Modern approaches to the assessment of demand sensitivity based on uplift models and neural networks

D. Ustinova, A. Kurmukova, E. Avdotin, K. Onlabek

Skolkovo Institute of Science and Technology

ML 2021 final project defence

March 21, 2021

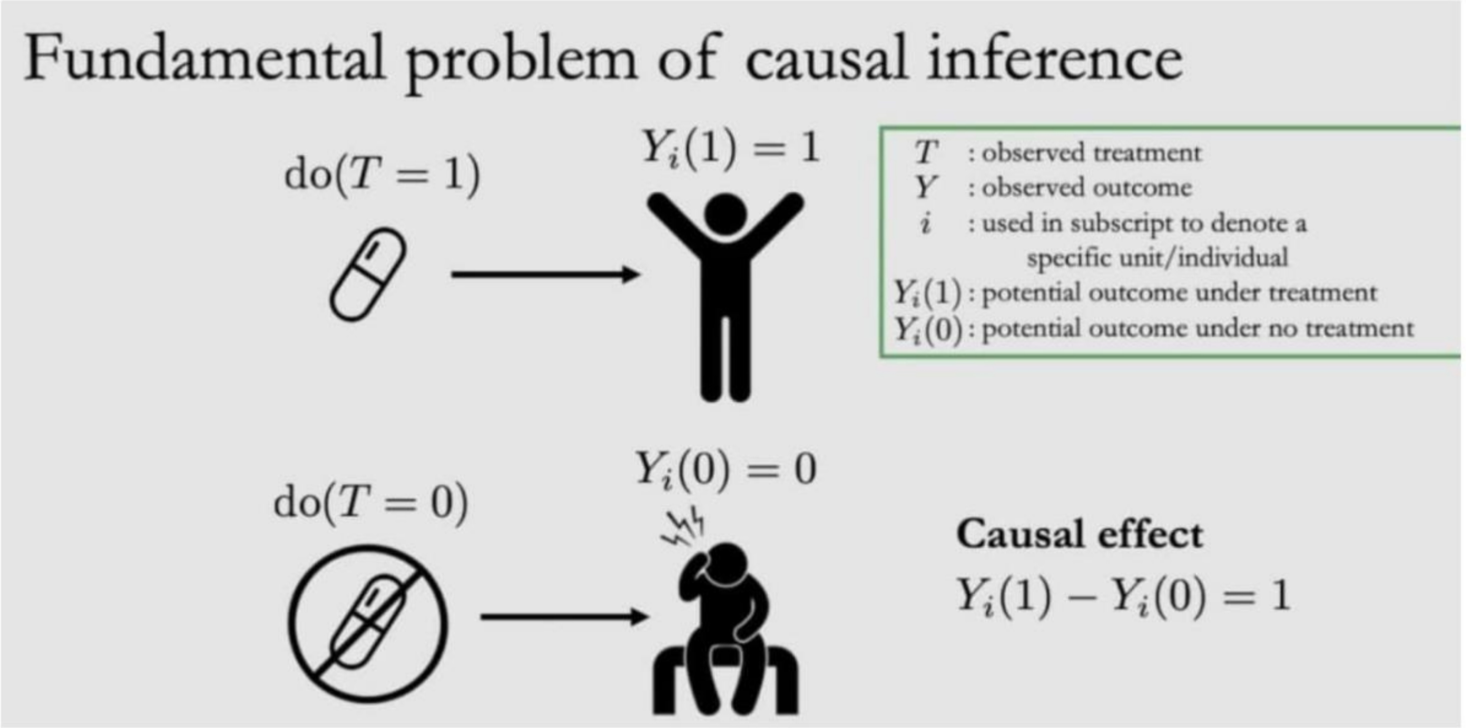
Moscow, Russia

Problem statement:

- **Uplift modeling**: predicts the incremental effect of a treatment on an individual's behaviour.
- **Example**: a company wants to send a customer an SMS with a discount offer.
- **Goal**: to detect the persuadable group of people and offer them the discount.

Response if Treated	N	**Do-Not-Disturb** ***c***	**Lost Cause** ***d***
	Y	Sure Thing *b*	Persuadable *a*
		Y	N
		Response if <u>not</u> treated	

Problem statement:



- Can't take and don't take the pill simultaneously $\implies$ it is not possible to observe and count the uplift (or casual effect) directly.

Solution:

In classical approach:

- Casual effect:

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

- CATE (Conditional Average Treatment Effect):

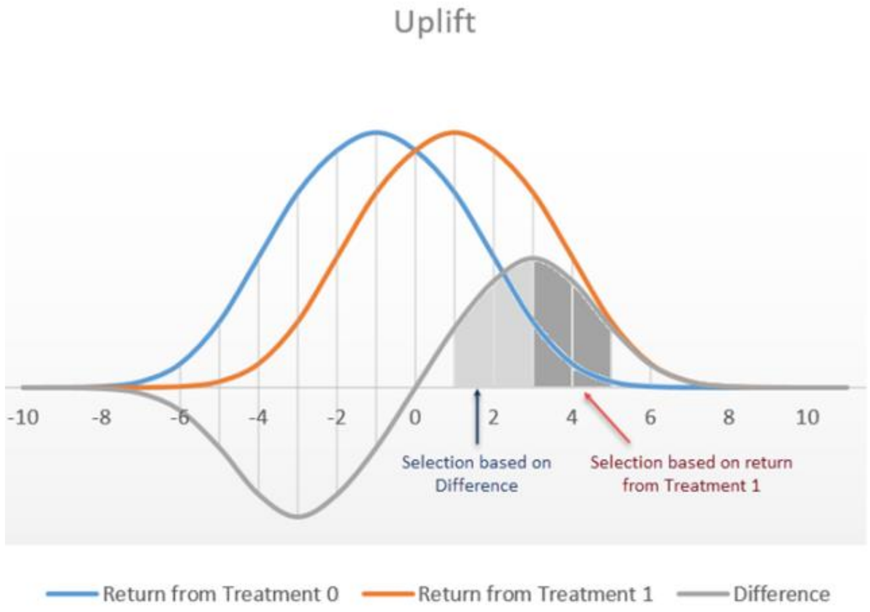
$$CATE = \mathsf{E}\left[Y_i^1 \mid X_i\right] - \mathsf{E}\left[Y_i^0 \mid X_i\right].$$

In uplift modelling:

$$uplift = \widehat{CATE} = \mathsf{E}\left[Y_i \mid X_i = x, W_i = 1\right] -$$

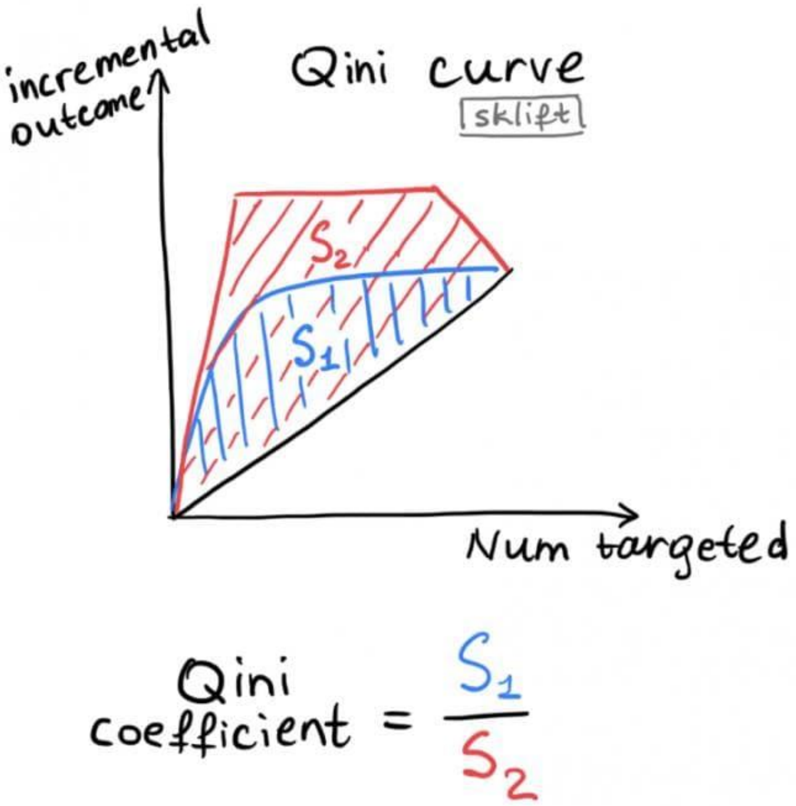
$$-\mathsf{E}\left[Y_i \mid X_i = x, W_i = 0\right],$$

where $Y_i = W_i Y_i^1 + (1 - W_i) Y_i^0$ is customer's observed response.



Uplift metrics

- uplift@k – the value of uplift at top k% of data
- area under uplift curve:
- area under qini curve



$$uplift\_curve(t) = \left(\frac{Y_t^T}{N_t^T} - \frac{Y_t^C}{N_t^C}\right)(N_t^T + N_t^C)$$

$$qini\_curve(t) = Y_t^T - \frac{Y_t^C N_t^T}{N_t^C},$$

where Y_t^T and Y_t^C are targets in treatment and control groups, respectively. N_t^T and N_t^T are sizes of groups and t is the number of targeted objects.

https://m.habr.com/ru/company/ru_mts/blog/538934/

Meta-learners: S-learner

S-learner calculates the treatment effect using a single machine learning model

- fitting

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

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

- inference

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

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

https://habr.com/ru/company/ru_mts/blog/485980/

Meta-learners: T-learner

T-learner is a Two-Model approach

- fitting

$$model^T = fit\left(\underbrace{\begin{matrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{p1} & \cdots & x_{pk} \end{matrix}}_{X_{train\_treat}}, \quad \underbrace{\begin{matrix} y_1 \\ \cdots \\ y_p \end{matrix}}_{Y_{train\_treat}}\right), \quad model^C = fit\left(\underbrace{\begin{matrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{q1} & \cdots & x_{qk} \end{matrix}}_{X_{train\_control}}, \quad \underbrace{\begin{matrix} y_1 \\ \cdots \\ y_q \end{matrix}}_{Y_{train\_control}}\right)$$

- inference

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

https://habr.com/ru/company/ru_mts/blog/485980/

Meta-learners: X-learner

$$model^T = fit\left(\underset{X_{train\_treat}}{\begin{matrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{p1} & \cdots & x_{pk} \end{matrix}},\ \underset{Y_{train\_treat}}{\begin{matrix} y_1 \\ \cdots \\ y_p \end{matrix}}\right),\ model^C = fit\left(\underset{X_{train\_control}}{\begin{matrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{q1} & \cdots & x_{qk} \end{matrix}},\ \underset{Y_{train\_control}}{\begin{matrix} y_1 \\ \cdots \\ y_q \end{matrix}}\right)$$

$$model^T_{new} = fit\left(\underset{X_{train\_treat}}{\begin{matrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{p1} & \cdots & x_{pk} \end{matrix}},\ \underset{\widetilde{D}^T}{\begin{matrix} \tilde{d}^T_1 \\ \cdots \\ \tilde{d}^T_p \end{matrix}}\right),\ model^C_{new} = fit\left(\underset{X_{train\_control}}{\begin{matrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{q1} & \cdots & x_{qk} \end{matrix}},\ \underset{\widetilde{D}^C}{\begin{matrix} \tilde{d}^C_1 \\ \cdots \\ \tilde{d}^C_q \end{matrix}}\right)$$

$$\hat{Y}^C = \begin{matrix} model^c \\ predict \end{matrix}\underset{X_{train\_treat}}{\begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{p1} & \cdots & x_{pk} \end{pmatrix}};\ \hat{Y}^T = \begin{matrix} model^T \\ predict \\ proba \end{matrix}\underset{X_{train\_control}}{\begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{q1} & \cdots & x_{qk} \end{pmatrix}}$$

$$\widetilde{D}^T = Y_{train\_treat} - \hat{Y}^C;\ \widetilde{D}^C = \hat{Y}^T - Y_{train\_control}$$

$$g \cdot \begin{matrix} model^C_{new} \\ predict \end{matrix}\underset{X_{test}}{\begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mk} \end{pmatrix}} + (1-g) \cdot \begin{matrix} model^T_{new} \\ predict \end{matrix}\underset{X_{test}}{\begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mk} \end{pmatrix}} = \underset{uplift}{\begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix}}$$

https://habr.com/ru/company/ru_mts/blog/485980/

Meta-learners: R-learner

R-learner is a two-step algorithm

- Fitting two base models using cross-validation and predict on holdout fold i:
  - the outcomes m(−i)(Xi)
  - the propensity scores e(−i)(Xi)

- Minimising of R-loss:

$$\tau^*() = argmin_\tau\{\frac{1}{n}\sum_{i=1}^{n}[(Y_i - \hat{m}^{(-i)}(X_i)) - $$
$$-(W_i - \hat{e}^{(-i)}(X_i))\tau(X_i)]^2 + \Lambda(\tau())\},$$

Tree-based approaches: Uplift Tree

The tree is constructed to maximize the difference between distribution divergence for the treatment and control groups before and after the split. The best split in each node is chosen based on the gain:

$$D_{gain} = D_{aftersplit}(P^T, P^C) - D_{beforesplit}(P^T, P^C).$$

Three approaches to calculate the divergence:

- the Kullback-Leibler (KL) divergence:

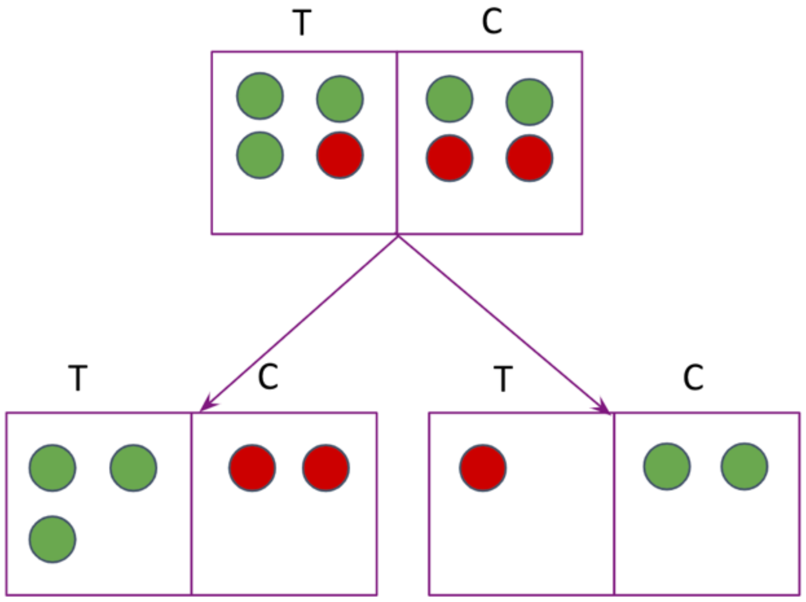
$$KL(P : Q) = \sum_i p_i \log \frac{p_i}{q_i};$$

- the Euclidean distance:

$$ED(P : Q) = \sum_i (p_i - q_i)^2;$$

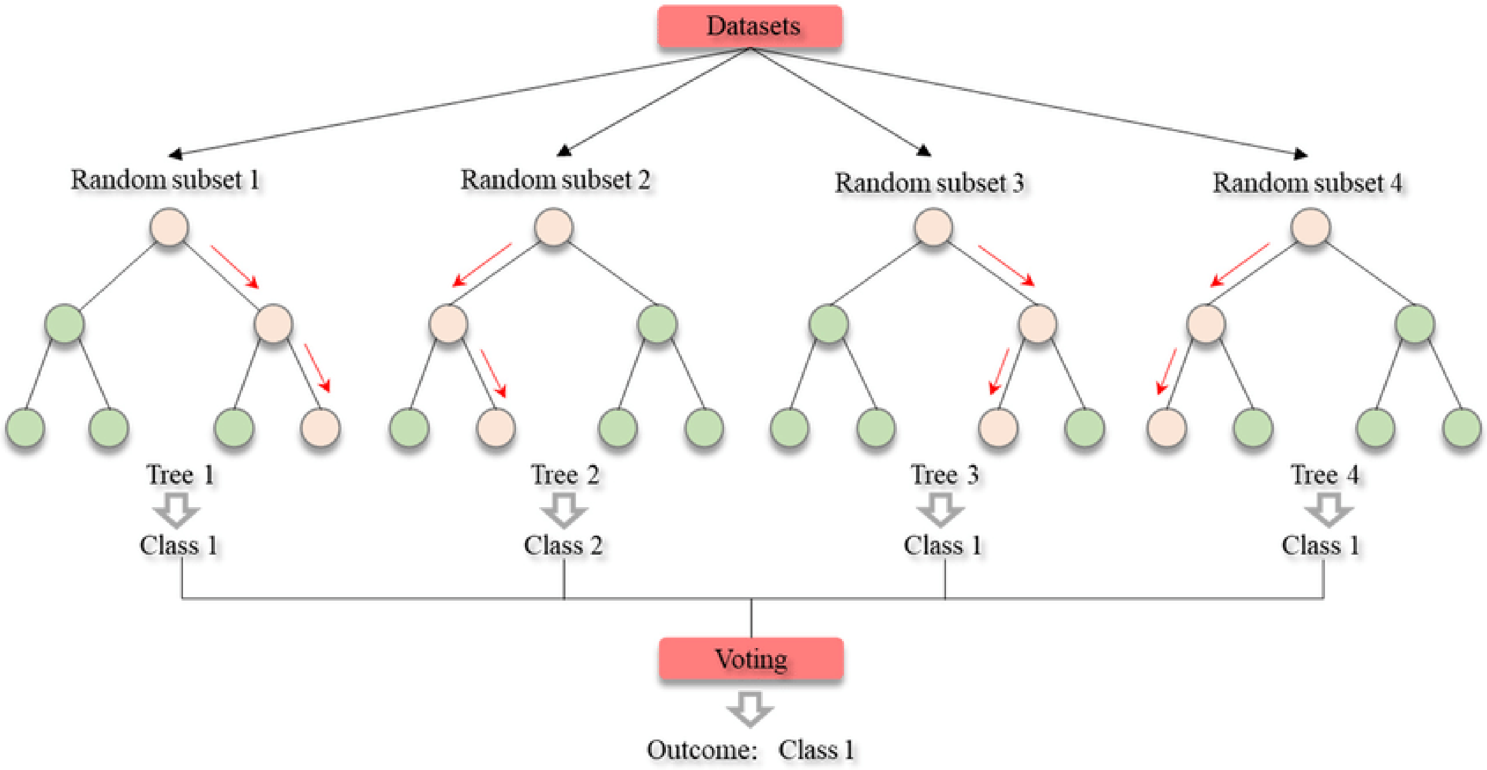
- the Chi-squared divergence:

$$\xi^2(P : Q) = \sum_i \frac{(p_i - q_i)^2}{q_i}.$$



https://tech.wayfair.com/data-science/2019/10/modeling-uplift-directly-uplift-decision-tree-with-kl-divergence-and-euclidean-distance-as-splitting-criteria/

Tree-based approaches: Random Forest



DOI: 10.3390/su11216159

Boosting

In the project we considered Uplift AdaBoost algorithm:

1. Initialize weights $w_{1,i}^T, w_{1,i}^C$
2. For $m \leftarrow 1, \ldots, M$
   (a) $w_{m,i}^T \leftarrow \frac{w_{m,i}^T}{\sum_j w_{m,j}^T + \sum_j w_{m,j}^C}$; $w_{m,i}^C \leftarrow \frac{w_{m,i}^C}{\sum_j w_{m,j}^T + \sum_j w_{m,j}^C}$
   (b) Build a base model h_m on $\mathcal{D}$ with $w_{m,i}^T, w_{m,i}^C$
   (c) Compute the treatment and control errors ϵ_m^T, ϵ_m^C
   (d) Compute $\beta_m^T(\epsilon_m^T, \epsilon_m^C)$, $\beta_m^C(\epsilon_m^T, \epsilon_m^C)$
   (e) If $\beta_m^T = \beta_m^C = 1$ or $\epsilon_m^T \notin (0, \frac{1}{2})$ or $\epsilon_m^C \notin (0, \frac{1}{2})$:
      i. choose random weights $w_{m,i}^T$, $w_{m,i}^C$
      ii. continue with next boosting iteration
   (f) $w_{m+1,i}^T \leftarrow w_{m,i}^T \cdot (\beta_m^T)^{1[h_m(x_i^T) = y_i^T]}$
   (g) $w_{m+1,i}^C \leftarrow w_{m,i}^C \cdot (\beta_m^C)^{1[h_m(x_i^C) = 1 - y_i^C]}$
   (h) $\beta_m \leftarrow \min\{\beta_m^T, \beta_m^C\}$
   (i) Add h_m with coefficient β_m to the ensemble

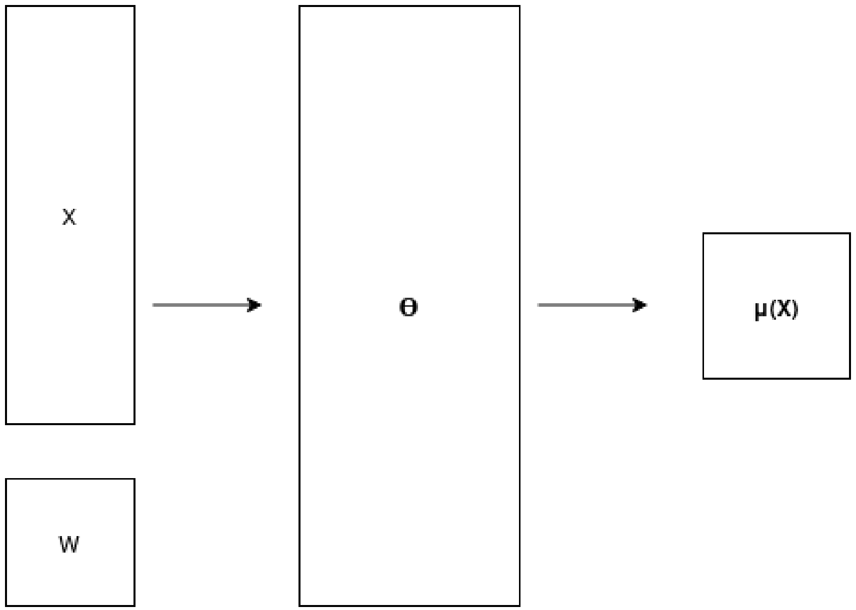
Output: The final hypothesis

$$h_f(x) = \begin{cases} 1 & if \ \sum_{m=1}^M \left(\log \frac{1}{\beta_m}\right) h_m(x) \geq \frac{1}{2} \sum_{m=1}^M \log \frac{1}{\beta_m}, \\ 0 & otherwise. \end{cases} \quad (2)$$

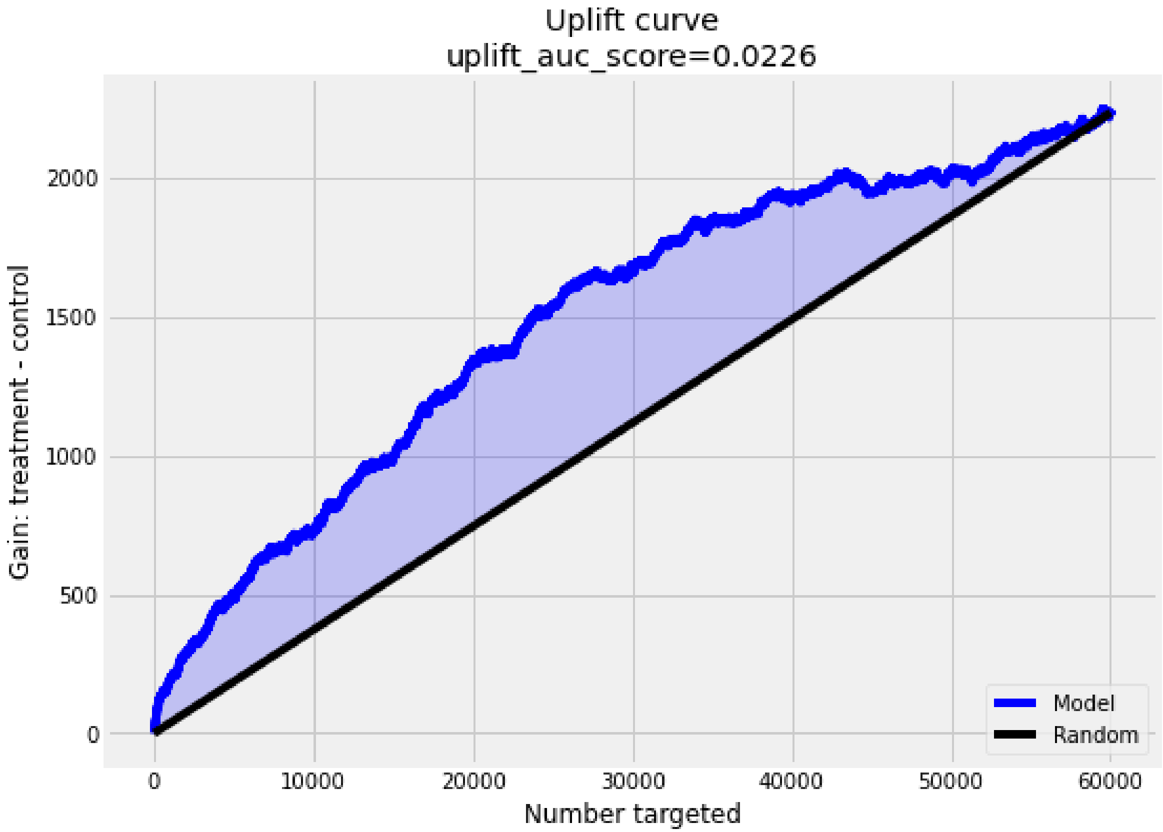
Neural Network for uplift modeling

- Transformed outcome is used as target
- MLP architecture
- Transformed output depending on treatment
- Training and validation on X5 dataset

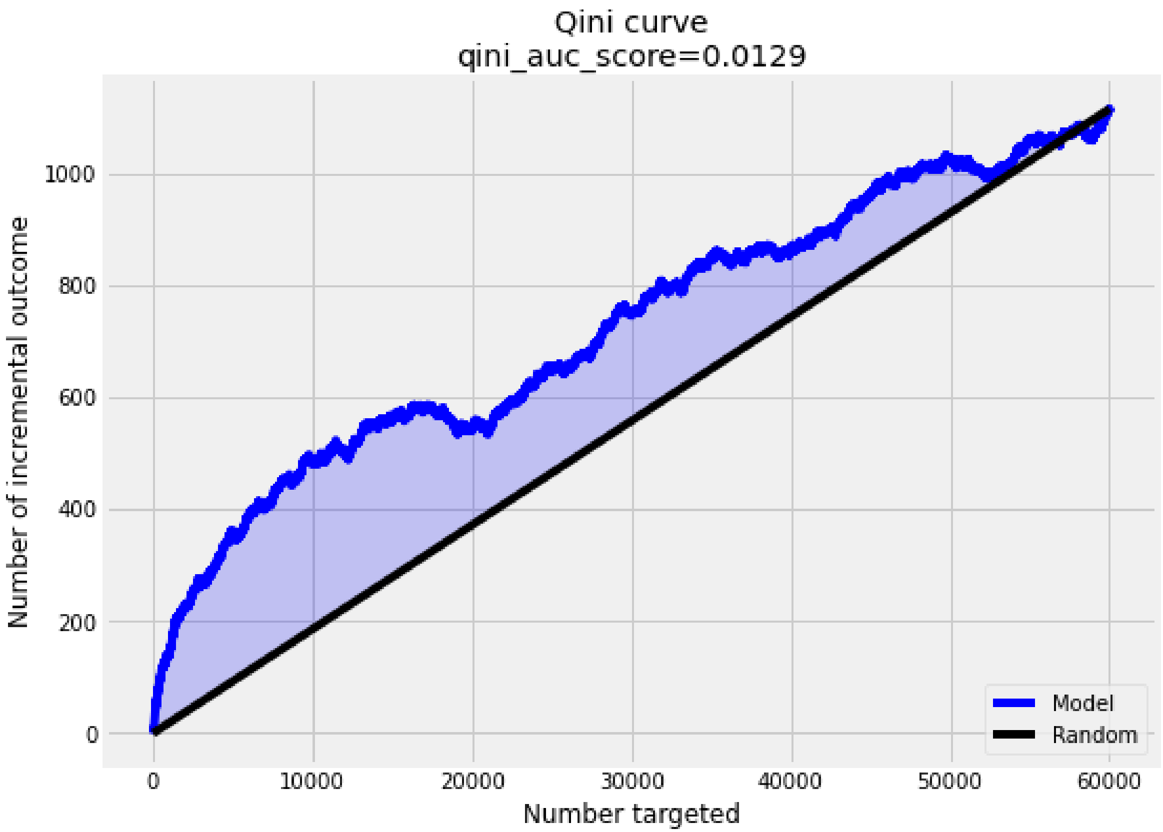
$$\hat{y}_i = \begin{cases} \mu(x_i), & \text{if } w_i = 1 \\ -\mu(x_i), & \text{if } w_i = 0 \end{cases},$$



Results: meta-learners

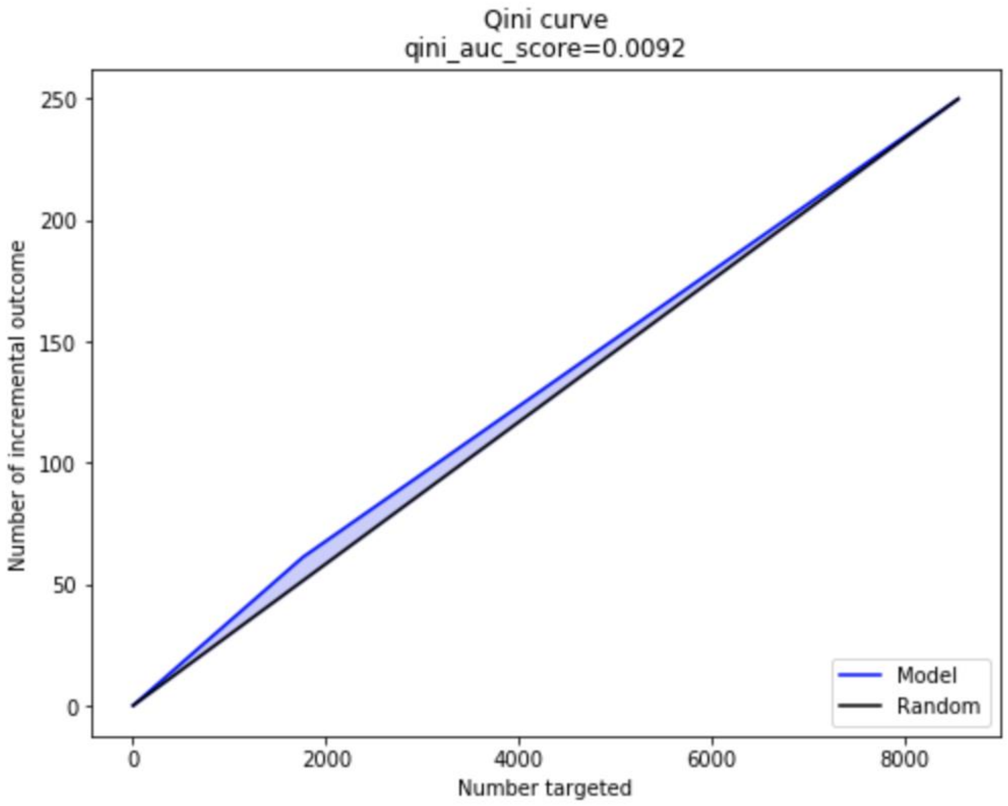


Uplift curve for S learner
on X5 dataset

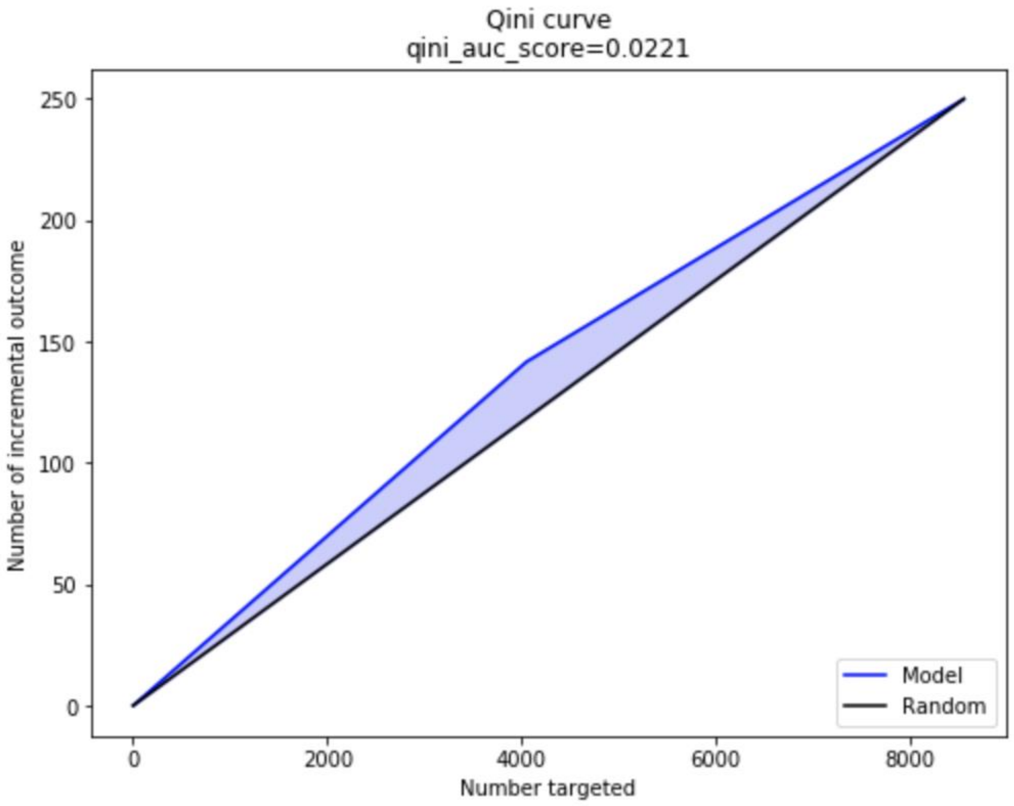


Qini curve for T learner on
X5 dataset

Results: random forest and boosting

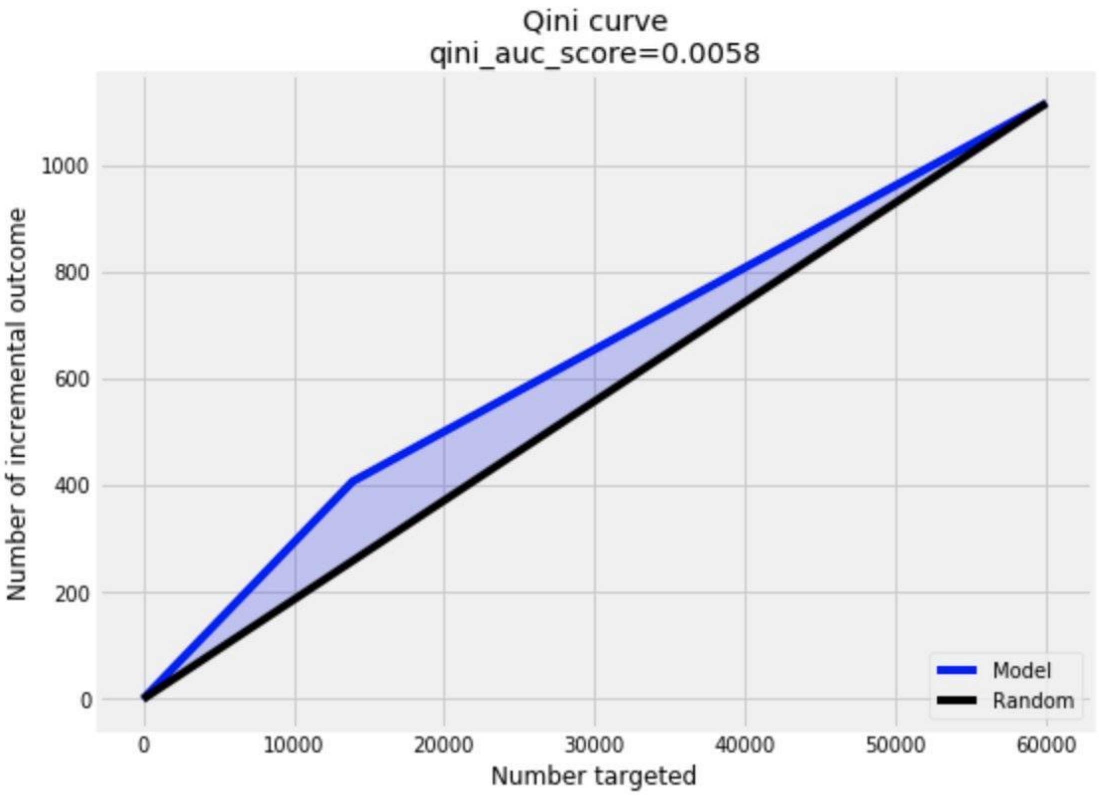


Qini curve for random forest
On MineThatData

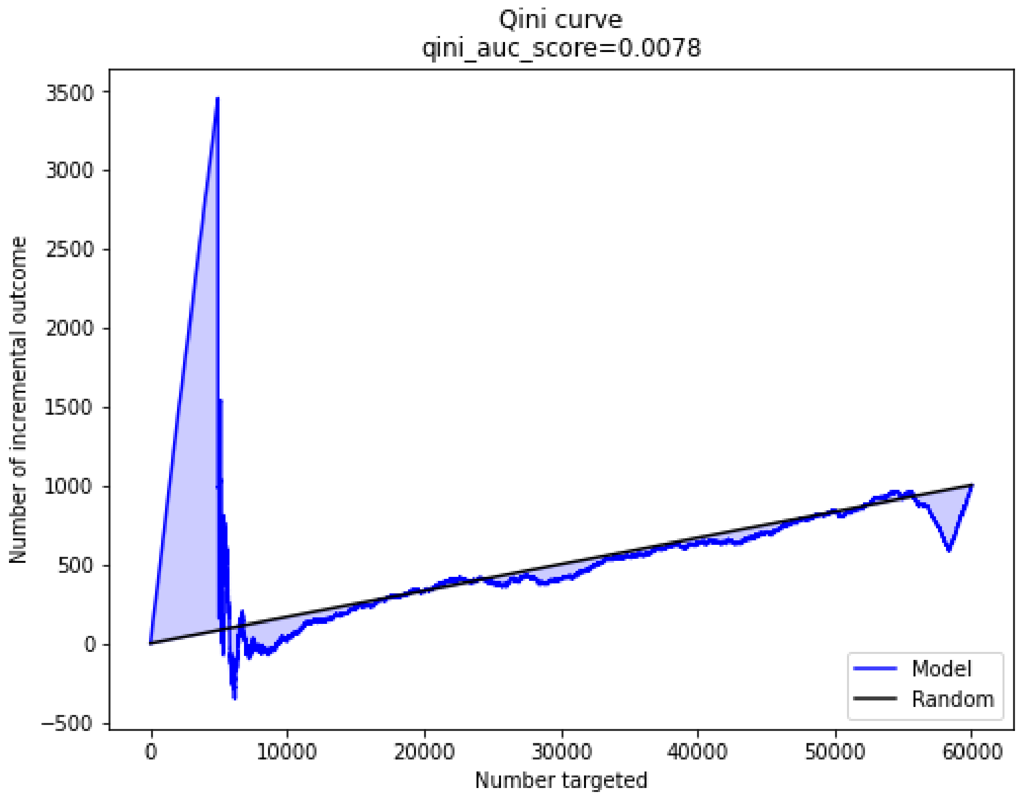


Qini curve for boosting
On MineThatData

Results: trees and neural network



Qini curve for tree with
depth 8, X5 dataset



Qini curve for
Neural Network, X5 dataset

Conclusion

- Meta-learners (S-, T-, X- and R-learners)
- Tree-based approaches (Uplift Tree, Random Forest)
- Boosting (AdaBoost)
- Neural networks
- Comparison on 3 datasets

Thank you for your attention!