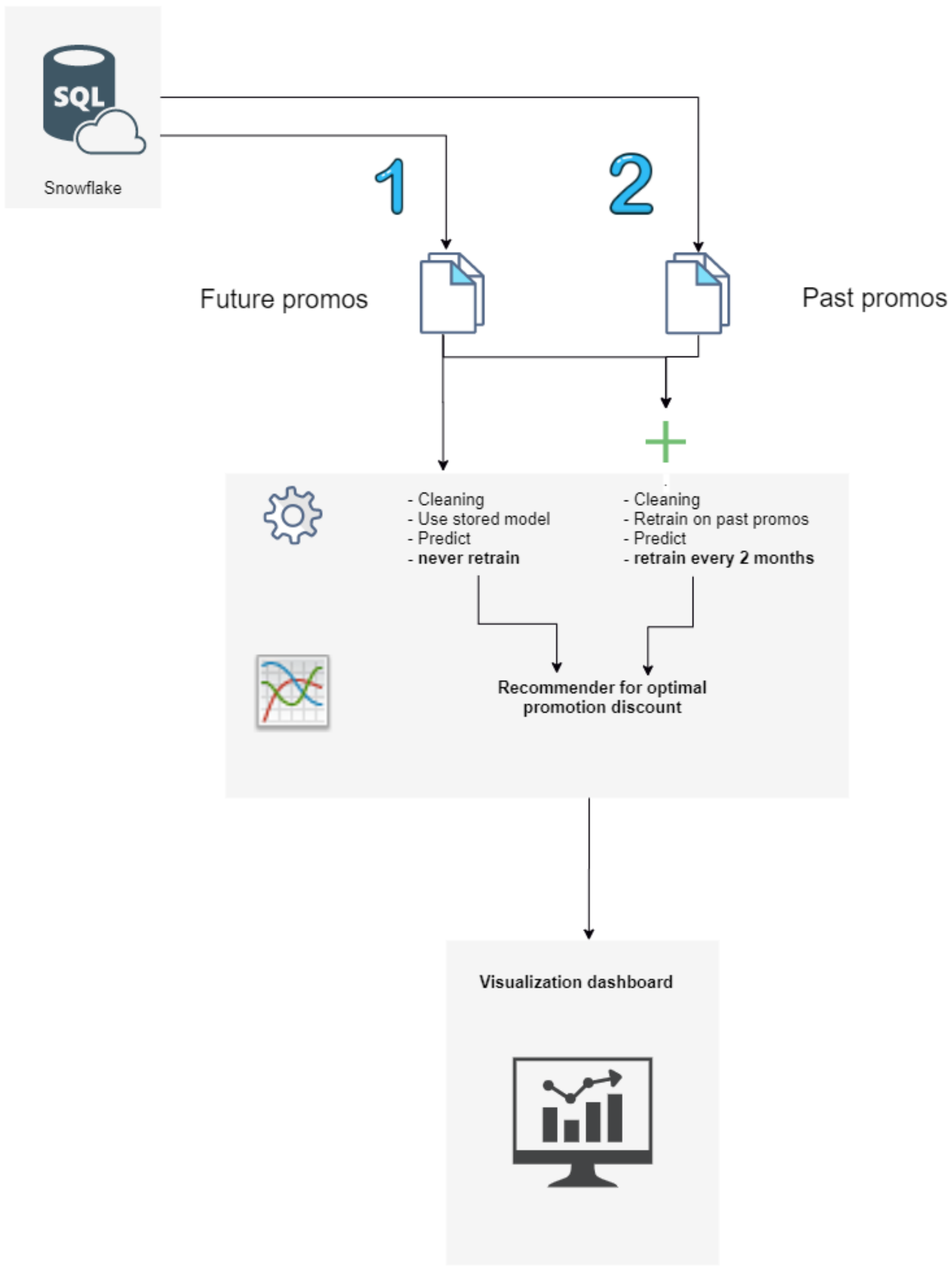


Figure 14: Model architecture in production

## 6.2 Dashboard for visualization

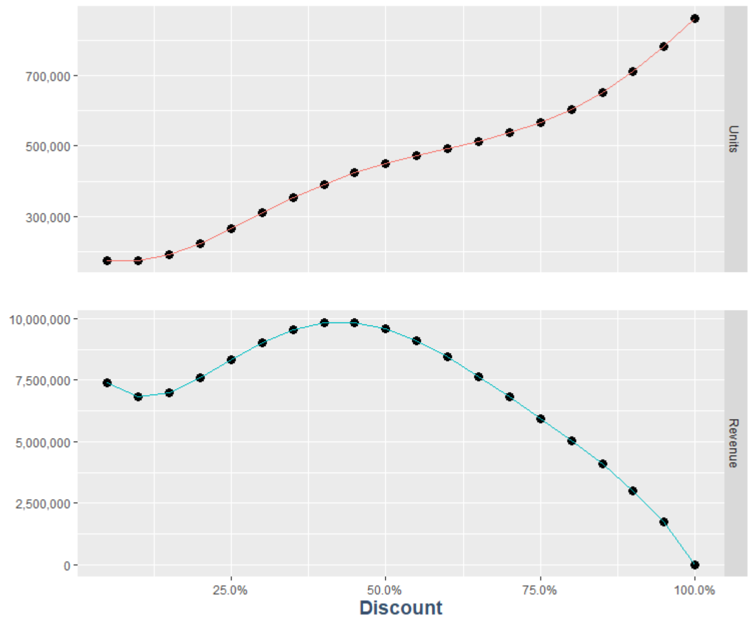
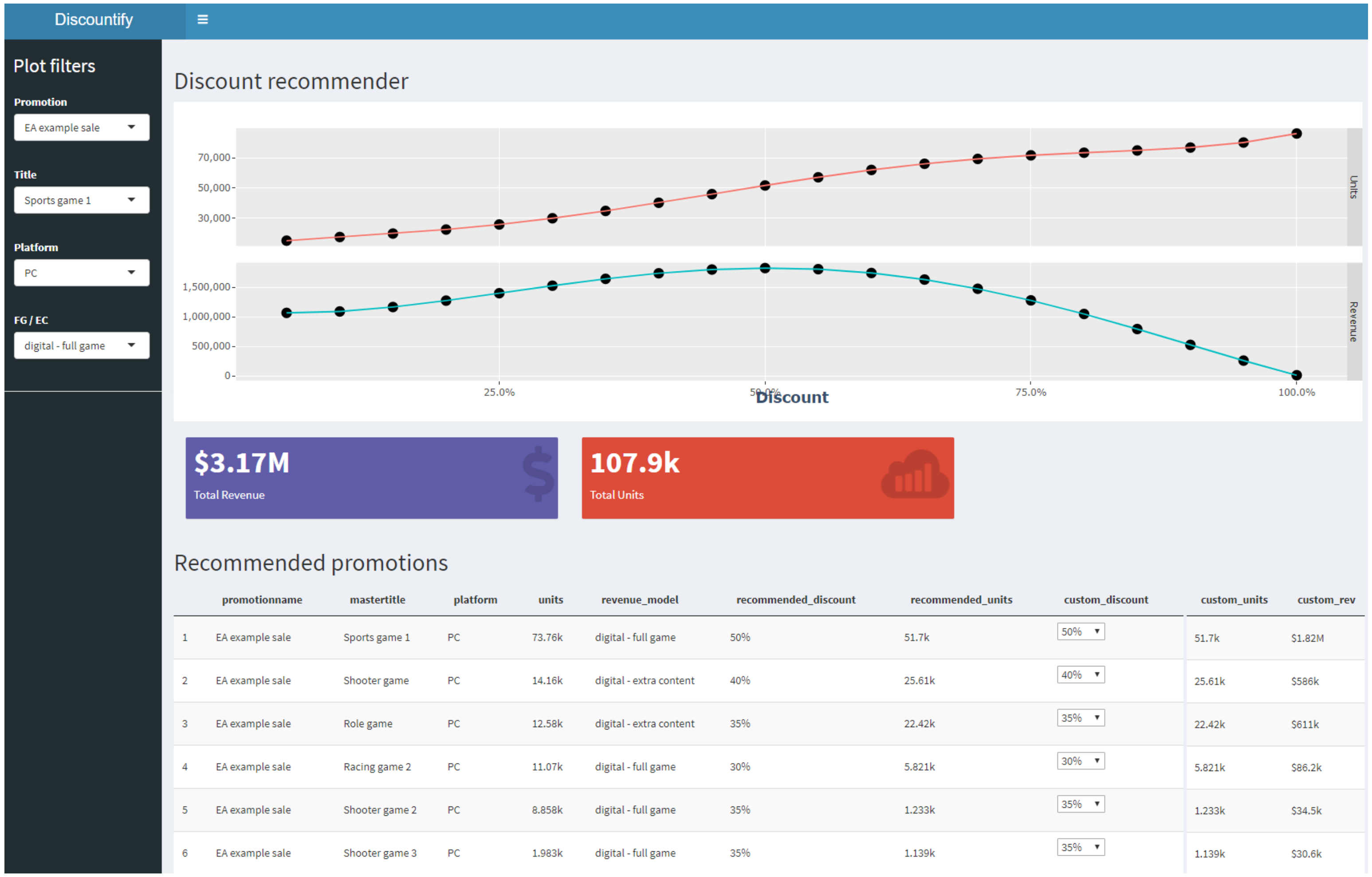


Figure 15: Recommender system output from the dashboard

Figure 15 shows the output of a recommendation for a 2-week promotion for a sports game in 2019. The forecasted units are displayed on top, and the corresponding revenue in the bottom, with a prediction for every 5% increment in the discount rate. The recommended optimal discount rate is the one that maximizes the revenue, and is 40% in this example, for $\sim$ 400'000 forecasted units to be sold.

We create a dashboard that outputs such a recommendation curve, for a selected **promotion**, **game**, **platform** & whether the item is a **full game** or **extra content**. As shown in Figure 16, each row represents a promotion with a widget to fine-tune the discount rate, which the user can play with and get a reactive response for the forecasted units and revenue.



Recommended promotions

| | promotionname | mastertitle | platform | units | revenue_model | recommended_discount | recommended_units | custom_discount | custom_units | custom_rev |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | EA example sale | Sports game 1 | PC | 73.76k | digital - full game | 50% | 51.7k | 50% | 51.7k | $1.82M |
| 2 | EA example sale | Shooter game | PC | 14.16k | digital - extra content | 40% | 25.61k | 40% | 25.61k | $586k |
| 3 | EA example sale | Role game | PC | 12.58k | digital - extra content | 35% | 22.42k | 35% | 22.42k | $611k |
| 4 | EA example sale | Racing game 2 | PC | 11.07k | digital - full game | 30% | 5.821k | 30% | 5.821k | $86.2k |
| 5 | EA example sale | Shooter game 2 | PC | 8.858k | digital - full game | 35% | 1.233k | 35% | 1.233k | $34.5k |
| 6 | EA example sale | Shooter game 3 | PC | 1.983k | digital - full game | 35% | 1.139k | 35% | 1.139k | $30.6k |

Figure 16: Dashboard screenshot

## 7 Conclusion

We proposed a novel uplift modeling approach for digital promotions that not only forecasts the sales in units for each planned promotion, but also extends them to multiple case scenarios by using various discount rates, and recommends the optimal one that maximizes revenue. We first started by applying some basic linear regression models followed by two tree-based models: RF and RERF regressors, which we mostly focus one since they outperformed the linear models by far. We have also seen that native tree-based models such as a Vanilla Random Forest suffer tremendously from the lack of extrapolation. Random Forests use a fully non-parametric predictive algorithm, and they may not efficiently incorporate known relationships between the response and the predictors, particularly between the discount rate and the uplift. To address this issue, we implemented a RERF [13] regressor that managed to overcome this limitation, and accurately predicts uplifts even for extreme discount rates. Using this model, we improved the WMAPE of the predictions from 35% to 22%. Finally, we built a recommender system on top of this model and displayed the output on a dashboard that is currently in production and updated on a day-to-day basis.

## 8 Future direction

The current predictive model mostly uses game-specific features such as the franchise the game belongs to, the release date, and the genre category. A next step would be to add some more environment variables such as player review scores; the quality of the launch campaign; the awareness of the game; the media spend allocated; web-scrapping social platform (e.g., Twitter, Facebook, Twitch, Instagram) to extract player sentiment, to name just a few. Incorporating these features can reveal some new dependencies and add more explanatory power. Also, being able to predict at finer granularity would bring a significant added value, such as splitting the predictions by game editions (Standard, Deluxe, Super Deluxe), and improving the WMAPE at country level.

# 9 Appendix

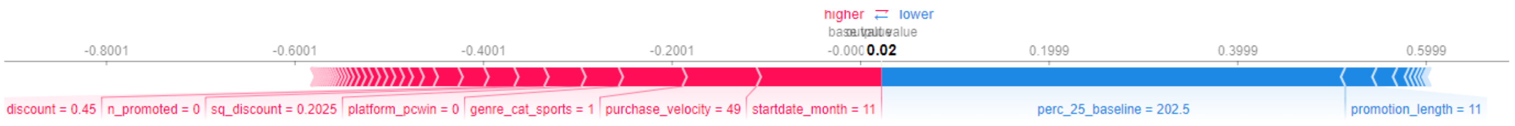
We gather in the appendix the methods/modules/libraries used during this thesis

## 9.1 SHAP (SHapley Additive exPlanations)

SHAP proposes an interesting approach to explain the output of any machine learning model. Once a model is trained, SHAP offers a `TreeExplainer`[15] feature that explains what drives the response variable for a single data point. The `Python` package `shap` can be downloaded from PyPI. It is a handy tool that helped us understand the behavior of the model and investigate a model's output.

Below is an example of the `TreeExplainer` on one prediction of the RERF model.

Figure 17: SHAP TreeExplainer [15][16]



## 9.2 Smoothing function

```python
import numpy as np
from scipy.optimize import curve_fit

# Function to define a degree 5 polynomial
def poly(x, a, b, c, d, e, f):
    return a*x**5+ b*x**4 + c*x**3 + d*x**2 + e*x + f

x = np.linspace(0.05, 1, 20) # discount rates by 5% increments
y = np.array(raw_units) # raw units output by RERF

# Get the weights using a non-linear least squares
weights, _ = curve_fit(poly, x, y)

# Creates the smooth units curve
y_smooth = poly(x, *weights)
```

Listing 1: Python code to obtain the smooth curve from the RERF output. (Used in Section 5.3)

## References

[1] McCarthy's 4Ps, Marketing mix, Wikipedia https://en.wikipedia.org/wiki/Marketing_mix#McCarthy's_4_Ps

[2] Maciej Jaskowski and Szymon Jaroszewicz. Uplift modeling for clinical trial data, 2012.

[3] Causal Inference and Uplift Modeling, A review of the literature. Pierre Gutierrez and Jean-Yves Gérardy, JMLR: Workshop and Conference Proceedings 67:1–13, 2016 http://proceedings.mlr.press/v67/gutierrez17a/gutierrez17a.pdf

[4] The true lift model: a novel data mining approach to response modeling in database marketing. Victor SY Lo. ACM SIGKDD Explorations Newsletter, pages: 78–86, 2002.

[5] Facilitating score and causal inference trees for large observational studies. Xiaogang Su, Juanjuan Fan, Richard A Levine, Xin Yan and Joseph Kang. Journal of Machine Learning Research, 2012.

[6] Optimal personalized treatment rules for marketing interventions: A review of methods, a new proposal, and an insurance case study. Leo Guelman, Montserrat Guillen, Ana M Pérez-Marin, et al. Research Group on Risk in Insurance and Finance, UB Riskcenter, 2014.

[7] Support Vector Machines for Uplift Modeling, Ł. Zaniewicz and S. Jaroszewicz. 2013 IEEE 13th International Conference on Data Mining Workshops, Dallas, TX, 2013, pp. 131-138.

[8] Real-World Uplift Modelling with Significance-Based Uplift Trees. (2012). Radcliffe, Nicholas J. and Patrick D. Surry.

[9] Ensemble Methods and Random Forests, ECE 543 Course: Statistical Learning Theory, University of Illinois. May 2017 https://courses.engr.illinois.edu/ece543/sp2017/projects/Vaishnavi%20Subramanian.pdf

[10] Random Forests, Stanford University, Deparment of Statistics. http://statweb.stanford.edu/~tibs/book/chap17

[11] Random Forests, Leo Breiman, Statistics Department, University of California, Berkeley. January 2001 https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

[12] Random Forest and its implementation. Towards Data Science, 2019. https://towardsdatascience.com/random-forest-and-its-implementation-71824ced454f

[13] Regression-Enhanced Random Forests. Haozhe Zhang, Dan Nettleton, Zhengyuan Zhu. April 2019 https://arxiv.org/abs/1904.10416

[14] A Unified Approach to Interpreting Model Predictions, Scott M. Lundberg, Su-In Lee. 2017 http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[15] TreeExplainer - From local explanations to global understanding with explainable AI for trees. Lundberg, Scott M. and Erion, Gabriel and Chen, Hugh and DeGrave, Alex and Prutkin, Jordan M. and Nair, Bala and Katz, Ronit and Himmelfarb, Jonathan and Bansal, Nisha and Lee, Su-In. Machine Learning Intelligence Journal. 2020. Nature Publishing Group, Volume 2, Number 1, Pages 2.

[16] Forceplot - Explainable machine-learning predictions. Lundberg, Scott M and Nair, Bala and Vavilala, Monica S and Horibe, Mayumi and Eisses, Michael J and Adams, Trevor and Liston, David E and Low, Daniel King-Wai and Newman, Shu-Fang and Kim, Jerry and others. Nature Biomedical Engineering Journal. 2018. Nature Publishing Group, Voulme 2, Number 10, Pages 749.

[17] Jenkins, open source automation server. https://jenkins.io/doc/