scikit-uplift

Release 0.2.0

May 31, 2020

Contents

scikit-uplift (sklift) is a Python module for basic approaches of uplift modeling built on top of scikit-learn.

Uplift prediction aims to estimate the causal impact of a treatment at the individual level.

Read more about uplift modeling problem in *User Guide*, also articles in russian on habr.com: Part 1 and Part 2.

CHAPTER 1

Features

- Comfortable and intuitive style of modelling like scikit-learn;
- Applying any estimator adheres to scikit-learn conventions;
- All approaches can be used in sklearn.pipeline. See example of usage: ;
- Almost all implemented approaches solve both the problem of classification and regression;
- A lot of metrics (Such as *Area Under Uplift Curve* or *Area Under Qini Curve*) are implemented to evaluate your uplift model;
- Useful graphs for analyzing the built model.

The package currently supports the following methods:

1. Solo Model (aka Treatment Dummy and Treatment interaction) approach
2. Class Transformation (aka Class Variable Transformation or Revert Label) approach
3. Two Models (aka naïve approach, or difference score method, or double classifier approach) approach, including Dependent Data Representation

And the following metrics:

1. Uplift@k
2. Area Under Uplift Curve
3. Area Under Qini Curve
4. Weighted average uplift

CHAPTER 2

Project info

- GitHub repository: https://github.com/maks-sh/scikit-uplift
- Github examples: https://github.com/maks-sh/scikit-uplift/tree/master/notebooks
- Documentation: https://scikit-uplift.readthedocs.io/en/latest/
- Contributing guide: https://scikit-uplift.readthedocs.io/en/latest/contributing.html
- License: MIT

CHAPTER 3

Community

We welcome new contributors of all experience levels.

- Please see our Contributing Guide for more details.
- By participating in this project, you agree to abide by its Code of Conduct.

3.1 Installation

Install the package by the following command from PyPI:

```
pip install scikit-uplift
```

Or install from source:

```
git clone https://github.com/maks-sh/scikit-uplift.git
cd scikit-uplift
python setup.py install
```

3.2 Quick Start

See the **RetailHero tutorial notebook** (EN , RU) for details.

Train and predict your uplift model

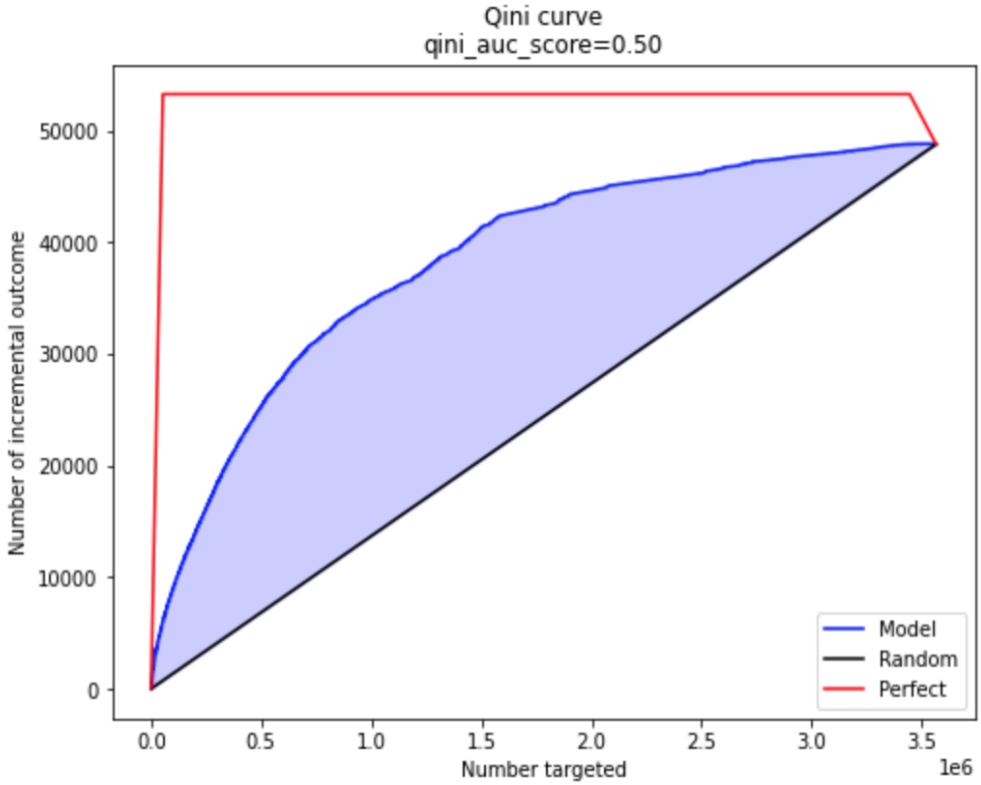
```
# import approaches
from sklift.models import SoloModel, ClassTransformation, TwoModels
# import any estimator adheres to scikit-learn conventions.
from catboost import CatBoostClassifier


# define models
treatment_model = CatBoostClassifier(iterations=50, thread_count=3,
                                     random_state=42, silent=True)
control_model = CatBoostClassifier(iterations=50, thread_count=3,
                                   random_state=42, silent=True)

# define approach
tm = TwoModels(treatment_model, control_model, method='vanilla')
# fit model
tm = tm.fit(X_train, y_train, treat_train)

# predict uplift
uplift_preds = tm.predict(X_val)
```

Evaluate your uplift model

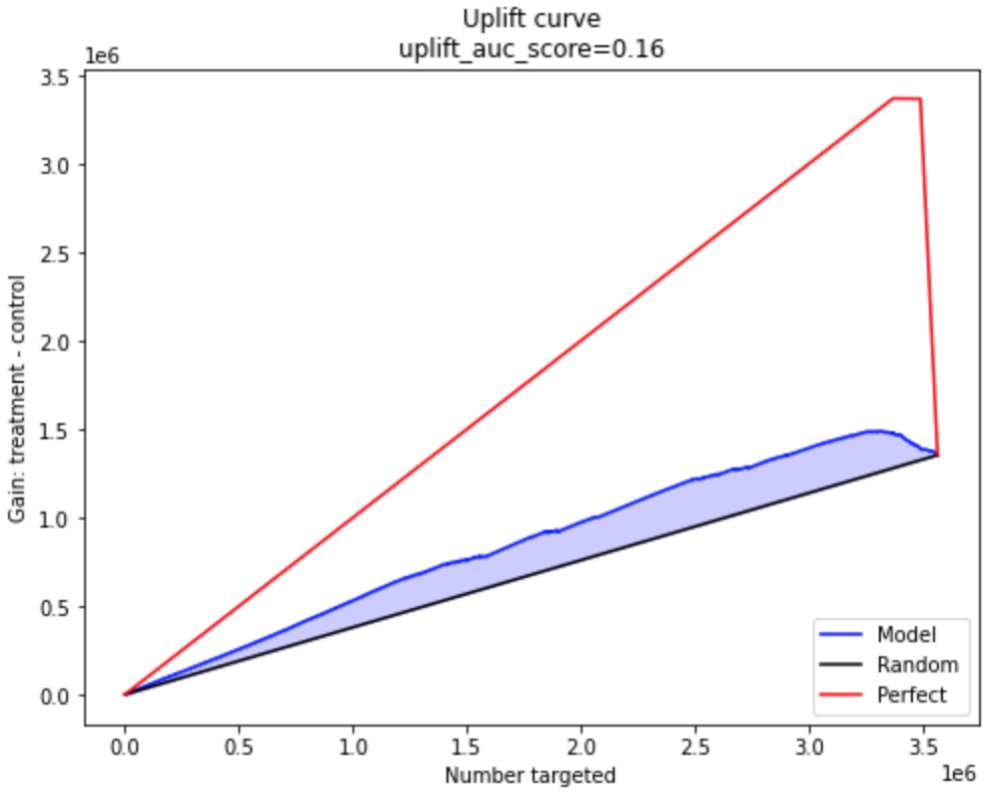
```
# import metrics to evaluate your model
from sklift.metrics import (
    uplift_at_k, uplift_auc_score, qini_auc_score, weighted_average_uplift
)


# Uplift@30%
tm_uplift_at_k = uplift_at_k(y_true=y_val, uplift=uplift_preds,
                             treatment=treat_val,
                             strategy='overall', k=0.3)

# Area Under Qini Curve
tm_qini_auc = qini_auc_score(y_true=y_val, uplift=uplift_preds,
                             treatment=treat_val)

# Area Under Uplift Curve
tm_uplift_auc = uplift_auc_score(y_true=y_val, uplift=uplift_preds,
                                 treatment=treat_val)

# Weighted average uplift
tm_wau = weighted_average_uplift(y_true=y_val, uplift=uplift_preds,
                                 treatment=treat_val)
```

Vizualize the results

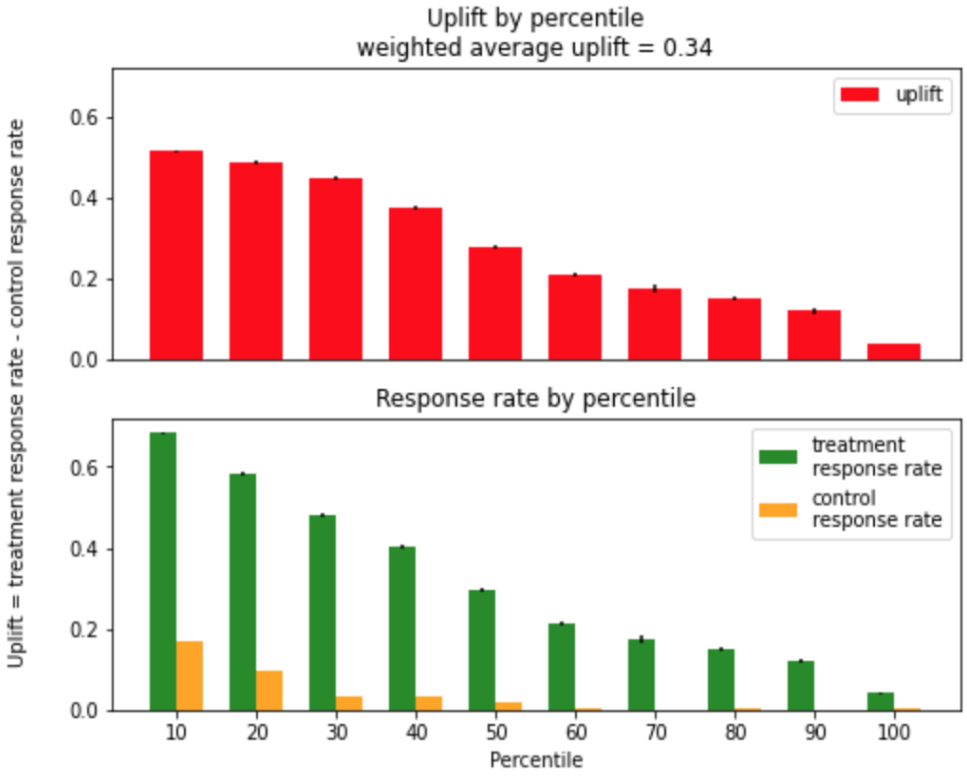
```
from sklift.viz import plot_qini_curve

plot_qini_curve(y_true=y_val, uplift=uplift_preds, treatment=treat_val)
```




```
from sklift.viz import plot_uplift_curve

plot_uplift_curve(y_true=y_val, uplift=uplift_preds, treatment=treat_val)
```




```
from sklift.viz import plot_uplift_by_percentile

plot_uplift_by_percentile(y_true=y_val, uplift=uplift_preds,
                          treatment=treat_val, kind='bar')
```



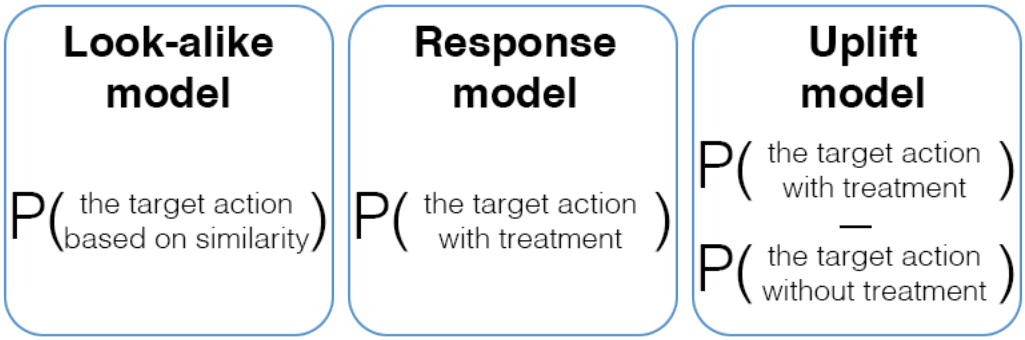
3.3 User Guide

Uplift modeling estimates the effect of communication action on some customer outcome and gives an opportunity to efficiently target customers which are most likely to respond to a marketing campaign. It is relatively easy to implement, but surprisingly poorly covered in the machine learning courses and literature. This guide is going to shed some light on the essentials of causal inference estimating and uplift modeling.

3.3.1 Introduction

Uplift vs other models

Companies use various channels to promote a product to a customer: it can be SMS, push notification, chatbot message in social networks, and many others. There are several ways to use machine learning to select customers for a marketing campaign:



- *The Look-alike model* (or Positive Unlabeled Learning) evaluates a probability that the customer is going to accomplish a target action. A training dataset contains known positive objects (for instance, users who have installed an app) and random negative objects (a random subset of all other customers who have not installed the app). The model searches for customers who are similar to those who made the target action.
- *The Response model* evaluates the probability that the customer is going to accomplish the target action if there was a communication (a.k.a treatment). In this case the training dataset is data collected after some interaction with the customers. In contrast to the first approach, we have confirmed positive and negative observations at our disposal (for instance, the customer who decides to issue a credit card or to decline an offer).
- *The Uplift model* evaluates the net effect of communication by trying to select only those customers who are going to perform the target action only when there is some advertising exposure presenting to them. The model predicts a difference between the customer's behavior when there is a treatment (communication) and when there is no treatment (no communication).

When should we use uplift modeling?

Uplift modeling is used when the customer's target action is likely to happen without any communication. For instance, we want to promote a popular product but we don't want to spend our marketing budget on customers who will buy the product anyway with or without communication. If the product is not popular and it is has to be promoted to be bought, then a task turns to the response modeling task.

References

1 Radcliffe, N.J. (2007). Using control groups to target on predicted lift: Building and assessing uplift model. Direct Market J Direct Market Assoc Anal Council, 1:14–21, 2007.

Causal Inference: Basics

In a perfect world, we want to calculate a difference in a person's reaction received communication and the reaction without receiving any communication. But there is a problem: we can not make a communication (send an e-mail) and do not make a communication (no e-mail) at the same time.

Denoting Y_i^1 person i's outcome when receives the treatment (a presence of the communication) and Y_i^0 i's outcome when he receives no treatment (control, no communication), the *causal effect* τ_i of the treatment *vis-a-vis* no treatment is given by:

$$\tau_i = Y_i^1 - Y_i^0$$

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

$$CATE = E[Y_i^1|X_i] - E[Y_i^0|X_i]$$

Where X_i - features vector describing i-th person.

We can observe neither causal effect nor CATE for the i-th object, and, accordingly, we can't optimize it. But we can

estimate CATE or *uplift* of an object:

$$\textbf{uplift} = \widehat{CATE} = E[Y_i|X_i = x, W_i = 1] - E[Y_i|X_i = x, W_i = 0]$$

Where:

- $W_i \in 0, 1$ - a binary variable: 1 if person i receives the treatment *treatment group*, and 0 if person i receives no treatment *control group*;
- Y_i - person i's observed outcome, which is actually equal:

$$Y_i = W_i * Y_i^1 + (1 - W_i) * Y_i^0 = \begin{cases} Y_i^1, & \text{if } W_i = 1 \\ Y_i^0, & \text{if } W_i = 0 \end{cases}$$

This won't identify the CATE unless one is willing to assume that W_i is independent of Y_i^1 and Y_i^0 conditional on X_i. This assumption is the so-called *Unconfoundedness Assumption* or the *Conditional Independence Assumption* (CIA) found in the social sciences and medical literature. This assumption holds true when treatment assignment is random conditional on X_i. Briefly this can be written as:

$$CIA : \{Y_i^0, Y_i^1\} \perp\!\!\!\perp W_i|X_i$$

Also introduce additional useful notation. Let us define the *propensity score*, $p(X_i) = P(W_i = 1|X_i)$, i.e. the probability of treatment given X_i.

References

1 Gutierrez, P., & Gérardy, J. Y. (2017). Causal Inference and Uplift Modelling: A Review of the Literature. In International Conference on Predictive Applications and APIs (pp. 1-13).

Data collection

We need to evaluate a difference between two events that are mutually exclusive for a particular customer (either we communicate with a person, or we don't; you can't do both actions at the same time). This is why there are additional requirements for collecting data when building an uplift model.

There are few additional steps different from a standard data collection procedure. You should run an experiment:

1. Randomly divide a representative part of the customer base into a treatment (receiving communication) and a control (receiving no communication) groups;
2. Evaluate the marketing experiment for the treatment group.

Data collected from the marketing experiment consists of the customer's responses to the marketing offer (target).

The only difference between the experiment and the future uplift model's campaign is a fact that in the first case we choose random customers to make a promotion. In the second case the choice of a customer to communicate with is based on the predicted value returned by the uplift model. If the marketing campaign significantly differs from the experiment used to collect data, the model will be less accurate.

There is a trick: before running the marketing campaign, it is recommended to randomly subset a small part of the customer base and divide it into a control and a treatment group again, similar to the previous experiment. Using this data, you will not only be able to accurately evaluate the effectiveness of the campaign but also collect additional data for a further model retraining.

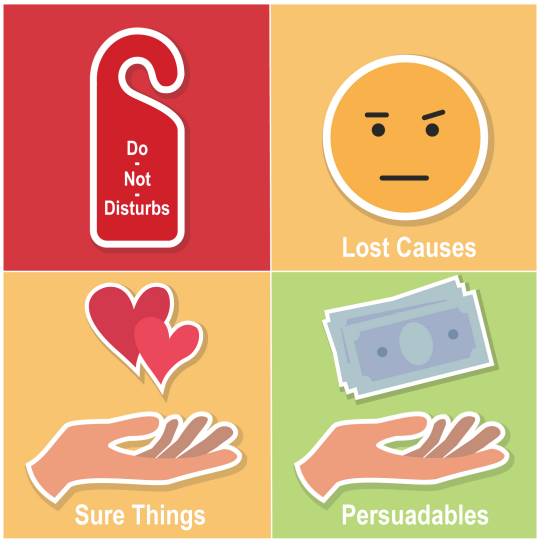
It is recommended to configure a development of the uplift model and the campaign launch as an iterative process: each iteration will collect new training data. It should consist of a mix of a random customer subset and customers selected by the model.

References

1 Verbeke, Wouter & Baesens, Bart & Bravo, Cristián. (2018). Profit Driven Business Analytics: A Practitioner's Guide to Transforming Big Data into Added Value.

Types of customers

We can determine 4 types of customers based on a response to a treatment:



- *Do-Not-Disturbs (a.k.a. Sleeping-dogs)* have a strong negative response to a marketing communication. They are going to purchase if *NOT* treated and will *NOT* purchase *IF* treated. It is not only a wasted marketing budget but also a negative impact. For instance, customers targeted could result in rejecting current products or services. In terms of math: $W_i = 1, Y_i = 0$ or $W_i = 0, Y_i = 1$.
- *Lost Causes* will *NOT* purchase the product *NO MATTER* they are contacted or not. The marketing budget in this case is also wasted because it has no effect. In terms of math: $W_i = 1, Y_i = 0$ or $W_i = 0, Y_i = 0$.
- *Sure Things* will purchase *ANYWAY* no matter they are contacted or not. There is no motivation to spend the budget because it also has no effect. In terms of math: $W_i = 1, Y_i = 1$ or $W_i = 0, Y_i = 1$.
- *Persuadables* will always respond *POSITIVE* to the marketing communication. They is going to purchase *ONLY* if contacted (or sometimes they purchase *MORE* or *EARLIER* only if contacted). This customer's type should be the only target for the marketing campaign. In terms of math: $W_i = 0, Y_i = 0$ or $W_i = 1, Y_i = 1$.

Because we can't communicate and not communicate with the customer at the same time, we will never be able to observe exactly which type a particular customer belongs to.

Depends on the product characteristics and the customer base structure some types may be absent. In addition, a customer response depends heavily on various characteristics of the campaign, such as a communication channel or a type and a size of the marketing offer. To maximize profit, these parameters should be selected.

Thus, when predicting uplift score and selecting a segment by the highest score, we are trying to find the only one type: **persuadables**.

References

1 Kane, K., V. S. Y. Lo, and J. Zheng. Mining for the Truly Responsive Customers and Prospects Using True-Lift Modeling: Comparison of New and Existing Methods. Journal of Marketing Analytics 2 (4): 218–238. 2014.

2 Verbeke, Wouter & Baesens, Bart & Bravo, Cristián. (2018). Profit Driven Business Analytics: A Practitioner's Guide to Transforming Big Data into Added Value.

3.3.2 Models

Single model approaches

Single model with treatment as feature

The most intuitive and simple uplift modeling technique. A training set consists of two groups: treatment samples and control samples. There is also a binary treatment flag added as a feature to the training set. After the model is trained, at the scoring time it is going to be applied twice: with the treatment flag equals *1* and with the treatment flag equals *0*. Subtracting these model's outcomes for each test sample, we will get an estimate of the uplift.

The training process:

$$fit\begin{pmatrix} x_{11} & \cdots & x_{1k} & w_1 & y_1 \\ \vdots & \ddots & \vdots & \dots, & \dots \\ x_{n1} & \cdots & x_{nk} & w_n & y_n \end{pmatrix}$$

$$X_{train} \quad W_{train} \quad Y_{train}$$

The process of applying the model:

$$predict\ proba\begin{pmatrix} x_{11} & \cdots & x_{1k} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_{m1} & \cdots & x_{mk} & 1 \end{pmatrix} - predict\ proba\begin{pmatrix} x_{11} & \cdots & x_{1k} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ x_{m1} & \cdots & x_{mk} & 0 \end{pmatrix} = \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix}$$

$$X_{test} \quad W_1 \qquad\qquad X_{test} \quad W_0 \quad uplift$$

Hint: In sklift this approach corresponds to the `SoloModel` class and the **dummy** method.

Treatment interaction

The single model approach has various modifications. For instance, we can update the number of attributes in the training set by adding the product of each attribute and the treatment flag:

client_id	x_1	x_2	...	w	y
6d330e	1	42	...	1	**0**
b77945	0	19	...	0	**1**
6e6894	0	23	...	1	**1**
...	...	...	...	...	**...**
15696e	1	60	...	0	**0**

$\Rightarrow$

client_id	x_1	x_2	...	w	x_1*w	x_2*w	...	y
6d330e	1	42	...	1	1	42	...	**0**
b77945	0	19	...	0	0	0	...	**1**
6e6894	0	23	...	1	0	23	...	**1**
...	...	...	...	...	...	...	...	**...**
15696e	1	60	...	0	0	0	...	**0**

Hint: In sklift this approach corresponds to the *SoloModel* class and the **treatment_interaction** method.

References

1 Lo, Victor. (2002). The True Lift Model - A Novel Data Mining Approach to Response Modeling in Database Marketing. SIGKDD Explorations. 4. 78-86.

Examples using `sklift.models.SoloModel`

1. The overview of the basic approaches to solving the Uplift Modeling problem

In English		nbviewer	github
In Russian		nbviewer	github

Class Transformation

Warning: This approach is only suitable for classification problem

Simple yet powerful and mathematically proven uplift modeling method, presented in 2012. The main idea is to predict a slightly changed target Z_i:

$$Z_i = Y_i \cdot W_i + (1 - Y_i) \cdot (1 - W_i),$$

- Z_i - new target for the i customer;
- Y_i - previous target for the i customer;
- W_i - treatment flag assigned to the i customer.

In other words, the new target equals 1 if a response in the treatment group is as good as a response in the control group and equals 0 otherwise:

$$Z_i = \begin{cases} 1, & \text{if } W_i = 1 \text{ and } Y_i = 1 \\ 1, & \text{if } W_i = 0 \text{ and } Y_i = 0 \\ 0, & \text{otherwise} \end{cases}$$

Let's go deeper and estimate the conditional probability of the target variable:

$$\begin{aligned}
&P(Z = 1|X = x) = \\
&= P(Z = 1|X = x, W = 1) \cdot P(W = 1|X = x) + \\
&+ P(Z = 1|X = x, W = 0) \cdot P(W = 0|X = x) = \\
&= P(Y = 1|X = x, W = 1) \cdot P(W = 1|X = x) + \\
&+ P(Y = 0|X = x, W = 0) \cdot P(W = 0|X = x).
\end{aligned}$$

We assume that W is independent of $X = x$ by design. Thus we have: $P(W|X = x) = P(W)$ and

$$\begin{aligned}
&P(Z = 1|X = x) = \\
&= P^T(Y = 1|X = x) \cdot P(W = 1) + \\
&+ P^C(Y = 0|X = x) \cdot P(W = 0)
\end{aligned}$$

Also, we assume that $P(W = 1) = P(W = 0) = \frac{1}{2}$, which means that during the experiment the control and the treatment groups were divided in equal proportions. Then we get the following:

$$\begin{aligned}
&P(Z = 1|X = x) = \\
&= P^T(Y = 1|X = x) \cdot \frac{1}{2} + P^C(Y = 0|X = x) \cdot \frac{1}{2} \Rightarrow
\end{aligned}$$

$$\begin{aligned}
&2 \cdot P(Z = 1|X = x) = \\
&= P^T(Y = 1|X = x) + P^C(Y = 0|X = x) = \\
&= P^T(Y = 1|X = x) + 1 - P^C(Y = 1|X = x) \Rightarrow \\
&\Rightarrow P^T(Y = 1|X = x) - P^C(Y = 1|X = x) = \\
&= uplift = 2 \cdot P(Z = 1|X = x) - 1
\end{aligned}$$

Thus, by doubling the estimate of the new target Z and subtracting one we will get an estimation of the uplift:

$$uplift = 2 \cdot P(Z = 1) - 1$$

This approach is based on the assumption: $P(W = 1) = P(W = 0) = \frac{1}{2}$. That is the reason that it has to be used only in cases where the number of treated customers (communication) is equal to the number of control customers (no communication).

Hint: In sklift this approach corresponds to the `ClassTransformation` class.

References

1 Maciej Jaskowski and Szymon Jaroszewicz. Uplift modeling for clinical trial data. ICML Workshop on Clinical Data Analysis, 2012.

Examples using `sklift.models.ClassTransformation`

1. The overview of the basic approaches to the Uplift Modeling problem

In English		nbviewer	github
In Russian		nbviewer	github

2. The 2nd place solution of X5 RetailHero uplift contest by Kirill Liksakov

In English	nbviewer	github

Two models approaches

The two models approach can be found in almost every uplift modeling research. It is often used as a baseline model.

Two independent models

Hint: In sklift this approach corresponds to the `sklift.models.TwoModels` class and the **vanilla** method.

The main idea is to estimate the conditional probabilities of the treatment and control groups separately.

1. Train the first model using the treatment set.
2. Train the second model using the control set.
3. Inference: subtract the control model scores from the treatment model scores.

The training process:

$$model^T = fit\begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{p1} & \cdots & x_{pk} \end{pmatrix}, \begin{pmatrix} y_1 \\ \cdots \\ y_p \end{pmatrix}, \quad model^C = fit\begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{q1} & \cdots & x_{qk} \end{pmatrix}, \begin{pmatrix} y_1 \\ \cdots \\ y_q \end{pmatrix}$$

$$X_{train\_treat} \quad Y_{train\_treat} \qquad\qquad X_{train\_control} \quad Y_{train\_control}$$

The process of applying the model:

$$\begin{matrix} model^T \\ predict \\ proba \end{matrix}\begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mk} \end{pmatrix} - \begin{matrix} model^c \\ predict \\ proba \end{matrix}\begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mk} \end{pmatrix} = \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix}$$

$$X_{test} \qquad\qquad X_{test} \qquad\qquad uplift$$

The main disadvantage of this method is that if the uplift signal is weak, it can be lost since both models focus on predicting an original response, not the uplift.

Two dependent models

The dependent data representation approach is based on the classifier chain method originally developed for multi-class classification problems. The idea is that if there are L different labels, you can build L different classifiers, each of which solves the problem of binary classification and in the learning process, each subsequent classifier uses the predictions of the previous ones as additional features. The authors of this method proposed to use the same idea to solve the problem of uplift modeling in two stages.

Hint: In sklift this approach corresponds to the `TwoModels` class and the **ddr_control** method.

At the beginning we train the classifier based on the control data:

$$P^C = P(Y = 1|X, W = 0),$$

Next, we estimate the P_C predictions and use them as a feature for the second classifier. It effectively reflects a dependency between treatment and control datasets:

$$P^T = P(Y = 1|X, P_C(X), W = 1)$$

To get the uplift for each observation, calculate the difference:

$$uplift(x_i) = P^T(x_i, P_C(x_i)) - P^C(x_i)$$

Intuitively, the second classifier learns the difference between the expected probability in the treatment and the control sets which is the uplift.

The training process:

$$model^C = fit\begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{q1} & \cdots & x_{qk} \end{pmatrix}, \begin{pmatrix} y_1 \\ \cdots \\ y_q \end{pmatrix},$$

$X_{train\_control}$ $Y_{train\_control}$

$$P^C = \begin{matrix} model^c \\ predict \\ proba \end{matrix}\begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{p1} & \cdots & x_{pk} \end{pmatrix}, \quad model^T = fit\begin{pmatrix} x_{11} & \cdots & x_{1k} & P_1^c & y_1 \\ \vdots & \ddots & \vdots & \vdots & \cdots \\ x_{p1} & \cdots & x_{pk} & P_p^c & y_p \end{pmatrix},$$

X_{train_treat} X_{train_treat} P^c Y_{train_treat}

The process of applying the model:

$$\begin{matrix} model^T \\ predict \\ proba \end{matrix}\begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mk} \end{matrix}\begin{matrix} model^c \\ predict \\ proba \end{matrix}\begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mk} \end{pmatrix}\Bigg) - \begin{matrix} model^c \\ predict \\ proba \end{matrix}\begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mk} \end{pmatrix} = \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix}$$

X_{test} $P^C(X_{test})$ X_{test} $uplift$

Similarly, you can first train the P_T classifier and then use its predictions as a feature for the P_C classifier.

Hint: In sklift this approach corresponds to the `TwoModels` class and the **ddr_treatment** method.

There is an important remark about the data nature. It is important to calibrate model's scores into probabilities if treatment and control data have a different nature. Model calibration techniques are well described in the scikit-learn documentation.

References

1 Betlei, Artem & Diemert, Eustache & Amini, Massih-Reza. (2018). Uplift Prediction with Dependent Feature Representation in Imbalanced Treatment and Control Conditions: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13–16, 2018, Proceedings, Part V. 10.1007/978-3-030-04221-9_5.

2 Zhao, Yan & Fang, Xiao & Simchi-Levi, David. (2017). Uplift Modeling with Multiple Treatments and General Response Types. 10.1137/1.9781611974973.66.

Examples using `sklift.models.TwoModels`

1. The overview of the basic approaches to solving the Uplift Modeling problem

In English		nbviewer	github
In Russian		nbviewer	github

3.3.3 Credits

Authors:

- Irina Elisova
- Maksim Shevchenko

Acknowledgements:

- Kirill Liksakov - uplift metrics research
- Alina Zhukova - artwork: User Guide cover and key pictures

3.3.4 Citations

If you find this User Guide useful for your research, please consider citing:

```
@misc{user-guide-for-uplift-modeling,
  author = {Maksim Shevchenko, Irina Elisova},
  title = {User Guide for uplift modeling and casual inference},
  year = {2020},
  publisher = {GitHub},
  journal = {GitHub repository},
  howpublished = {\url{https://scikit-uplift.readthedocs.io/en/latest/user_guide/
↪index.html}}
}
```

3.4 API sklift

This is the modules reference of scikit-uplift.

3.4.1 sklift.models

See *Models* section of the User Guide for further details.

sklift.models.SoloModel

class sklift.models.models.**SoloModel**(*estimator*, *method='dummy'*)

aka Treatment Dummy approach, or Single model approach, or S-Learner.

Fit solo model on whole dataset with 'treatment' as an additional feature.

Each object from the test sample is scored twice: with the communication flag equal to 1 and equal to 0. Subtracting the probabilities for each observation, we get the uplift.

Return delta of predictions for each example.

Read more in the *User Guide*.

Parameters

- **estimator** (*estimator object implementing 'fit'*) – The object to use to fit the data.

- **method** (*string, 'dummy' or 'treatment_interaction', default='dummy'*) – Specifies the approach:
  - **'dummy':** Single model;
  - **'treatment_interaction':** Single model including treatment interactions.

trmnt_preds_
Estimator predictions on samples when treatment.

Type array-like, shape (n_samples,)

ctrl_preds_
Estimator predictions on samples when control.

Type array-like, shape (n_samples,)

Example:

```
# import approach
from sklift.models import SoloModel
# import any estimator adheres to scikit-learn conventions
from catboost import CatBoostClassifier


sm = SoloModel(CatBoostClassifier(verbose=100, random_state=777))  # define␣
↪approach
sm = sm.fit(X_train, y_train, treat_train, estimator_fit_params={{'plot': True})
↪# fit the model
uplift_sm = sm.predict(X_val)  # predict uplift
```

References

Lo, Victor. (2002). The True Lift Model - A Novel Data Mining Approach to Response Modeling in Database Marketing. SIGKDD Explorations. 4. 78-86.

See also:

Other approaches:

- *ClassTransformation*: Class Variable Transformation approach.
- *TwoModels*: Double classifier approach.

Other:

- *plot_uplift_preds()*: Plot histograms of treatment, control and uplift predictions.

fit(*X*, *y*, *treatment*, *estimator_fit_params=None*)
Fit the model according to the given training data.

For each test example calculate predictions on new set twice: by the first and second models. After that calculate uplift as a delta between these predictions.

Return delta of predictions for each example.

Parameters

- **X** (*array-like, shape (n_samples, n_features)*) – Training vector, where n_samples is the number of samples and n_features is the number of features.
- **y** (*array-like, shape (n_samples,)*) – Target vector relative to X.

- **treatment** (*array-like, shape (n_samples,)*) – Binary treatment vector relative to X.
- **estimator_fit_params** (*dict, optional*) – Parameters to pass to the fit method of the estimator.

Returns self

Return type object

predict(*X*)

Perform uplift on samples in X.

Parameters **X** (*array-like, shape (n_samples, n_features)*) – Training vector, where n_samples is the number of samples and n_features is the number of features.

Returns uplift

Return type array (shape (n_samples,))

sklift.models.ClassTransformation

class sklift.models.models.**ClassTransformation**(*estimator*)

aka Class Variable Transformation or Revert Label approach.

Redefine target variable, which indicates that treatment make some impact on target or did target is negative without treatment: `Z = Y * W + (1 - Y)(1 - W)`,

where `Y` - target vector, `W` - vector of binary communication flags.

Then, `Uplift ~ 2 * (Z == 1) - 1`

Returns only uplift predictions.

Read more in the *User Guide*.

Parameters **estimator** (*estimator object implementing 'fit'*) – The object to use to fit the data.

Example:

```
# import approach
from sklift.models import ClassTransformation
# import any estimator adheres to scikit-learn conventions
from catboost import CatBoostClassifier


# define approach
ct = ClassTransformation(CatBoostClassifier(verbose=100, random_state=777))
# fit the model
ct = ct.fit(X_train, y_train, treat_train, estimator_fit_params={{'plot': True})
# predict uplift
uplift_ct = ct.predict(X_val)
```

References

Maciej Jaskowski and Szymon Jaroszewicz. Uplift modeling for clinical trial data. ICML Workshop on Clinical Data Analysis, 2012.

See also:

Other approaches:

- *SoloModel*: Single model approach.
- *TwoModels*: Double classifier approach.

fit(*X*, *y*, *treatment*, *estimator_fit_params=None*)
Fit the model according to the given training data.

Parameters

- **X** (*array-like, shape (n_samples, n_features)*) – Training vector, where n_samples is the number of samples and n_features is the number of features.
- **y** (*array-like, shape (n_samples,)*) – Target vector relative to X.
- **treatment** (*array-like, shape (n_samples,)*) – Binary treatment vector relative to X.
- **estimator_fit_params** (*dict, optional*) – Parameters to pass to the fit method of the estimator.

Returns self

Return type object

predict(*X*)
Perform uplift on samples in X.

Parameters **X** (*array-like, shape (n_samples, n_features)*) – Training vector, where n_samples is the number of samples and n_features is the number of features.

Returns uplift

Return type array (shape (n_samples,))

sklift.models.TwoModels

class sklift.models.models.**TwoModels**(*estimator_trmnt*, *estimator_ctrl*, *method='vanilla'*)
aka naïve approach, or difference score method, or double classifier approach.

Fit two separate models: on the treatment data and on the control data.

Read more in the *User Guide*.

Parameters

- **estimator_trmnt** (*estimator object implementing 'fit'*) – The object to use to fit the treatment data.
- **estimator_ctrl** (*estimator object implementing 'fit'*) – The object to use to fit the control data.
- **method** (*string, 'vanilla', 'ddr_control' or 'ddr_treatment', default='vanilla'*) – Specifies the approach:
  - **'vanilla':** Two independent models;
  - **'ddr_control':** Dependent data representation (First train control estimator).
  - **'ddr_treatment':** Dependent data representation (First train treatment estimator).

trmnt_preds_
Estimator predictions on samples when treatment.

Type array-like, shape (n_samples,)

ctrl_preds_
Estimator predictions on samples when control.

Type array-like, shape (n_samples,)

Example:

```
# import approach
from sklift.models import TwoModels
# import any estimator adheres to scikit-learn conventions
from catboost import CatBoostClassifier


estimator_trmnt = CatBoostClassifier(silent=True, thread_count=2, random_state=42)
estimator_ctrl = CatBoostClassifier(silent=True, thread_count=2, random_state=42)

# define approach
tm_ctrl = TwoModels(
    estimator_trmnt=estimator_trmnt,
    estimator_ctrl=estimator_ctrl,
    method='ddr_control'
)

# fit the models
tm_ctrl = tm_ctrl.fit(
    X_train, y_train, treat_train,
    estimator_trmnt_fit_params={'cat_features': cat_features},
    estimator_ctrl_fit_params={'cat_features': cat_features}
)
uplift_tm_ctrl = tm_ctrl.predict(X_val)  # predict uplift
```

References Betlei, Artem & Diemert, Eustache & Amini, Massih-Reza. (2018). Uplift Prediction with Dependent Feature Representation in Imbalanced Treatment and Control Conditions: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13–16, 2018, Proceedings, Part V. 10.1007/978-3-030-04221-9_5.

Zhao, Yan & Fang, Xiao & Simchi-Levi, David. (2017). Uplift Modeling with Multiple Treatments and General Response Types. 10.1137/1.9781611974973.66.

See also:

Other approaches:

- *SoloModel*: Single model approach.
- *ClassTransformation*: Class Variable Transformation approach.

Other:

- *plot_uplift_preds()*: Plot histograms of treatment, control and uplift predictions.

fit (*X*, *y*, *treatment*, *estimator_trmnt_fit_params=None*, *estimator_ctrl_fit_params=None*)
Fit the model according to the given training data.

For each test example calculate predictions on new set twice: by the first and second models. After that calculate uplift as a delta between these predictions.

Return delta of predictions for each example.

Parameters

- **X** (*array-like, shape (n_samples, n_features)*) – Training vector, where n_samples is the number of samples and n_features is the number of features.
- **y** (*array-like, shape (n_samples,)*) – Target vector relative to X.
- **treatment** (*array-like, shape (n_samples,)*) – Binary treatment vector relative to X.
- **estimator_trmnt_fit_params** (*dict, optional*) – Parameters to pass to the fit method of the treatment estimator.
- **estimator_ctrl_fit_params** (*dict, optional*) – Parameters to pass to the fit method of the control estimator.

Returns self

Return type object

predict(*X*)

Perform uplift on samples in X.

Parameters **X** (*array-like, shape (n_samples, n_features)*) – Training vector, where n_samples is the number of samples and n_features is the number of features.

Returns uplift

Return type array (shape (n_samples,))

3.4.2 sklift.metrics

sklift.metrics.uplift_at_k

sklift.metrics.metrics.**uplift_at_k**(*y_true*, *uplift*, *treatment*, *strategy*, *k=0.3*)

Compute uplift at first k observations by uplift of the total sample.

Parameters

- **y_true** (*1d array-like*) – Correct (true) target values.
- **uplift** (*1d array-like*) – Predicted uplift, as returned by a model.
- **treatment** (*1d array-like*) – Treatment labels.
- **k** (*float or int*) – If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the computation of uplift. If int, represents the absolute number of samples.
- **strategy** (*string, ['overall', 'by_group']*) – Determines the calculating strategy.
  - **'overall':** The first step is taking the first k observations of all test data ordered by uplift prediction (overall both groups - control and treatment) and conversions in treatment and control groups calculated only on them. Then the difference between these conversions is calculated.
  - **'by_group':** Separately calculates conversions in top k observations in each group (control and treatment) sorted by uplift predictions. Then the difference between these conversions is calculated

Changed in version 0.1.0: Add supporting absolute values for `k` parameter Add parameter `strategy`

- **Returns** Uplift score at first k observations of the total sample.

Return type float

See also:

uplift_auc_score(): Compute normalized Area Under the Uplift curve from prediction scores.

qini_auc_score(): Compute normalized Area Under the Qini Curve from prediction scores.

sklift.metrics.uplift_curve

sklift.metrics.metrics.**uplift_curve**(*y_true*, *uplift*, *treatment*)

Compute Uplift curve.

For computing the area under the Uplift Curve, see *uplift_auc_score()*.

Parameters

- **y_true** (*1d array-like*) – Correct (true) target values.
- **uplift** (*1d array-like*) – Predicted uplift, as returned by a model.
- **treatment** (*1d array-like*) – Treatment labels.

Returns Points on a curve.

Return type array (shape = [>2]), array (shape = [>2])

See also:

uplift_auc_score(): Compute normalized Area Under the Uplift curve from prediction scores.

perfect_uplift_curve(): Compute the perfect Uplift curve.

plot_uplift_curve(): Plot Uplift curves from predictions.

qini_curve(): Compute Qini curve.

References

Devriendt, F., Guns, T., & Verbeke, W. (2020). Learning to rank for uplift modeling. ArXiv, abs/2002.05897.

sklift.metrics.perfect_uplift_curve

sklift.metrics.metrics.**perfect_uplift_curve**(*y_true*, *treatment*)

Compute the perfect (optimum) Uplift curve.

This is a function, given points on a curve. For computing the area under the Uplift Curve, see *uplift_auc_score()*.

Parameters

- **y_true** (*1d array-like*) – Correct (true) target values.
- **treatment** (*1d array-like*) – Treatment labels.

Returns Points on a curve.

Return type array (shape = [>2]), array (shape = [>2])

See also:

uplift_curve(): Compute the area under the Qini curve.

uplift_auc_score(): Compute normalized Area Under the Uplift curve from prediction scores.

plot_uplift_curve(): Plot Uplift curves from predictions.

sklift.metrics.uplift_auc_score

sklift.metrics.metrics.**uplift_auc_score**(*y_true*, *uplift*, *treatment*)

Compute normalized Area Under the Uplift Curve from prediction scores.

By computing the area under the Uplift curve, the curve information is summarized in one number. For binary outcomes the ratio of the actual uplift gains curve above the diagonal to that of the optimum Uplift Curve.

Parameters

- **y_true** (*1d array-like*) – Correct (true) target values.
- **uplift** (*1d array-like*) – Predicted uplift, as returned by a model.
- **treatment** (*1d array-like*) – Treatment labels.

Returns Area Under the Uplift Curve.

Return type float

See also:

uplift_curve(): Compute Uplift curve.

perfect_uplift_curve(): Compute the perfect (optimum) Uplift curve.

plot_uplift_curve(): Plot Uplift curves from predictions.

qini_auc_score(): Compute normalized Area Under the Qini Curve from prediction scores.

sklift.metrics.qini_curve

sklift.metrics.metrics.**qini_curve**(*y_true*, *uplift*, *treatment*)

Compute Qini curve.

For computing the area under the Qini Curve, see *qini_auc_score()*.

Parameters

- **y_true** (*1d array-like*) – Correct (true) target values.
- **uplift** (*1d array-like*) – Predicted uplift, as returned by a model.
- **treatment** (*1d array-like*) – Treatment labels.

Returns Points on a curve.

Return type array (shape = [>2]), array (shape = [>2])

See also:

uplift_curve(): Compute the area under the Qini curve.

perfect_qini_curve(): Compute the perfect Qini curve.

plot_qini_curves(): Plot Qini curves from predictions..

uplift_curve(): Compute Uplift curve.

References

Nicholas J Radcliffe. (2007). Using control groups to target on predicted lift: Building and assessing uplift model. Direct Marketing Analytics Journal, (3):14–21, 2007.

Devriendt, F., Guns, T., & Verbeke, W. (2020). Learning to rank for uplift modeling. ArXiv, abs/2002.05897.

sklift.metrics.perfect_qini_curve

sklift.metrics.metrics.**perfect_qini_curve**(*y_true*, *treatment*, *negative_effect=True*)

Compute the perfect (optimum) Qini curve.

For computing the area under the Qini Curve, see *qini_auc_score()*.

Parameters

- **y_true** (*1d array-like*) – Correct (true) target values.
- **treatment** (*1d array-like*) – Treatment labels.
- **negative_effect** (*bool*) – If True, optimum Qini Curve contains the negative effects (negative uplift because of campaign). Otherwise, optimum Qini Curve will not contain the negative effects.

Returns Points on a curve.

Return type array (shape = [>2]), array (shape = [>2])

See also:

qini_curve(): Compute Qini curve.

qini_auc_score(): Compute the area under the Qini curve.

plot_qini_curves(): Plot Qini curves from predictions..

sklift.metrics.qini_auc_score

sklift.metrics.metrics.**qini_auc_score**(*y_true*, *uplift*, *treatment*, *negative_effect=True*)

Compute normalized Area Under the Qini curve (aka Qini coefficient) from prediction scores.

By computing the area under the Qini curve, the curve information is summarized in one number. For binary outcomes the ratio of the actual uplift gains curve above the diagonal to that of the optimum Qini curve.

Parameters

- **y_true** (*1d array-like*) – Correct (true) target values.
- **uplift** (*1d array-like*) – Predicted uplift, as returned by a model.
- **treatment** (*1d array-like*) – Treatment labels.
- **negative_effect** (*bool*) – If True, optimum Qini Curve contains the negative effects (negative uplift because of campaign). Otherwise, optimum Qini Curve will not contain the negative effects.

  New in version 0.2.0.

Returns Qini coefficient.

Return type float

See also:

qini_curve(): Compute Qini curve.

perfect_qini_curve(): Compute the perfect (optimum) Qini curve.

plot_qini_curves(): Plot Qini curves from predictions..

uplift_auc_score(): Compute normalized Area Under the Uplift curve from prediction scores.

References

Nicholas J Radcliffe. (2007). Using control groups to target on predicted lift: Building and assessing uplift model. Direct Marketing Analytics Journal, (3):14–21, 2007.

sklift.metrics.weighted_average_uplift

sklift.metrics.metrics.**weighted_average_uplift**(*y_true*, *uplift*, *treatment*, *strategy='overall'*, *bins=10*)

Weighted average uplift.

It is an average of uplift by percentile. Weights are sizes of the treatment group by percentile.

Parameters

- **y_true** (*1d array-like*) – Correct (true) target values.
- **uplift** (*1d array-like*) – Predicted uplift, as returned by a model.
- **treatment** (*1d array-like*) – Treatment labels.
- **strategy** (*string, ['overall', 'by_group']*) – Determines the calculating strategy. Default is 'overall'.
  - **'overall':** The first step is taking the first k observations of all test data ordered by uplift prediction (overall both groups - control and treatment) and conversions in treatment and control groups calculated only on them. Then the difference between these conversions is calculated.
  - **'by_group':** Separately calculates conversions in top k observations in each group (control and treatment) sorted by uplift predictions. Then the difference between these conversions is calculated
- **bins** (*int*) – Determines the number of bins (and the relative percentile) in the data. Default is 10.

Returns Weighted average uplift.

Return type float

sklift.metrics.uplift_by_percentile

sklift.metrics.metrics.**uplift_by_percentile**(*y_true*, *uplift*, *treatment*, *strategy='overall'*, *bins=10*, *std=False*, *total=False*)

Compute metrics: uplift, group size, group response rate, standard deviation at each percentile.

Metrics in columns and percentiles in rows of pandas DataFrame:

- n_treatment, n_control - group sizes.
- response_rate_treatment, response_rate_control - group response rates.

- `uplift` - treatment response rate substract control response rate.
- `std_treatment`, `std_control` - (optional) response rates standard deviation.
- `std_uplift` - (optional) uplift standard deviation.

Parameters

- **y_true** (*1d array-like*) – Correct (true) target values.
- **uplift** (*1d array-like*) – Predicted uplift, as returned by a model.
- **treatment** (*1d array-like*) – Treatment labels.
- **strategy** (*string, ['overall', 'by_group']*) – Determines the calculating strategy. Default is 'overall'.
  - **'overall':** The first step is taking the first k observations of all test data ordered by uplift prediction (overall both groups - control and treatment) and conversions in treatment and control groups calculated only on them. Then the difference between these conversions is calculated.
  - **'by_group':** Separately calculates conversions in top k observations in each group (control and treatment) sorted by uplift predictions. Then the difference between these conversions is calculated
- **std** (*bool*) – If True, add columns with the uplift standard deviation and the response rate standard deviation. Default is False.
- **total** (*bool*) – If True, add the last row with the total values. Default is False. The total uplift is a weighted average uplift. See *weighted_average_uplift()*. The total response rate is a response rate on the full data amount.
- **bins** (*int*) – Determines the number of bins (and the relative percentile) in the data. Default is 10.

Returns DataFrame where metrics are by columns and percentiles are by rows.

Return type pandas.DataFrame

sklift.metrics.response_rate_by_percentile

sklift.metrics.metrics.**response_rate_by_percentile**(*y_true*, *uplift*, *treatment*, *group*, *strategy='overall'*, *bins=10*)

Compute response rate (target mean in the control or treatment group) at each percentile.

Parameters

- **y_true** (*1d array-like*) – Correct (true) target values.
- **uplift** (*1d array-like*) – Predicted uplift, as returned by a model.
- **treatment** (*1d array-like*) – Treatment labels.
- **group** (*string, ['treatment', 'control']*) – Group type for computing response rate: treatment or control.
  - **'treatment':** Values equal 1 in the treatment column.
  - **'control':** Values equal 0 in the treatment column.
- **strategy** (*string, ['overall', 'by_group']*) – Determines the calculating strategy. Default is 'overall'.

- **'overall':** The first step is taking the first k observations of all test data ordered by uplift prediction (overall both groups - control and treatment) and conversions in treatment and control groups calculated only on them. Then the difference between these conversions is calculated.
- **'by_group':** Separately calculates conversions in top k observations in each group (control and treatment) sorted by uplift predictions. Then the difference between these conversions is calculated.

- **bins** (*int*) – Determines the number of bins (and relative percentile) in the data. Default is 10.

Returns response rate at each percentile for control or treatment group, variance of the response rate at each percentile, group size at each percentile.

Return type array (shape = [>2]), array (shape = [>2]), array (shape = [>2])

sklift.metrics.treatment_balance_curve

sklift.metrics.metrics.**treatment_balance_curve**(*uplift*, *treatment*, *winsize*)

Compute the treatment balance curve: proportion of treatment group in the ordered predictions.

Parameters

- **uplift** (*1d array-like*) – Predicted uplift, as returned by a model.
- **treatment** (*1d array-like*) – Treatment labels.
- **winsize** (*int*) – Size of the sliding window for calculating the balance between treatment and control.

Returns Points on a curve.

Return type array (shape = [>2]), array (shape = [>2])

3.4.3 sklift.viz

sklift.viz.plot_uplift_preds

sklift.viz.base.**plot_uplift_preds**(*trmnt_preds*, *ctrl_preds*, *log=False*, *bins=100*)

Plot histograms of treatment, control and uplift predictions.

Parameters

- **trmnt_preds** (*1d array-like*) – Predictions for all observations if they are treatment.
- **ctrl_preds** (*1d array-like*) – Predictions for all observations if they are control.
- **log** (*bool, default False*) – Logarithm of source samples. Default is False.
- **bins** (*integer or sequence, default 100*) – Number of histogram bins to be used. If an integer is given, bins + 1 bin edges are calculated and returned. If bins is a sequence, gives bin edges, including left edge of first bin and right edge of last bin. In this case, bins is returned unmodified. Default is 100.

Returns Object that stores computed values.

sklift.viz.plot_qini_curve

sklift.viz.base.**plot_qini_curve**(*y_true*, *uplift*, *treatment*, *random=True*, *perfect=True*, *negative_effect=True*)

Plot Qini curves from predictions.

Parameters

- **y_true** (*1d array-like*) – Ground truth (correct) labels.
- **uplift** (*1d array-like*) – Predicted uplift, as returned by a model.
- **treatment** (*1d array-like*) – Treatment labels.
- **random** (*bool, default True*) – Draw a random curve. Default is True.
- **perfect** (*bool, default False*) – Draw a perfect curve. Default is True.
- **negative_effect** (*bool*) – If True, optimum Qini Curve contains the negative effects (negative uplift because of campaign). Otherwise, optimum Qini Curve will not contain the negative effects. Default is True.

Returns Object that stores computed values.

sklift.viz.plot_uplift_curve

sklift.viz.base.**plot_uplift_curve**(*y_true*, *uplift*, *treatment*, *random=True*, *perfect=True*)

Plot Uplift curves from predictions.

Parameters

- **y_true** (*1d array-like*) – Ground truth (correct) labels.
- **uplift** (*1d array-like*) – Predicted uplift, as returned by a model.
- **treatment** (*1d array-like*) – Treatment labels.
- **random** (*bool, default True*) – Draw a random curve. Default is True.
- **perfect** (*bool, default False*) – Draw a perfect curve. Default is True.

Returns Object that stores computed values.

sklift.viz.plot_treatment_balance_curve

sklift.viz.base.**plot_treatment_balance_curve**(*uplift*, *treatment*, *random=True*, *winsize=0.1*)

Plot Treatment Balance curve.

Parameters

- **uplift** (*1d array-like*) – Predicted uplift, as returned by a model.
- **treatment** (*1d array-like*) – Treatment labels.
- **random** (*bool, default True*) – Draw a random curve.
- **winsize** (*float, default 0.1*) – Size of the sliding window to apply. Should be between 0 and 1, extremes excluded.

Returns Object that stores computed values.

sklift.viz.plot_uplift_by_percentile

sklift.viz.base.**plot_uplift_by_percentile**(*y_true*, *uplift*, *treatment*, *strategy='overall'*, *kind='line'*, *bins=10*)

Plot uplift score, treatment response rate and control response rate at each percentile.

Treatment response rate ia a target mean in the treatment group. Control response rate is a target mean in the control group. Uplift score is a difference between treatment response rate and control response rate.

Parameters

- **y_true** (*1d array-like*) – Correct (true) target values.
- **uplift** (*1d array-like*) – Predicted uplift, as returned by a model.
- **treatment** (*1d array-like*) – Treatment labels.
- **strategy** (*string, ['overall', 'by_group']*) – Determines the calculating strategy. Default is 'overall'.
  - **'overall':** The first step is taking the first k observations of all test data ordered by uplift prediction (overall both groups - control and treatment) and conversions in treatment and control groups calculated only on them. Then the difference between these conversions is calculated.
  - **'by_group':** Separately calculates conversions in top k observations in each group (control and treatment) sorted by uplift predictions. Then the difference between these conversions is calculated.
- **kind** (*string, ['line', 'bar']*) – The type of plot to draw. Default is 'line'.
  - **'line':** Generates a line plot.
  - **'bar':** Generates a traditional bar-style plot.
- **bins** (*int*) – Determines number of bins (and the relative percentile) in the test data. Default is 10.

Returns Object that stores computed values.

3.5 Tutorials

3.5.1 Basic

It is better to start scikit-uplift from the basic tutorials.

The overview of the basic approaches to solving the Uplift Modeling problem

In English		nbviewer	github
In Russian		nbviewer	github

Example of usage model from sklift.models in sklearn.pipeline

In English		nbviewer	github
In Russian		nbviewer	github

3.6 Contributing to scikit-uplift

First off, thanks for taking the time to contribute!

All development is done on GitHub: https://github.com/maks-sh/scikit-uplift.

3.6.1 Submitting a bug report or a feature request

We use GitHub issues to track all bugs and feature requests. Feel free to open an issue if you have found a bug or wish to see a feature implemented at https://github.com/maks-sh/scikit-uplift/issues.

3.6.2 Contributing code

How to contribute

The code in the master branch should meet the current release. So, please make a pull request to the `dev` branch.

1. Fork the project repository.
2. Clone your fork of the scikit-uplift repo from your GitHub account to your local disk:

```
$ git clone git@github.com:YourLogin/scikit-uplift.git
$ cd scikit-learn
```

3. Add the upstream remote. This saves a reference to the main scikit-uplift repository, which you can use to keep your repository synchronized with the latest changes:

```
$ git remote add upstream https://github.com/maks-sh/scikit-uplift.git
```

4. Synchronize your `dev` branch with the upstream `dev` branch:

```
$ git checkout dev
$ git pull upstream dev
```

5. Create a feature branch to hold your development changes:

```
$ git checkout -b feature/my_new_feature
```

   and start making changes. Always use a feature branch. It's a good practice.

6. Develop the feature on your feature branch on your computer, using Git to do the version control. When you're done editing, add changed files using `git add` and then `git commit`. Then push the changes to your GitHub account with:

```
$ git push -u origin feature/my_new_feature
```

7. Create a pull request from your fork into `dev` branch.

Styleguides

Python

We follow the PEP8 style guide for Python. Docstrings follow google style.

Git Commit Messages

- Use the present tense ("Add feature" not "Added feature")
- Use the imperative mood ("Move file to…" not "Moves file to…")
- Limit the first line to 72 characters or less
- Reference issues and pull requests liberally after the first line
- If you want to use emojis, use them at the beginning of the line.

3.7 Release History

3.7.1 Legend for changelogs

- something big that you couldn't do before.
- something that you couldn't do before.
- a miscellaneous minor improvement.
- something that previously didn't work as documentated – or according to reasonable expectations – should now work.
- you will need to change your code to have the same effect in the future; or a feature will be removed in the future.

3.7.2 Version 0.2.0

User Guide

- Add User Guide

sklift.models

- Add `treatment interaction` method to SoloModel approach by @AdiVarma27.

sklift.metrics

- Add uplift_by_percentile function by @ElisovaIra.
- Add weighted_average_uplift function by @ElisovaIra.
- Add perfect_uplift_curve function.
- Add perfect_qini_curve function.
- Add normalization in uplift_auc_score and qini_auc_score functions.
- Remove metrics `auuc` and `auqc`. In exchange for them use respectively uplift_auc_score and qini_auc_score

sklift.viz

- Add plot_uplift_curve function.
- Add plot_qini_curve function.
- Remove `plot_uplift_qini_curves`.

Miscellaneous

- Add contributors in main Readme and in main page of docs.
- Add contributing guide.
- Add code of conduct.
- Reformat Tutorials page.
- Add github buttons in docs.
- Add logo compatibility with pypi.

3.7.3 Version 0.1.2

sklift.models

- Fix bugs in TwoModels for regression problem.
- Minor code refactoring.

sklift.metrics

- Minor code refactoring.

sklift.viz

- Add bar plot in plot_uplift_by_percentile by @ElisovaIra.
- Fix bug in plot_uplift_by_percentile.
- Minor code refactoring.

3.7.4 Version 0.1.1

sklift.viz

- Add plot_uplift_by_percentile by @ElisovaIra.
- Fix bug with import plot_treatment_balance_curve.

sklift.metrics

- Add response_rate_by_percentile by @ElisovaIra.
- Fix bug with import uplift_auc_score and qini_auc_score.
- Fix typos in docstrings.

Miscellaneous

- Add tutorial "Example of usage model from sklift.models in sklearn.pipeline".
- Add link to Release History in main Readme.md.

3.7.5 Version 0.1.0

sklift.models

- Fix typo in TwoModels docstring by @spiaz.
- Improve docstrings and add references to all approaches.

sklift.metrics

- Add treatment_balance_curve by @spiaz.
- The metrics `auuc` and `auqc` are now respectively renamed to uplift_auc_score and qini_auc_score. So, `auuc` and `auqc` will be removed in 0.2.0.
- Add a new parameter `startegy` in uplift_at_k.

sklift.viz

- Add plot_treatment_balance_curve by @spiaz.
- fix typo in plot_uplift_qini_curves by @spiaz.

Miscellaneous

- Remove sklift.preprocess submodule.
- Add compatibility of tutorials with colab and add colab buttons by @ElMaxuno.
- Add Changelog.
- Change the documentation structure. Add next pages: Tutorials, Release History and Hall of fame.

3.8 Hall of Fame

Here are the links to the competitions, names of the winners and to their solutions, where scikit-uplift was used.

3.8.1 X5 RetailHero Uplift Modeling contest

2. **Kirill Liksakov** solution

CHAPTER 4

Papers and materials

1. **Gutierrez, P., & Gérardy, J. Y.** Causal Inference and Uplift Modelling: A Review of the Literature. In International Conference on Predictive Applications and APIs (pp. 1-13).
2. **Artem Betlei, Criteo Research; Eustache Diemert, Criteo Research; Massih-Reza Amini, Univ. Grenoble Alpes** Dependent and Shared Data Representations improve Uplift Prediction in Imbalanced Treatment Conditions FAIM'18 Workshop on CausalML.
3. **Eustache Diemert, Artem Betlei, Christophe Renaudin, and Massih-Reza Amini. 2018.** A Large Scale Benchmark for Uplift Modeling. In Proceedings of AdKDD & TargetAd (ADKDD'18). ACM, New York, NY, USA, 6 pages.
4. **Athey, Susan, and Imbens, Guido. 2015.** Machine learning methods for estimating heterogeneous causal effects. Preprint, arXiv:1504.01132. Google Scholar.
5. **Oscar Mesalles Naranjo. 2012.** Testing a New Metric for Uplift Models. Dissertation Presented for the Degree of MSc in Statistics and Operational Research.
6. **Kane, K., V. S. Y. Lo, and J. Zheng. 2014.** Mining for the Truly Responsive Customers and Prospects Using True-Lift Modeling: Comparison of New and Existing Methods. Journal of Marketing Analytics 2 (4): 218–238.
7. **Maciej Jaskowski and Szymon Jaroszewicz.** Uplift modeling for clinical trial data. ICML Workshop on Clinical Data Analysis, 2012.
8. **Lo, Victor. 2002.** The True Lift Model - A Novel Data Mining Approach to Response Modeling in Database Marketing. SIGKDD Explorations. 4. 78-86.
9. **Zhao, Yan & Fang, Xiao & Simchi-Levi, David. 2017.** Uplift Modeling with Multiple Treatments and General Response Types. 10.1137/1.9781611974973.66.
10. **Nicholas J Radcliffe. 2007.** Using control groups to target on predicted lift: Building and assessing uplift model. Direct Marketing Analytics Journal, (3):14–21, 2007.
11. **Devriendt, F., Guns, T., & Verbeke, W. 2020.** Learning to rank for uplift modeling. ArXiv, abs/2002.05897.

CHAPTER 5

Tags

EN: uplift modeling, uplift modelling, causal inference, causal effect, causality, individual treatment effect, true lift, net lift, incremental modeling

RU: , Uplift

ZH: ,,,,,,

Index