

# Drzewa Wzmacniane Gradientowo

## Systemy uczące się - laboratorium

Mateusz Lango

Zakład Inteligentnych Systemów Wspomagania Decyzji  
Wydział Informatyki i Telekomunikacji  
Politechnika Poznańska

„Akademia Innowacyjnych Zastosowań Technologii Cyfrowych (AI Tech)”,  
projekt finansowany ze środków Programu Operacyjnego Polska Cyfrowa POPC.03.02.00-00-0001/20



**Fundusze  
Europejskie**  
Polska Cyfrowa

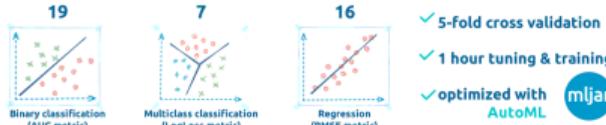


**Rzeczpospolita  
Polska**

**Unia Europejska**  
Europejski Fundusz  
Rozwoju Regionalnego



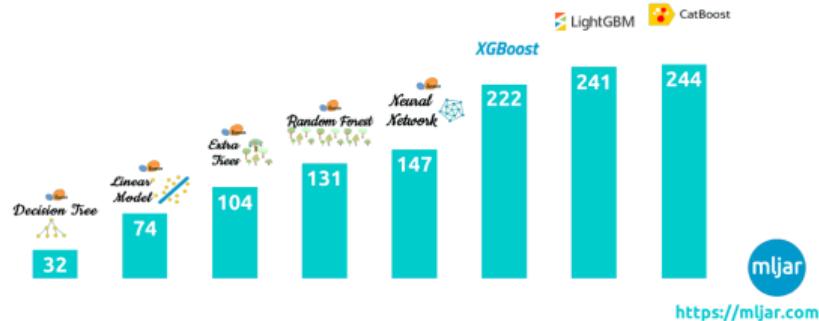
# Dlaczego warto znać wzmacnianie gradientowe (ang. *gradient boosting*)?



Number of wins in pairwise comparison

vs.	Decision Tree	Linear Model	Extra Trees	Random Forest	Neural Network	XGBoost	LightGBM	CatBoost
decision tree	0	23	40	40	36	41	42	42
linear	19	0	33	32	35	35	35	36
extra trees	3	10	0	33	30	39	41	41
random forest	2	10	9	0	22	38	41	41
neural network	6	8	13	20	0	34	35	36
xgboost	1	8	4	4	9	0	26	26
lightgbm	1	8	3	1	8	17	0	22
catboost	0	7	2	1	7	18	21	0

Total number of wins



# AdaBoost - pytania kontrolne

## Problem

*Jak działa algorytm AdaBoost?*

## Problem

*Porównaj AdaBoost z algorytmem bagging. Jakie są jego wady i zalety?*

## Problem

*Jak zmienia się suma wag błędnie klasyfikowanych przykładów w kolejnych iteracjach?*

## Problem

*Co to jest weak learner? Jakie algorytmy są dobrymi klasyfikatorami bazowymi dla boostingu?*

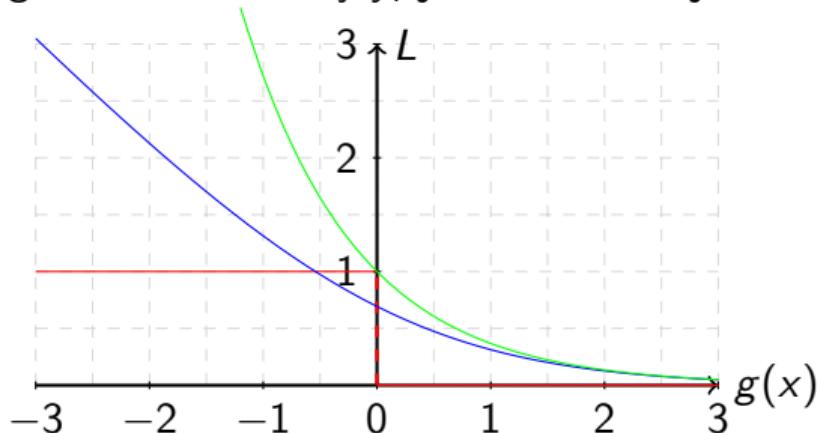
## Problem

*Jak zachowuje się boosting jeśli klasyfikator nie popełnia już żadnych błędów?*

# Eksponencjalna funkcja straty

$$\min_{\theta} \sum_{i=1}^n e^{-y_i g(x_i; \theta)}$$

gdzie wartość klasy  $y_i$  jest kodowana jako -1 i 1.



- Błąd tej postaci nazywamy błędem eksponencjalnym (ang. *exponential loss*).
- Zauważ, że postać błędu jest ogólna i za  $g(x)$  można wstawić inny model wiedzy np. wyrażenie liniowe, uzyskując inny algorytm wykorzystujący tę funkcję błędu.

# AdaBoost - przypomnienie

- ① Ustaw wagę przykładów na  $w_i = \frac{1}{N}$
- ② Dla każdego klasyfikatora
  - ① Wylosuj próbkę z rozkładu wag (lub użyj odpowiedniego klasyfikatora)
  - ② Oblicz ważony błąd klasyfikacji  $\epsilon_k$
  - ③ Ustaw wagę klasyfikatora  $\alpha_k$  bazując na jego błędzie
  - ④ Zwięksź wagę  $w_i$  błędnie klasyfikowanym przykładom

Ostatecznie decyzja jest podjmowana jako:  $y_m(x) = sign(\sum_{k=1}^m \alpha_k y_k(x))$

# AdaBoost - przypomnienie

- ① Ustaw wagi przykładów na  $w_i = \frac{1}{N}$
- ② Dla każdego klasyfikatora
  - ① Stwórz klasyfikator minimalizując

$$J_m = \sum_{n=1}^N w_i I[y_k \neq \hat{y}_k]$$

- ② Oblicz ważony błąd klasyfikacji  $\epsilon_k$
- ③ Ustaw wagę klasyfikatora  $\alpha_k$  bazując na jego błędzie
- ④ Zwięksź wagi  $w_i$  błędnie klasyfikowanym przykładom

Ostatecznie decyzja jest podjmowana jako:  $y_m(x) = \text{sign}(\sum_{k=1}^m \alpha_k y_k(x))$

# AdaBoost - przypomnienie

- ① Ustaw wagę przykładów na  $w_i = \frac{1}{N}$
- ② Dla każdego klasyfikatora
  - ① Stwórz klasyfikator minimalizując

$$J_m = \sum_{n=1}^N w_i I[y_k \neq \hat{y}_k]$$

- ② Oblicz ważony błąd klasyfikacji

$$\epsilon_k = \frac{\sum_{i=1}^n w_i I[y \neq \hat{y}]}{\sum_{i=1}^n w_i}$$

- ③ Ustaw wagę klasyfikatora  $\alpha_k$  bazując na jego błędzie
- ④ Zwięksź wagę  $w_i$  błędnie klasyfikowanym przykładom

Ostatecznie decyzja jest podjmowana jako:  $y_m(x) = \text{sign}(\sum_{k=1}^m \alpha_k y_k(x))$

# AdaBoost - przypomnienie

- ① Ustaw wagi przykładów na  $w_i = \frac{1}{N}$
- ② Dla każdego klasyfikatora
  - ① Stwórz klasyfikator minimalizując

$$J_m = \sum_{n=1}^N w_i I[y_k \neq \hat{y}_k]$$

- ② Oblicz ważony błąd klasyfikacji

$$\epsilon_k = \frac{\sum_{i=1}^n w_i I[y \neq \hat{y}]}{\sum_{i=1}^n w_i}$$

- ③ Ustaw wagę klasyfikatora  $\alpha_k$  na

$$\alpha_k = \ln \frac{1 - \epsilon_k}{\epsilon_k}$$

- ④ Zwięksź wagi  $w_i$  błędnie klasyfikowanym przykładom

Ostatecznie decyzja jest podjmowana jako:  $y_m(x) = \text{sign}(\sum_{k=1}^m \alpha_k y_k(x))$

# AdaBoost - przypomnienie

- ① Ustaw wagi przykładów na  $w_i = \frac{1}{N}$
- ② Dla każdego klasyfikatora
  - ① Stwórz klasyfikator minimalizując

$$J_m = \sum_{n=1}^N w_i I[y_k \neq \hat{y}_k]$$

- ② Oblicz ważony błąd klasyfikacji

$$\epsilon_k = \frac{\sum_{i=1}^n w_i I[y \neq \hat{y}]}{\sum_{i=1}^n w_i}$$

- ③ Ustaw wagę klasyfikatora  $\alpha_k$  na

$$\alpha_k = \ln \frac{1 - \epsilon_k}{\epsilon_k}$$

- ④ Zaktualizuj wagi

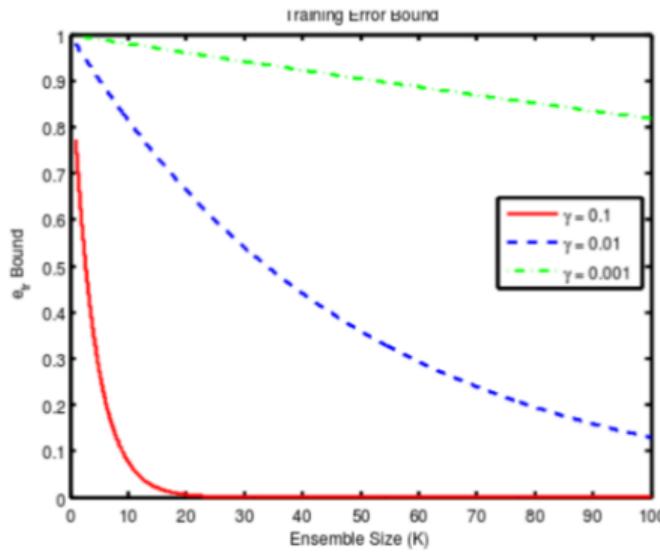
$$w_i = w_i \exp(\alpha_k I[\hat{y}_k(x_i) \neq y_i])$$

Ostatecznie decyzja jest podjmowana jako:  $y_m(x) = \text{sign}(\sum_{k=1}^m \alpha_k y_k(x))$

# AdaBoost - własności

## Theorem

Zakładając, że w każdej iteracji otrzymamy hipotezę dla której  $\epsilon_i \leq 1/2 - \epsilon$  błąd uczący jest co najwyżej  $\exp(-2\epsilon^2 T)$  czyli maleje eksponencjalnie!

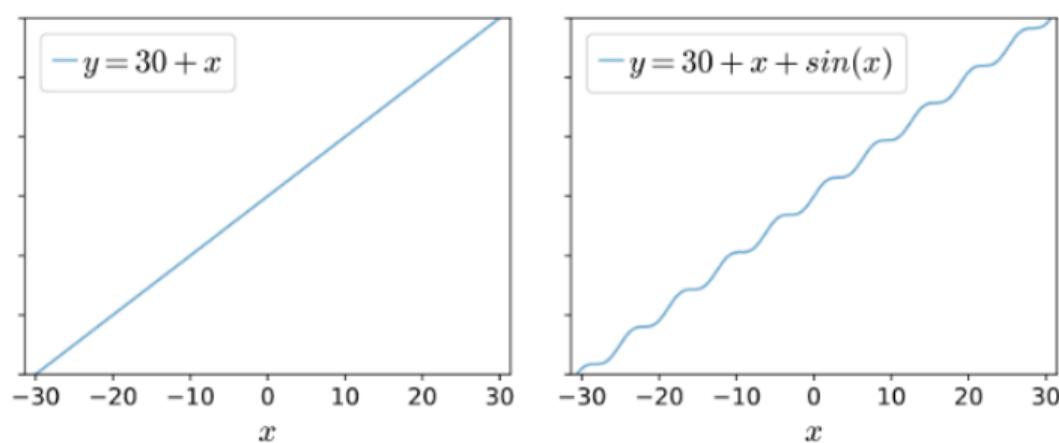
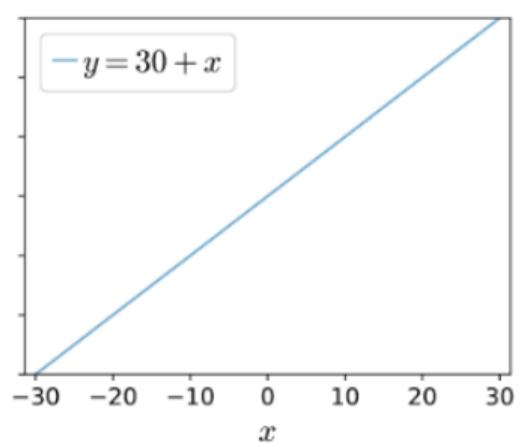
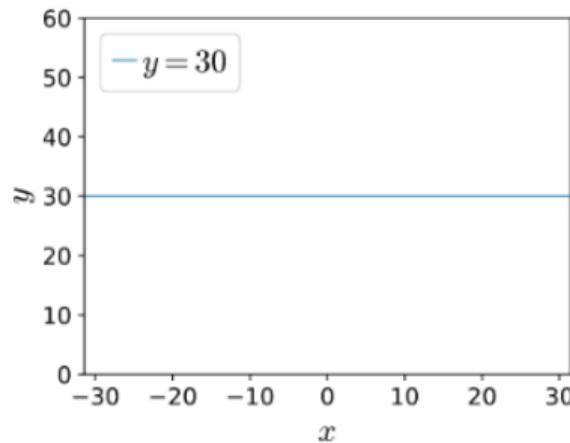
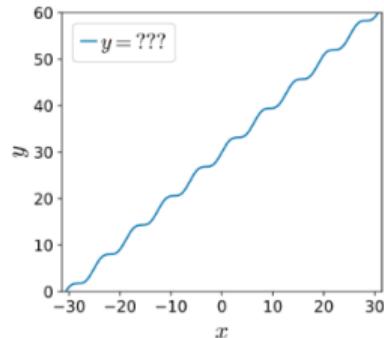


+ Dużo innych fajnych twierdzeń ;)

# Motywacja

- Strata eksponencjalna optymalizowana przez AdaBoost nie jest odpowiednia dla danych z dużą ilością szumu w etykietach czy z mocno nakładającymi się klasami
- W takich sytuacjach lepsze wyniki możemy osiągnąć np. stratą logistyczną
- Dlaczego nie zaprojektować boostingu, który optymalizuje taką funkcję straty?

# Modelowanie addytywne



$$F(x) = \sum_{m=1}^M w_m f_m(x; \theta_m)$$

$$f_1 = 30 \quad f_2 = x \quad f_3 = \sin(x)$$

# Dwa sposoby treningu modeli addytywnych

$$F(x) = \sum_{m=1}^M w_m f_m(x; \theta_m)$$

- trening łączny

$$\min_{w_1, w_2, \dots, w_M, \theta_1, \theta_2, \dots, \theta_M} \sum_{i=1}^N L(y_i, F(x_i)) = \sum_{i=1}^N L\left(y_i, \sum_{m=1}^M w_m f_m(x_i; \theta_m)\right)$$

- trening przyrostowy

$$\min_{w_t, \theta_t} \sum_{i=1}^N L\left(y_i, \sum_{m=1}^{t-1} w_m f_m(x_i; \theta_m) + w_t f_t(x_i; \theta_t)\right)$$

# Modelowanie addytywne - uczenie przyrostowe

Model powinien mieć postać<sup>1</sup>

$$\begin{aligned}\hat{y} &= F_M(x) \\ &= f_0(x) + f_1(x) + f_2(x) + \dots + f_M(x) \\ &= f_0(x) + \Delta_1(x) + \Delta_2(x) + \dots + \Delta_M(x) \\ &= f_0(x) + \sum_{m=1}^M \Delta_m(x)\end{aligned}$$

Po rozwinięciu rekurencyjnym:

$$\begin{aligned}F_0(x) &= f_0(x) \\ F_m(x) &= F_{m-1}(x) + \Delta_m(x)\end{aligned}$$

---

<sup>1</sup>W przypadku drzew wagi modeli  $w_m$  można ukryć w modelu poprzez przemnożenie wartości wszystkich liści przez  $w_m$

## L2 Boost

Proponujemy następujący algorytm boostingowy optymalizujący błąd kwadratowy tj.

$$\ell(y, \hat{y}) = (y - \hat{y})^2$$

**Algorithm:**  $l2boost(X, y, M, \eta)$  **returns** model  $F_M$

Let  $F_0(X) = \frac{1}{N} \sum_{i=1}^N y_i$ , mean of target  $y$  across all observations

**for**  $m = 1$  **to**  $M$  **do**

    Let  $\mathbf{r}_{m-1} = \mathbf{y} - F_{m-1}(X)$  be the residual direction vector

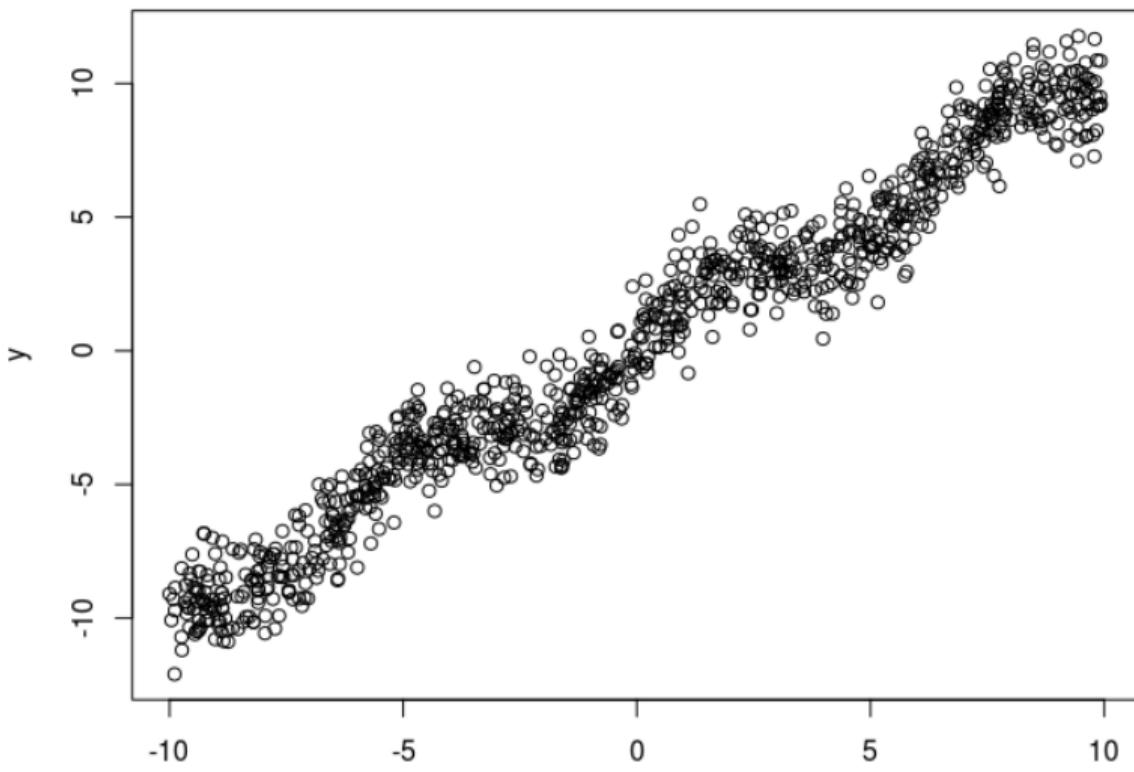
    Train regression tree  $\Delta_m$  on  $\mathbf{r}_{m-1}$ , minimizing squared error

$F_m(X) = F_{m-1}(X) + \eta \Delta_m(X)$

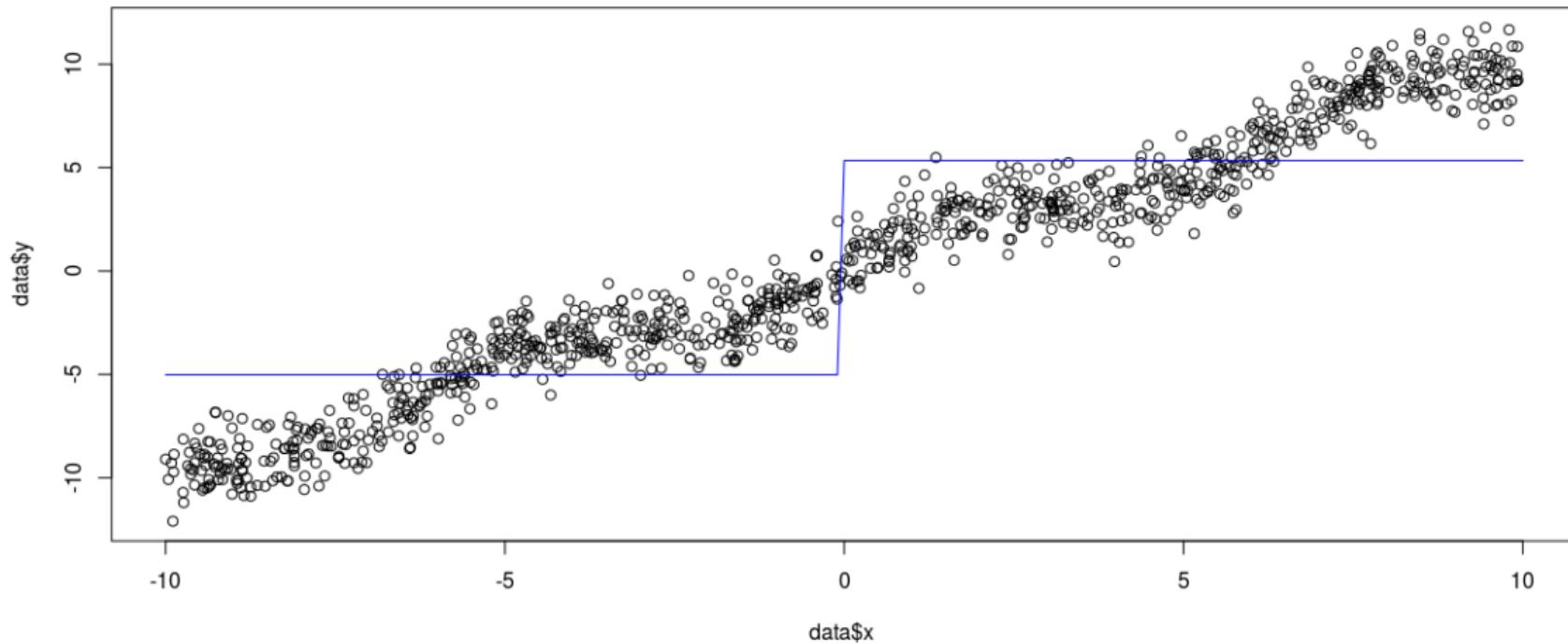
**end**

**return**  $F_M$

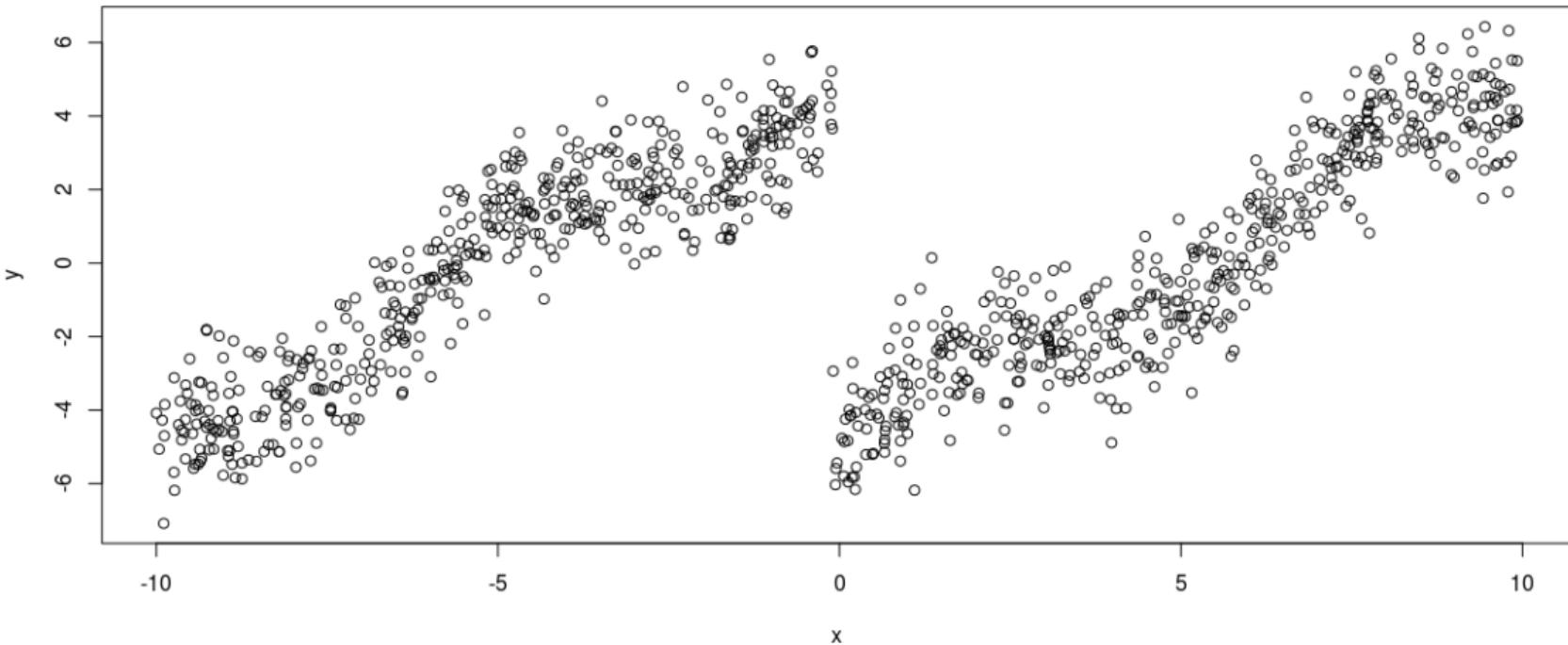
# Przykład: problem regresji



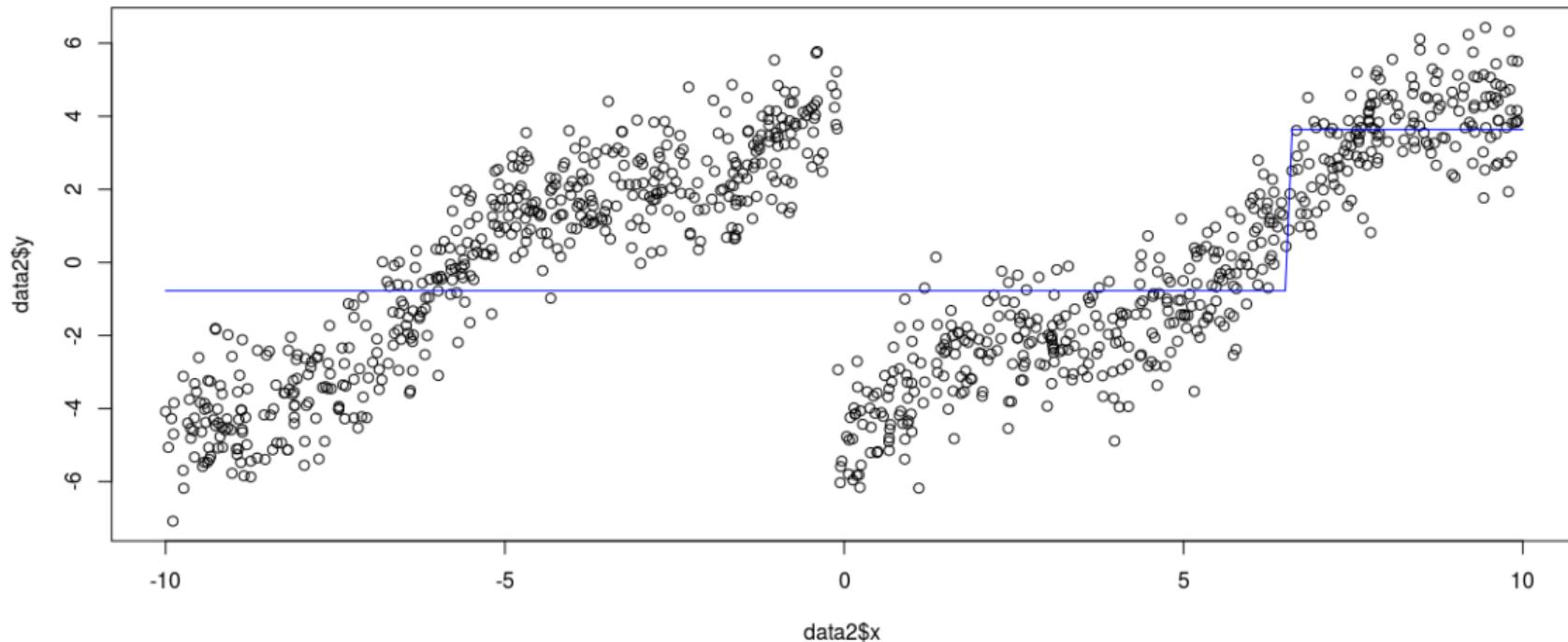
## L2Boost z decision stump - przykład



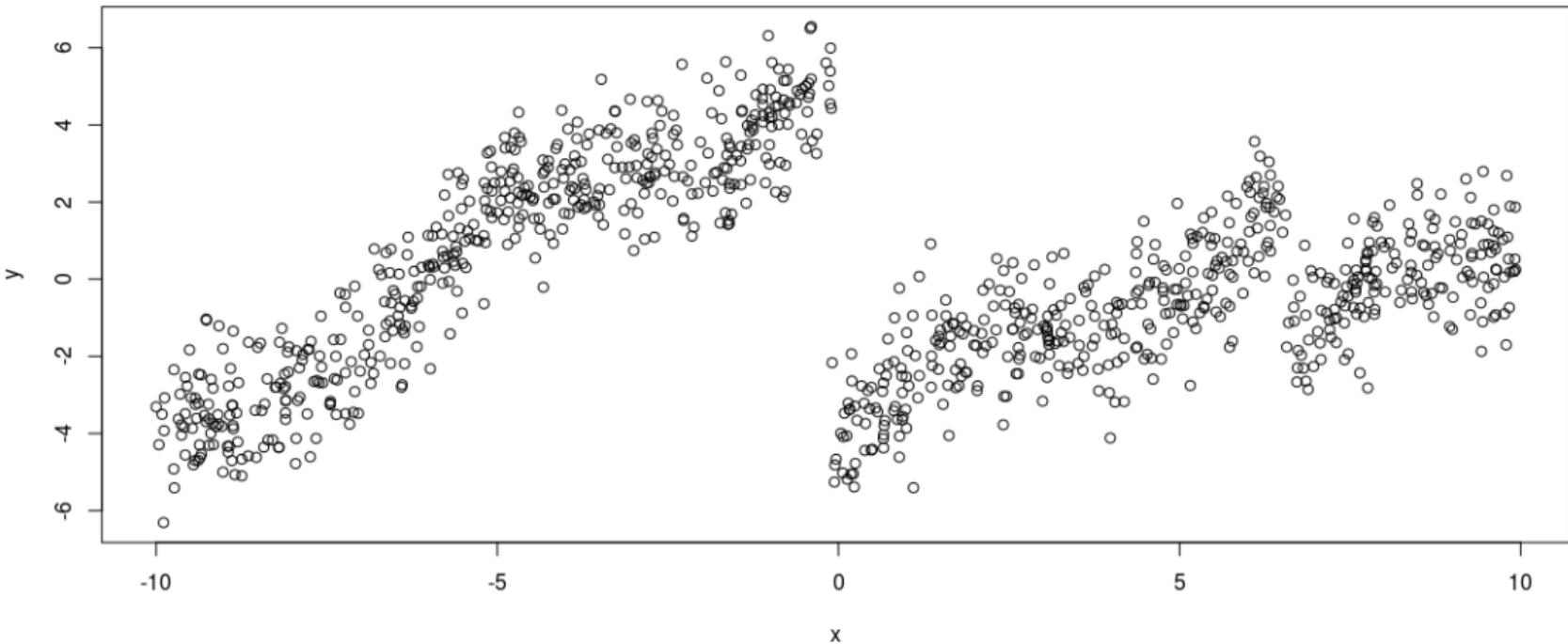
## L2Boost z decision stump - przykład



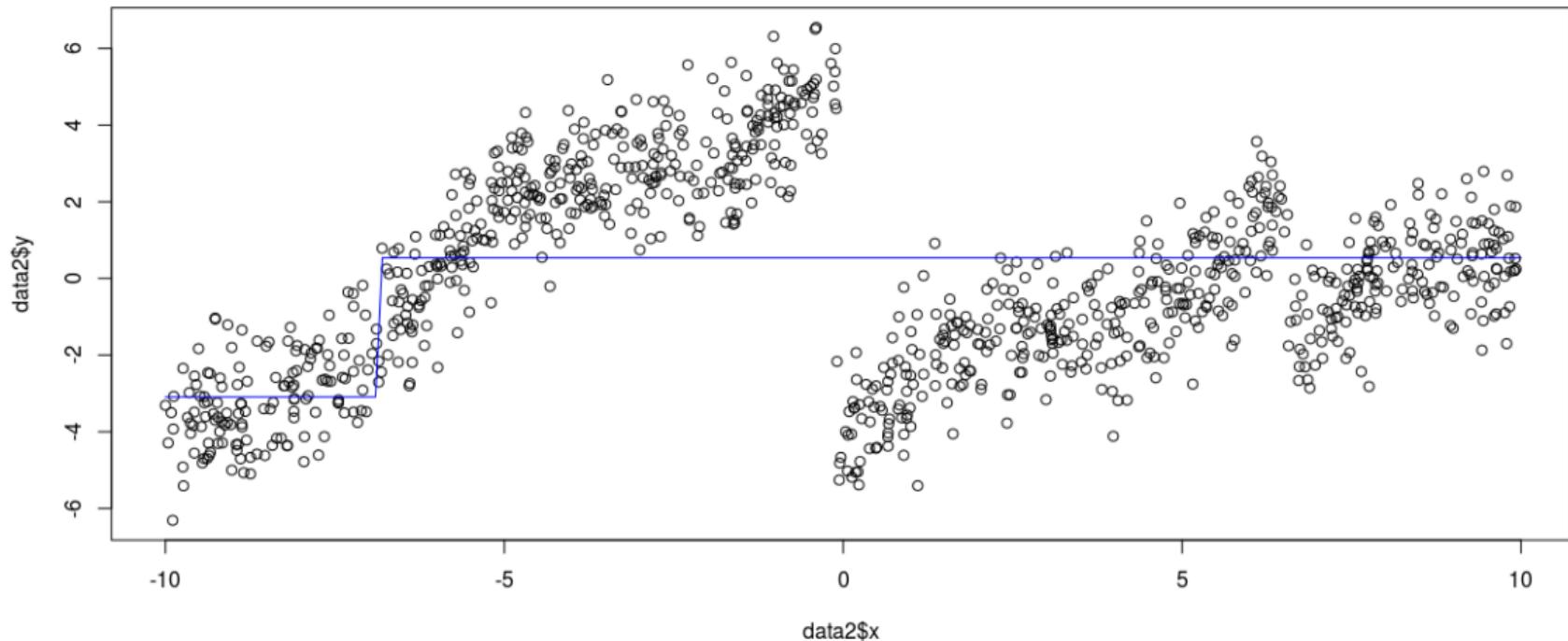
## L2Boost z decision stump - przykład



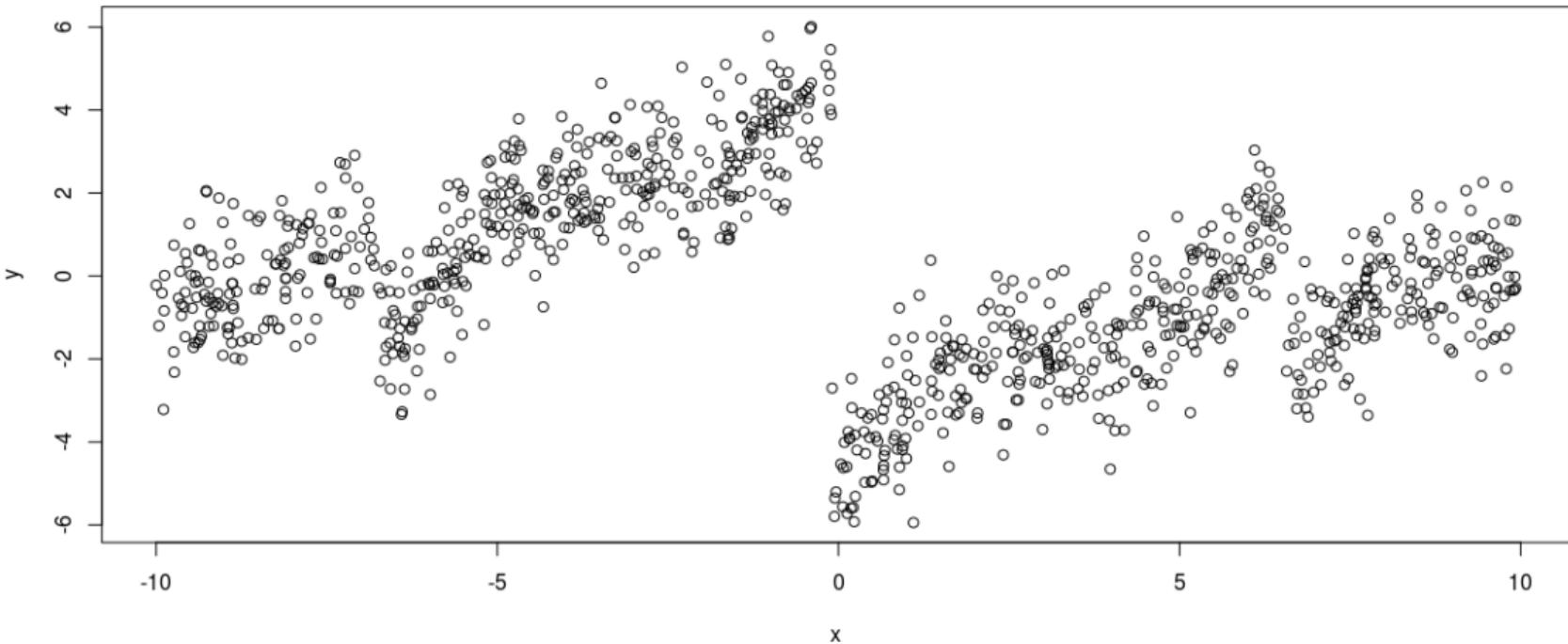
## L2Boost z decision stump - przykład



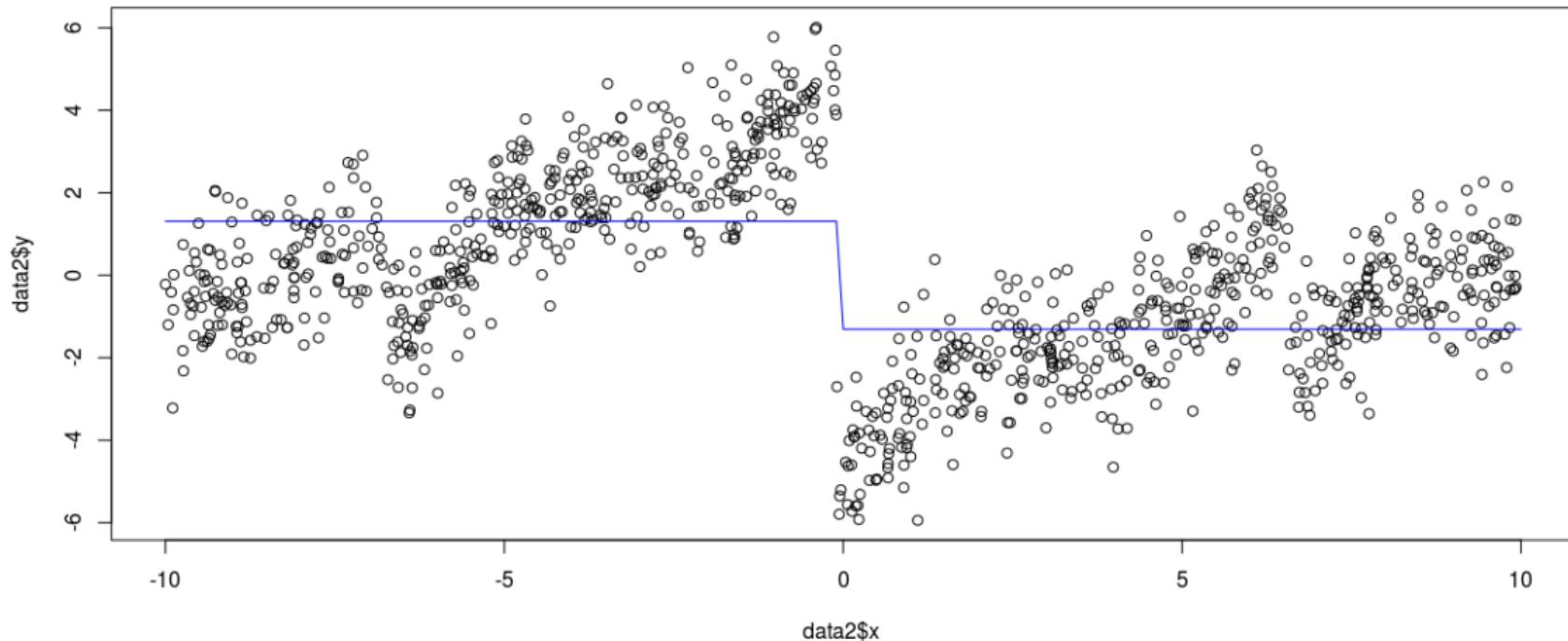
## L2Boost z decision stump - przykład



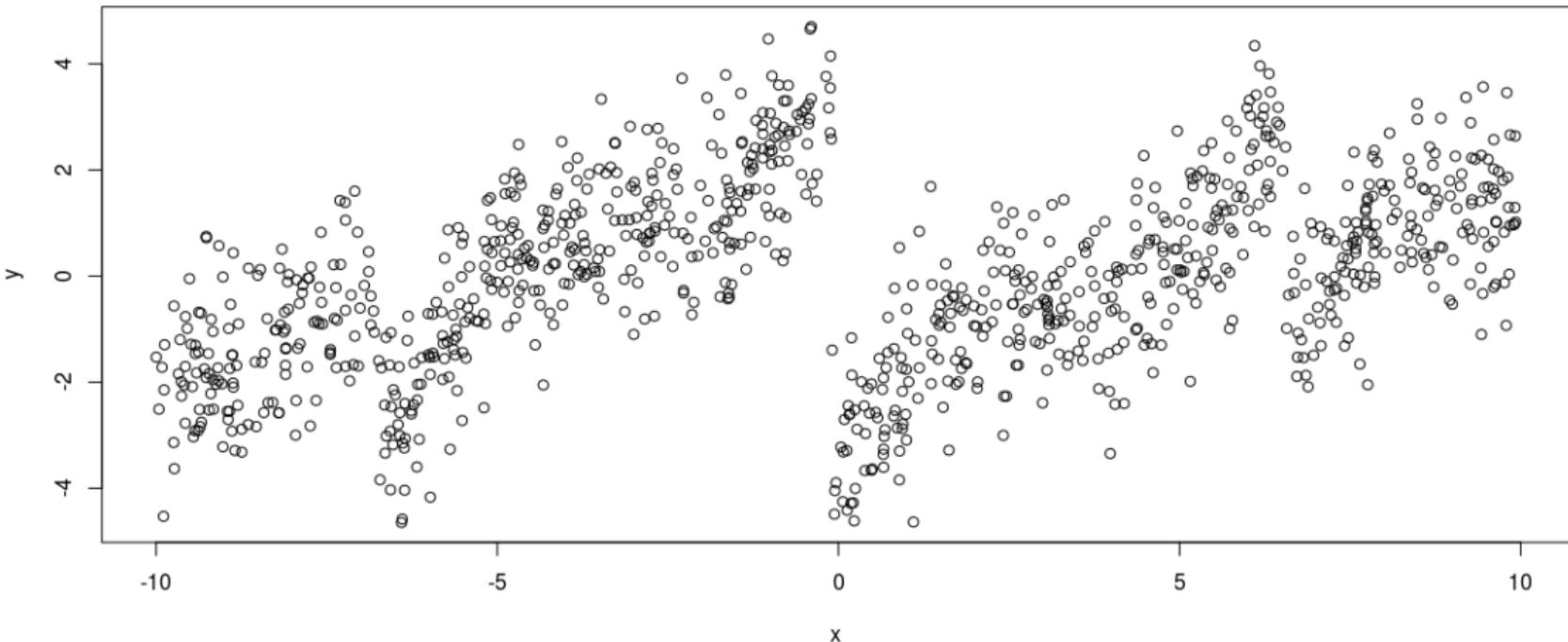
## L2Boost z decision stump - przykład



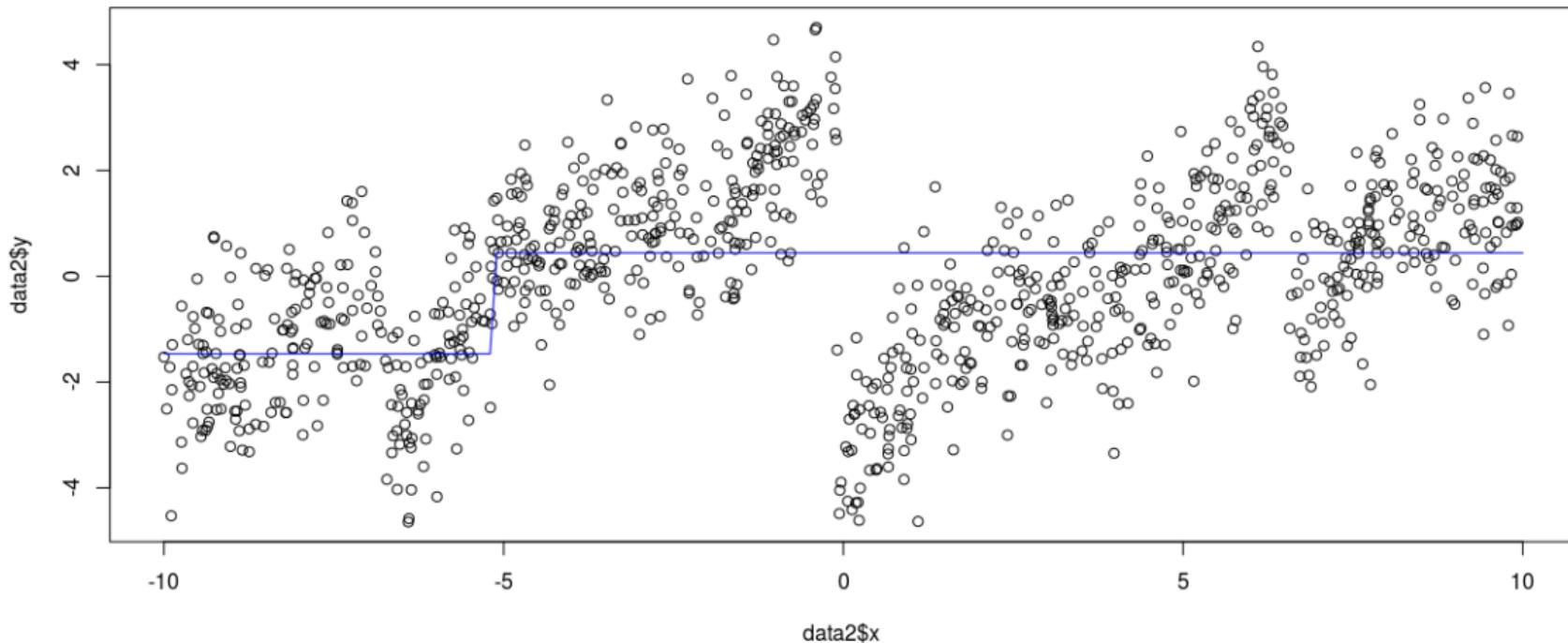
## L2Boost z decision stump - przykład



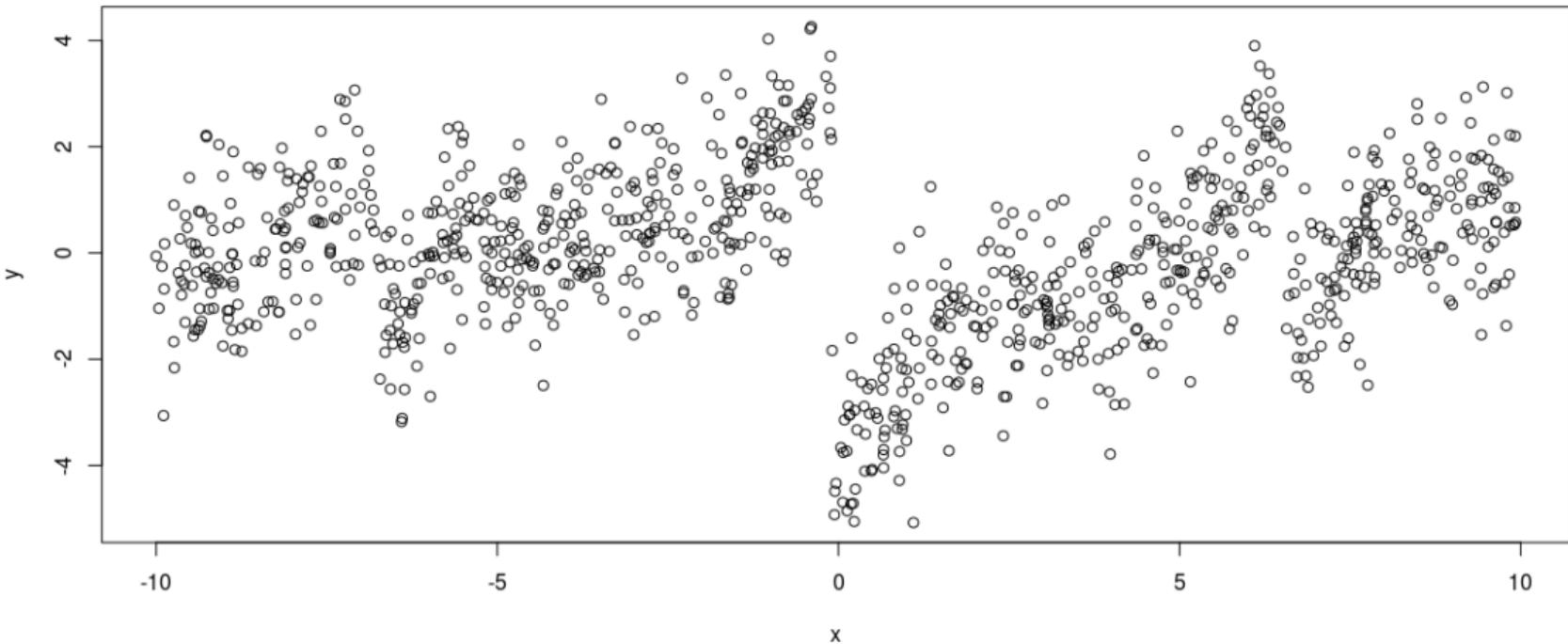
## L2Boost z decision stump - przykład



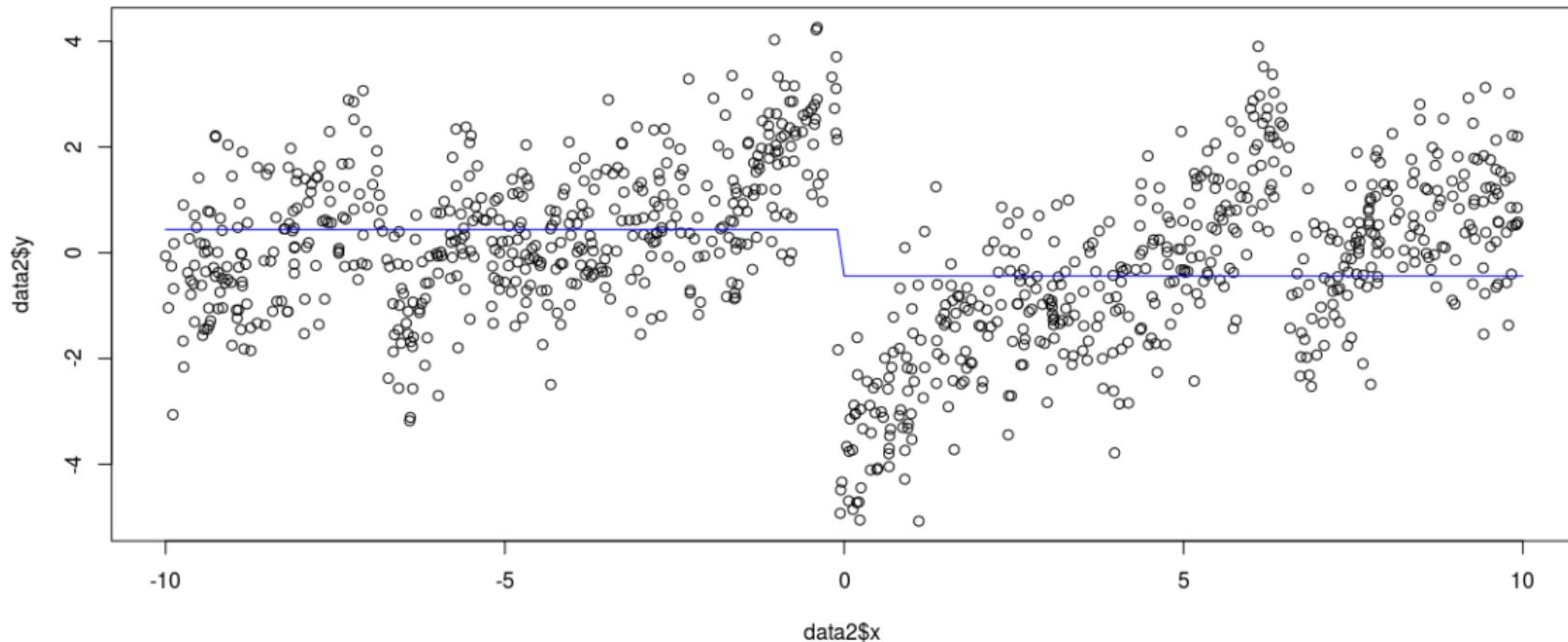
## L2Boost z decision stump - przykład



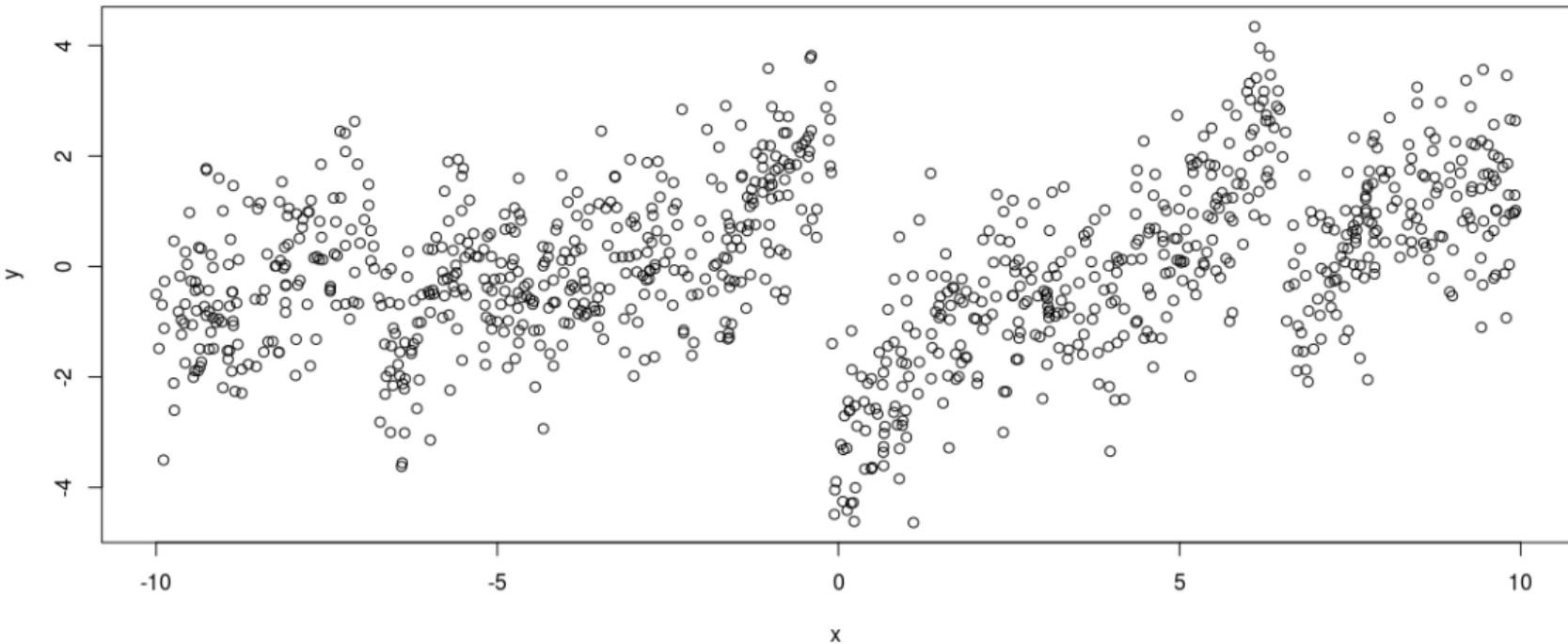
## L2Boost z decision stump - przykład



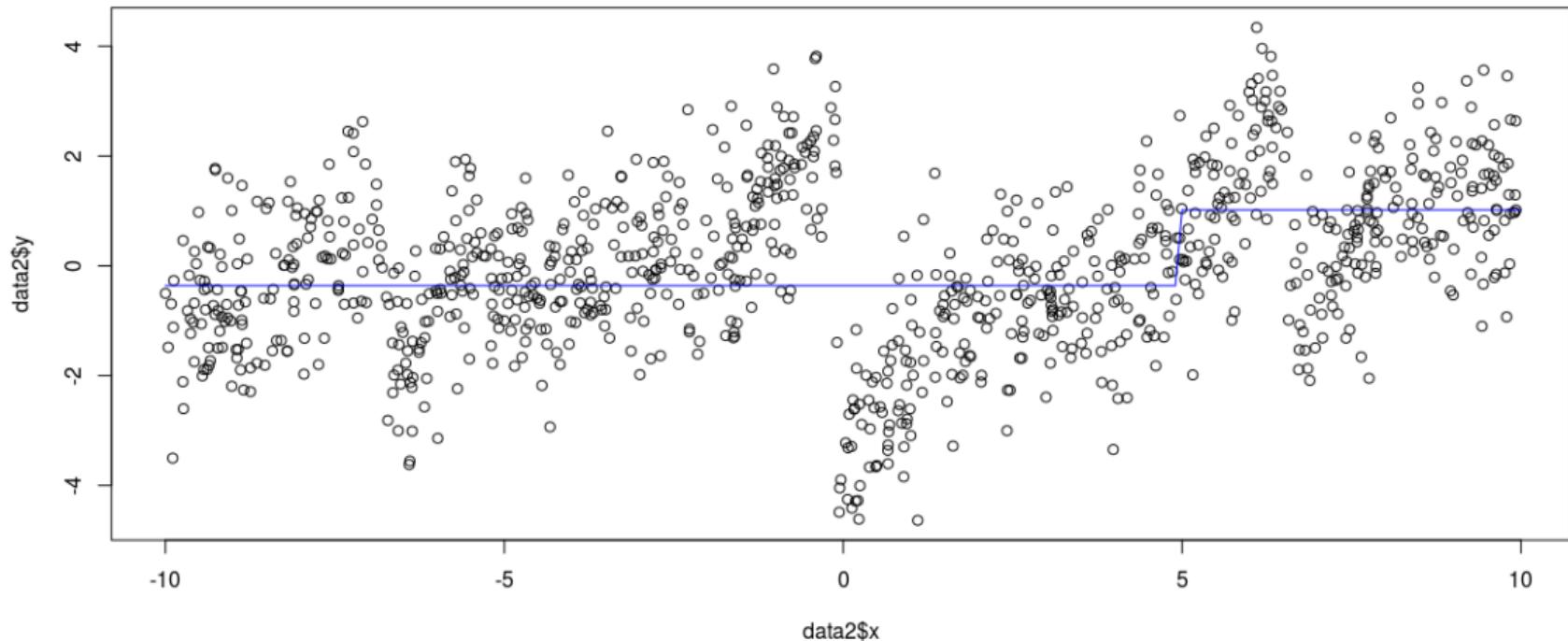
## L2Boost z decision stump - przykład



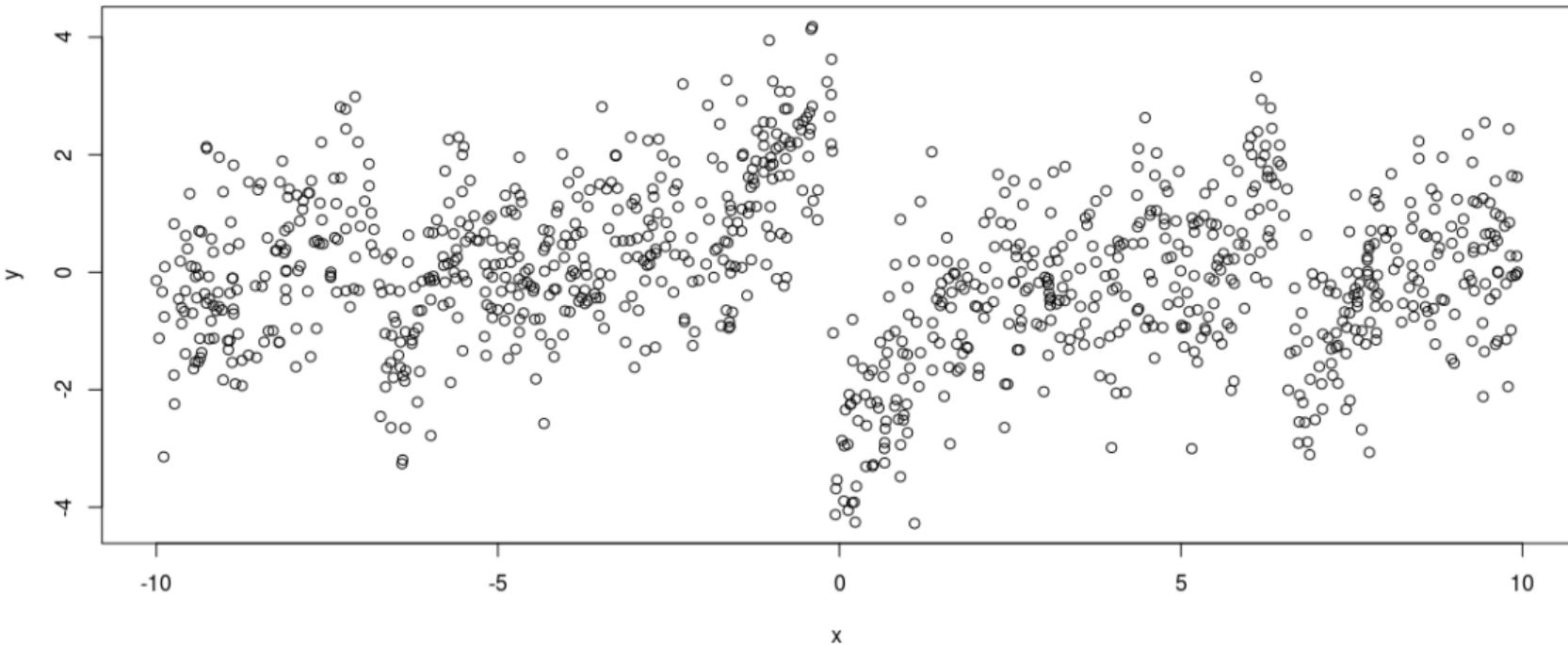
## L2Boost z decision stump - przykład



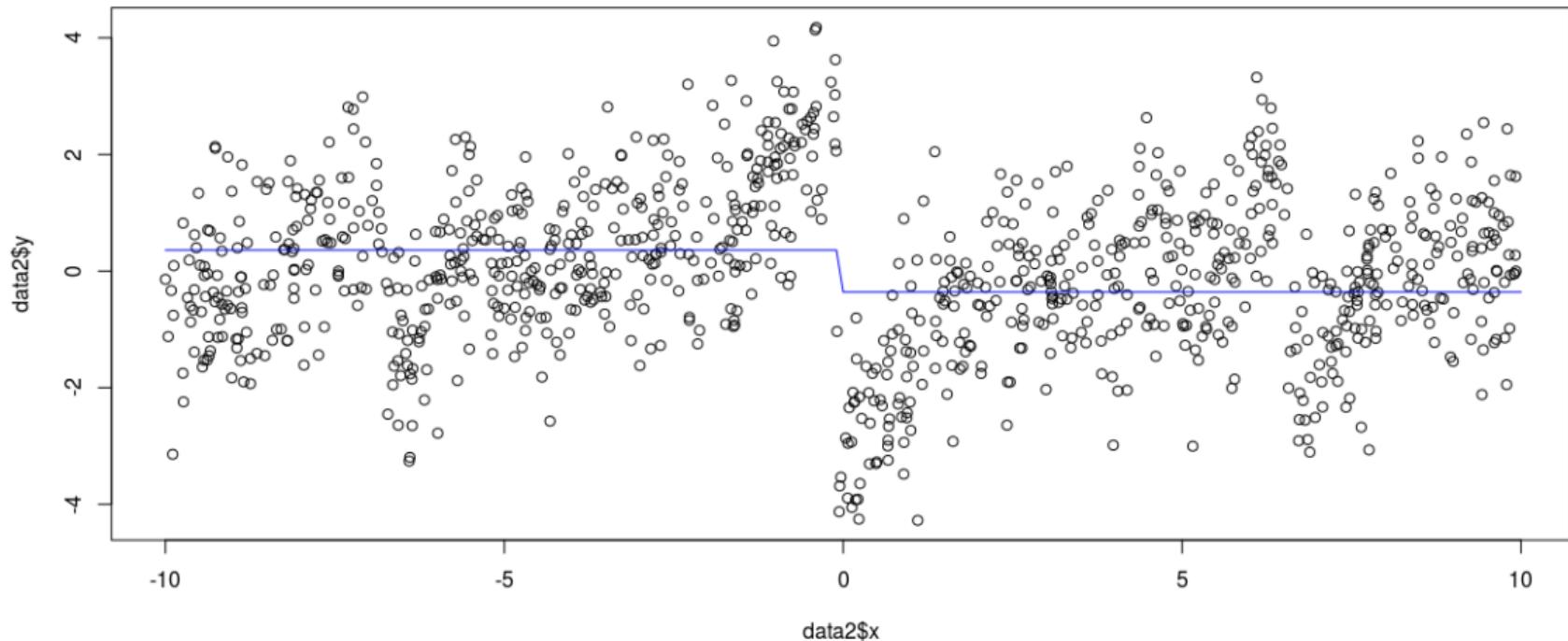
## L2Boost z decision stump - przykład



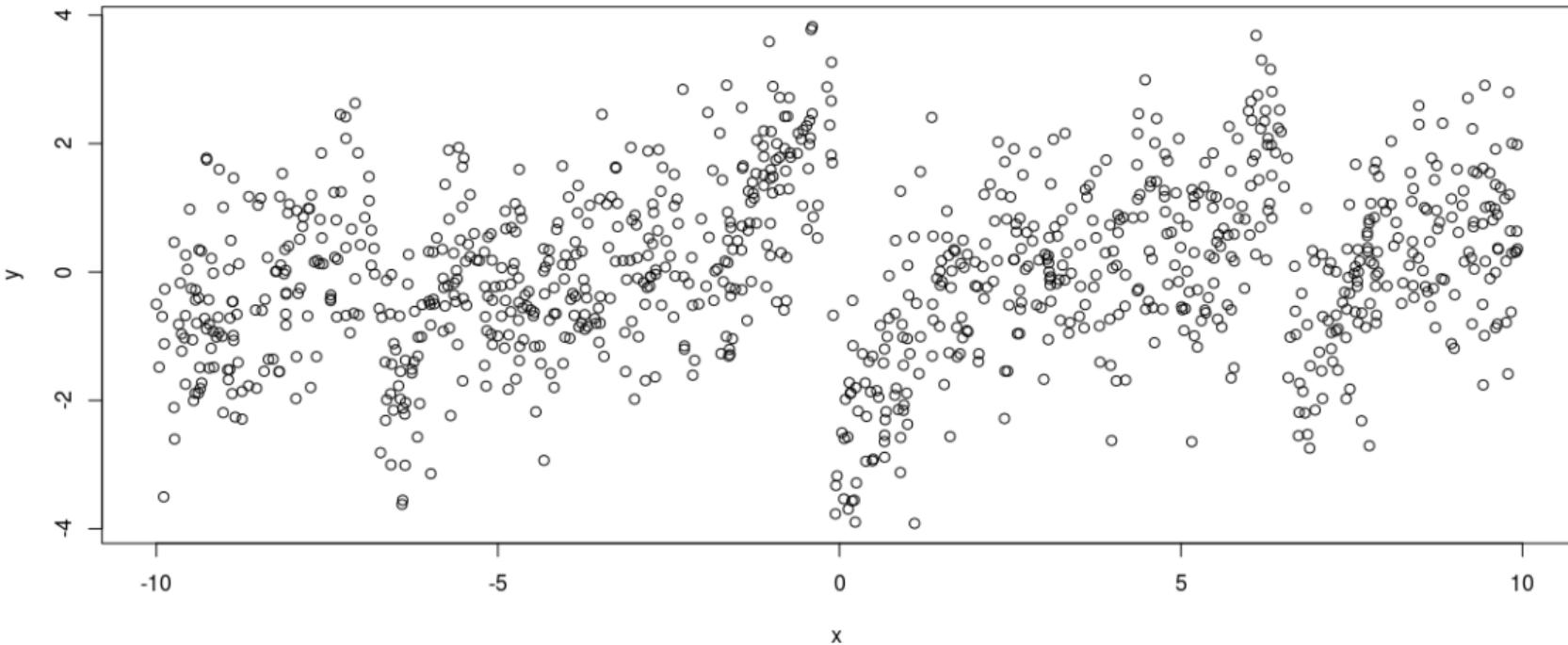
## L2Boost z decision stump - przykład



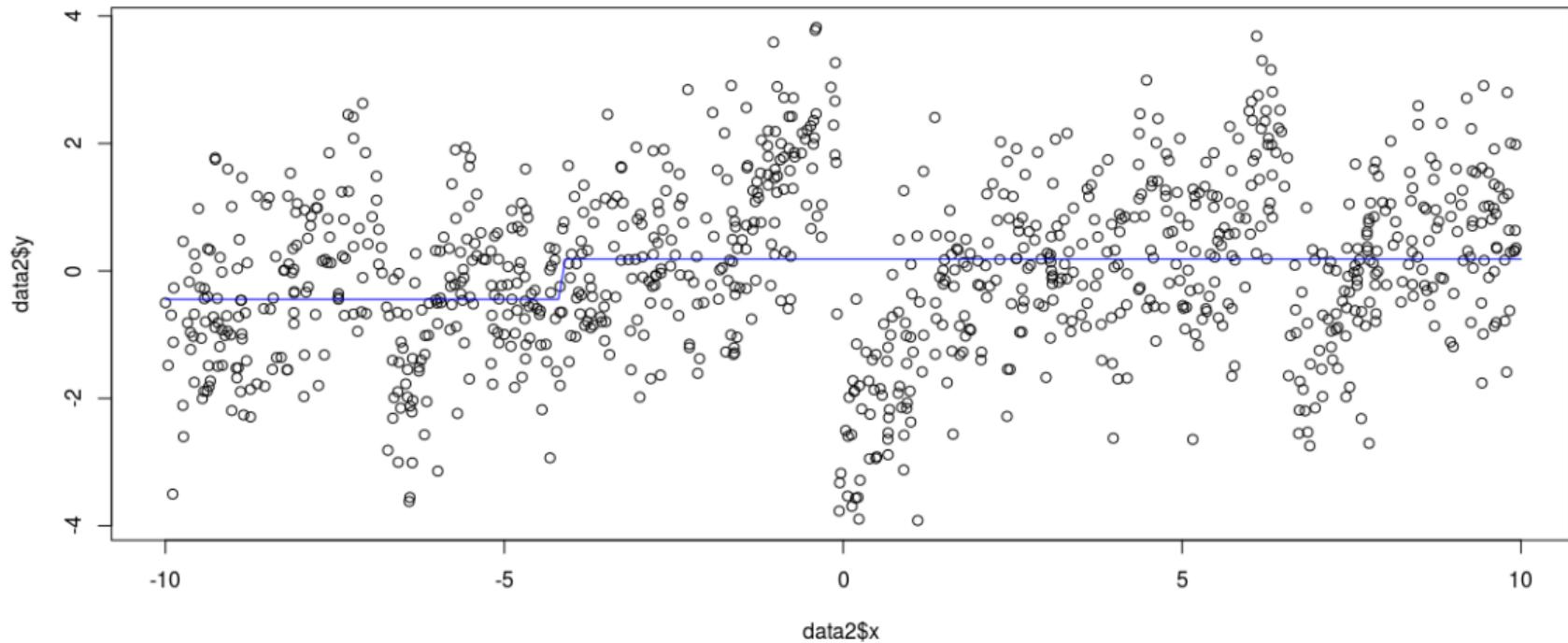
## L2Boost z decision stump - przykład



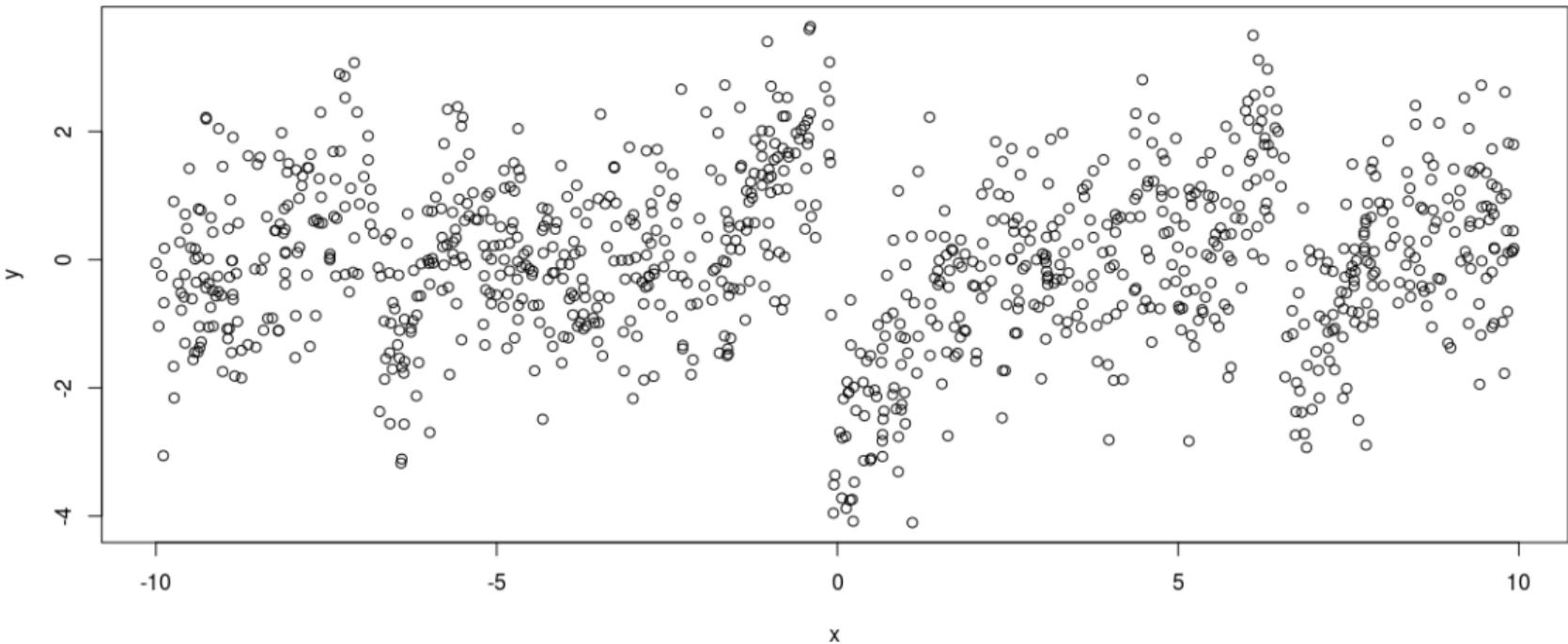
## L2Boost z decision stump - przykład



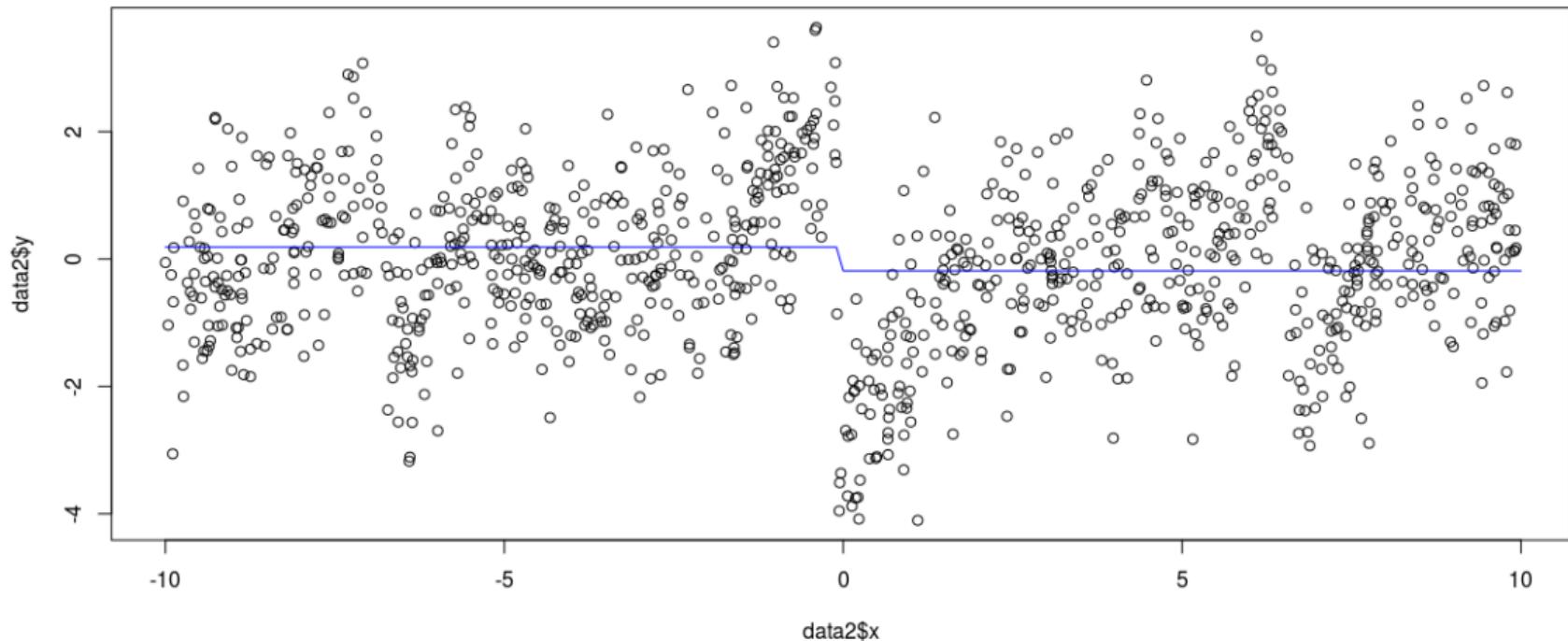
## L2Boost z decision stump - przykład



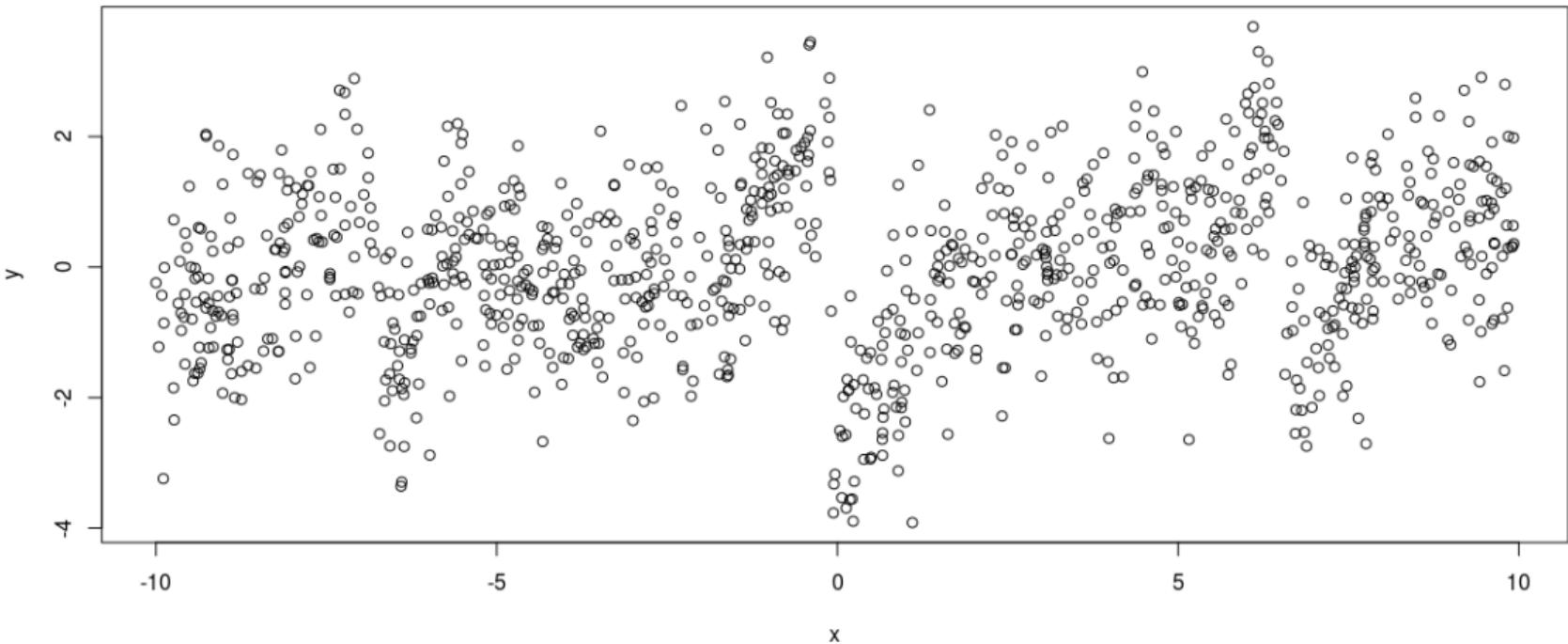
## L2Boost z decision stump - przykład



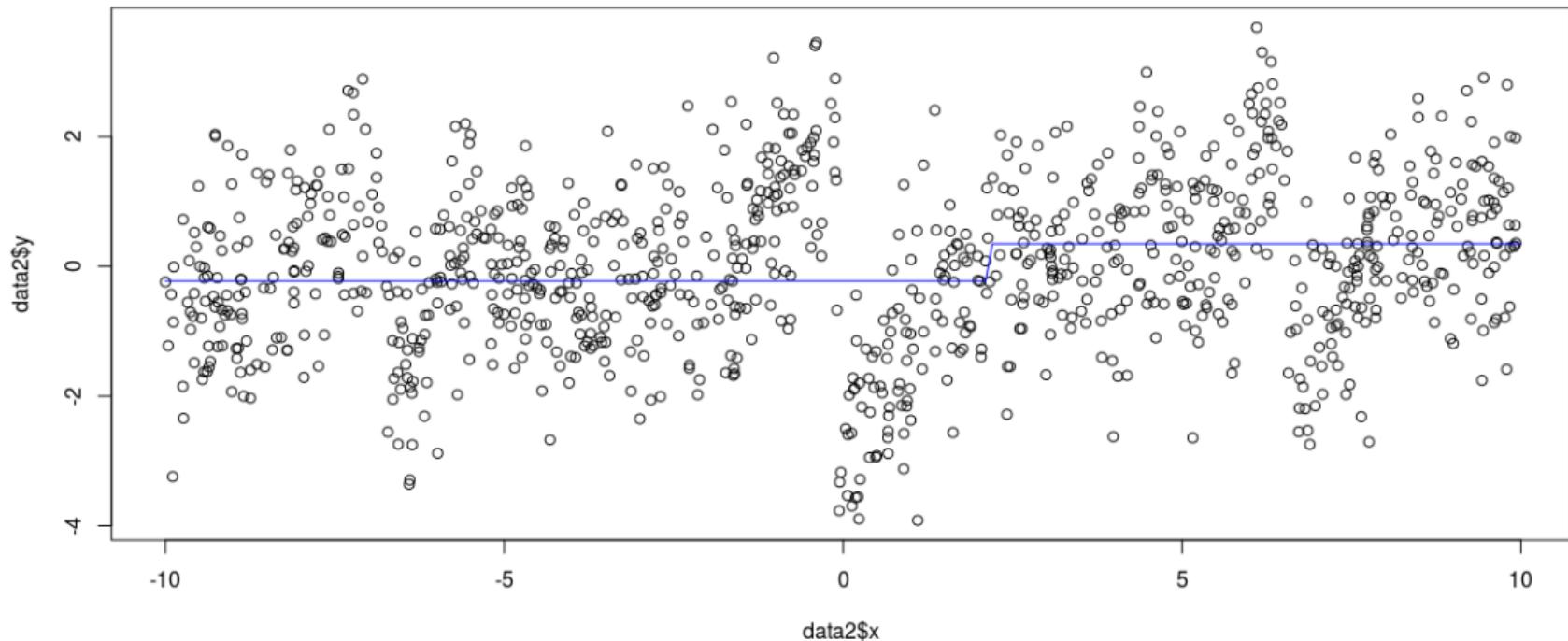
## L2Boost z decision stump - przykład



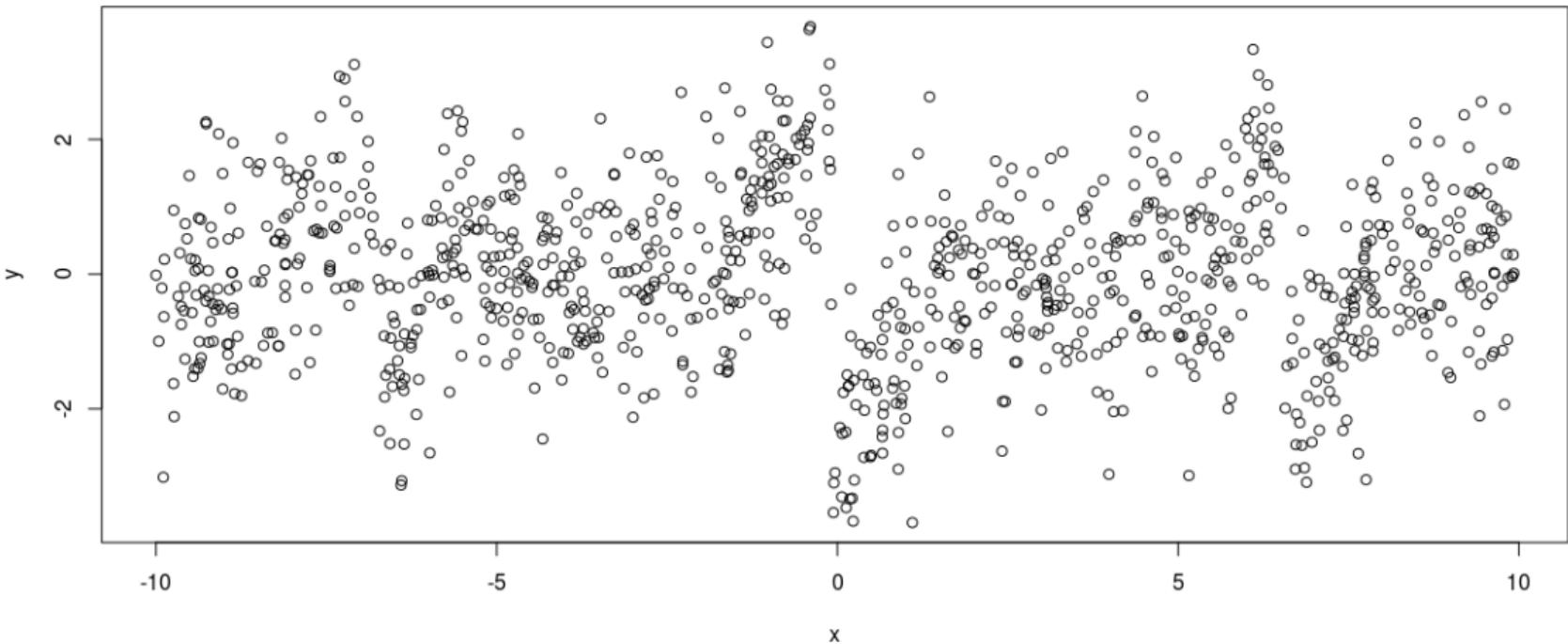
## L2Boost z decision stump - przykład



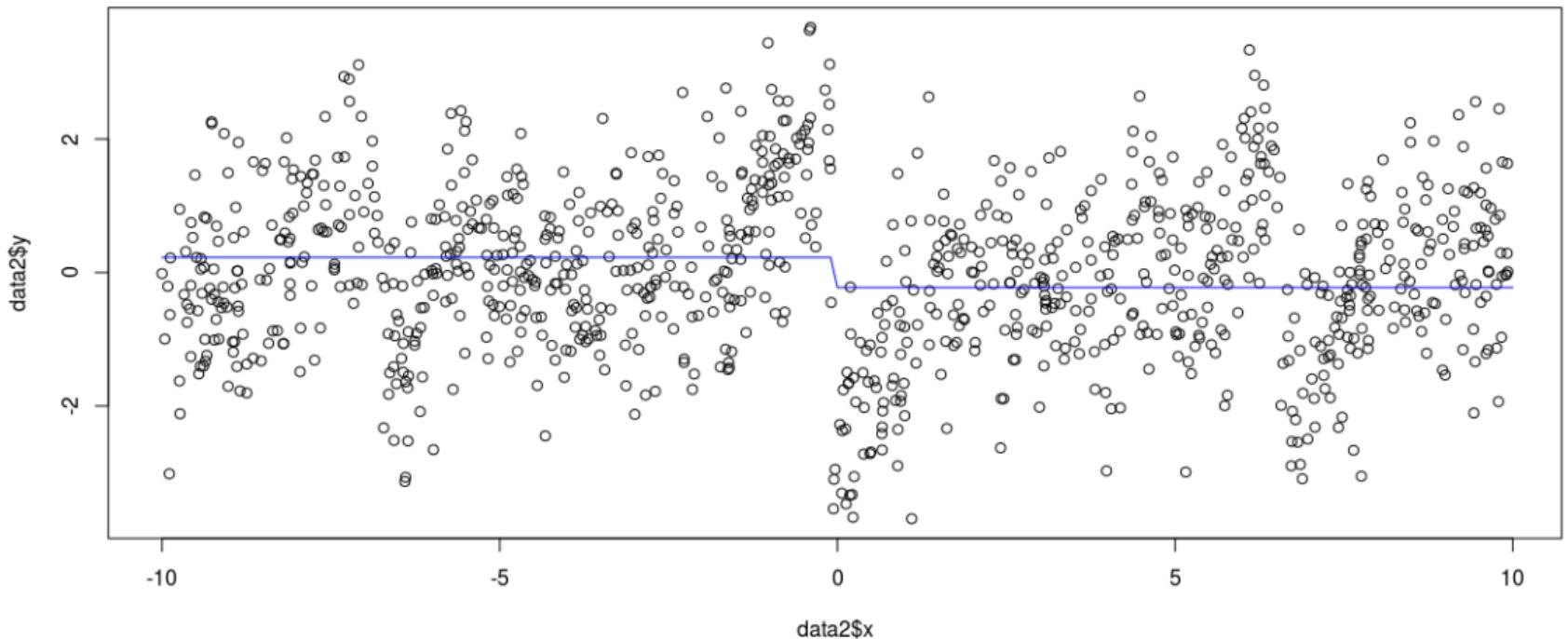
## L2Boost z decision stump - przykład



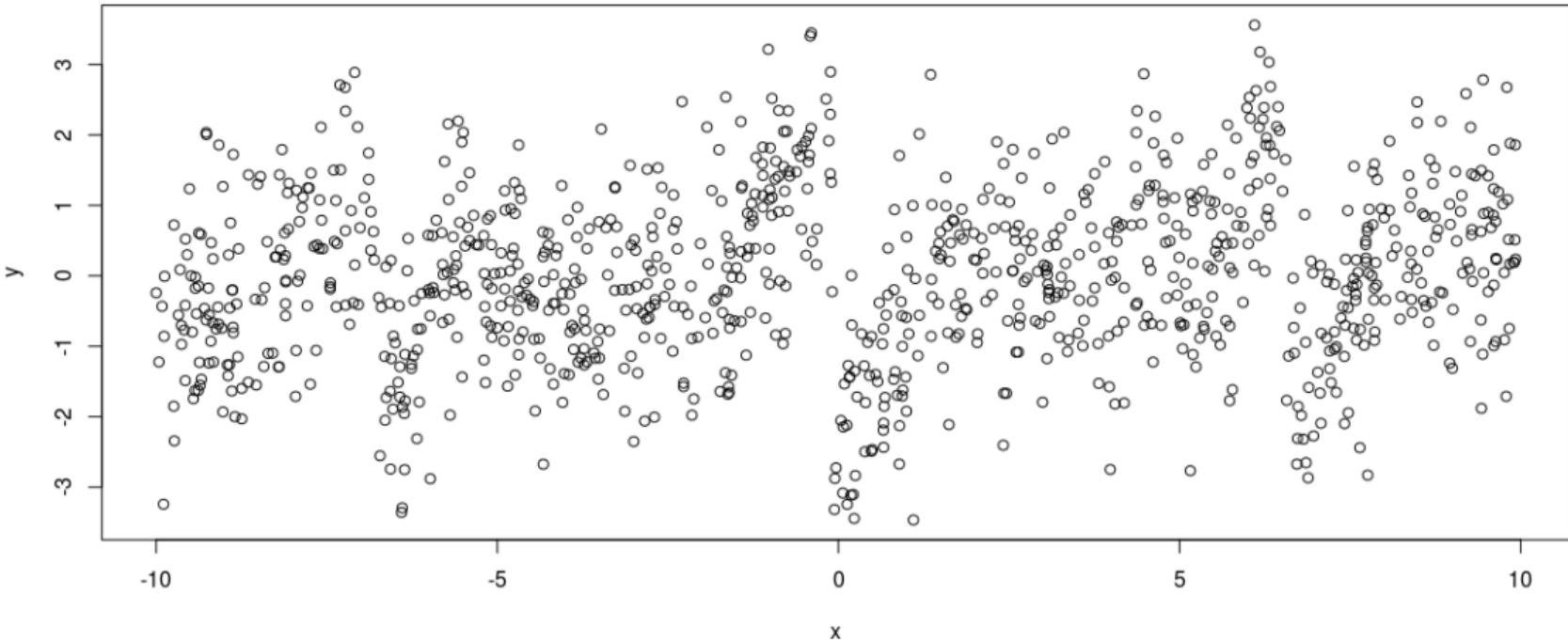
## L2Boost z decision stump - przykład



## L2Boost z decision stump - przykład

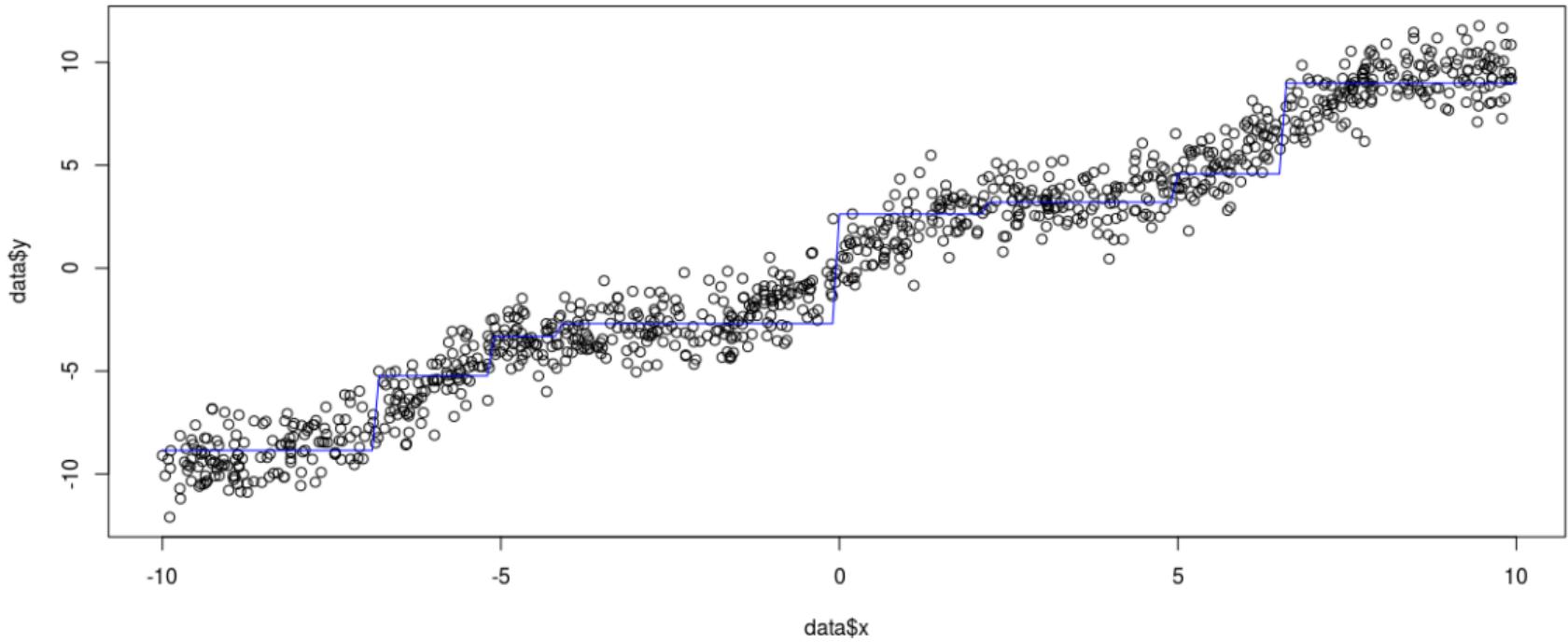


## L2Boost z decision stump - przykład

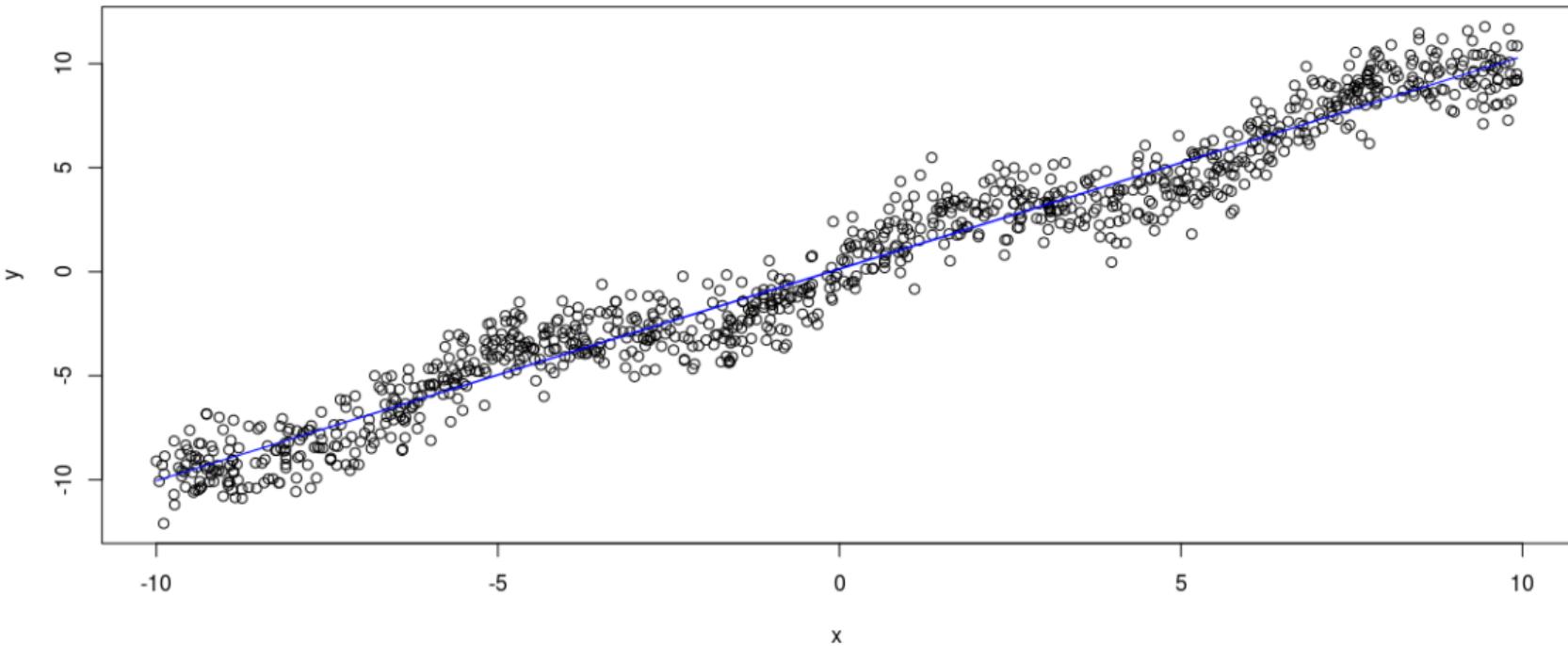


Potencjalnie dalej...

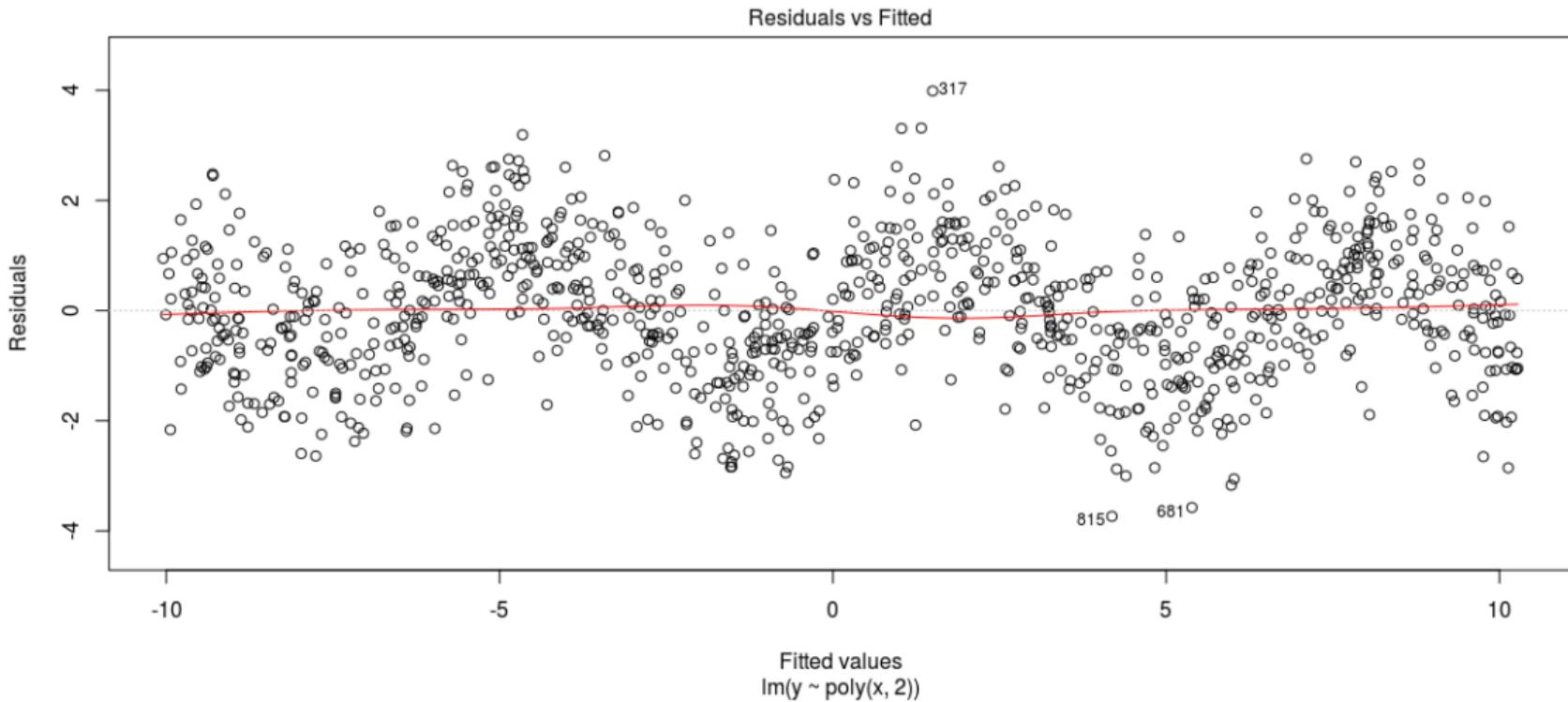
## L2Boost - ostateczny model



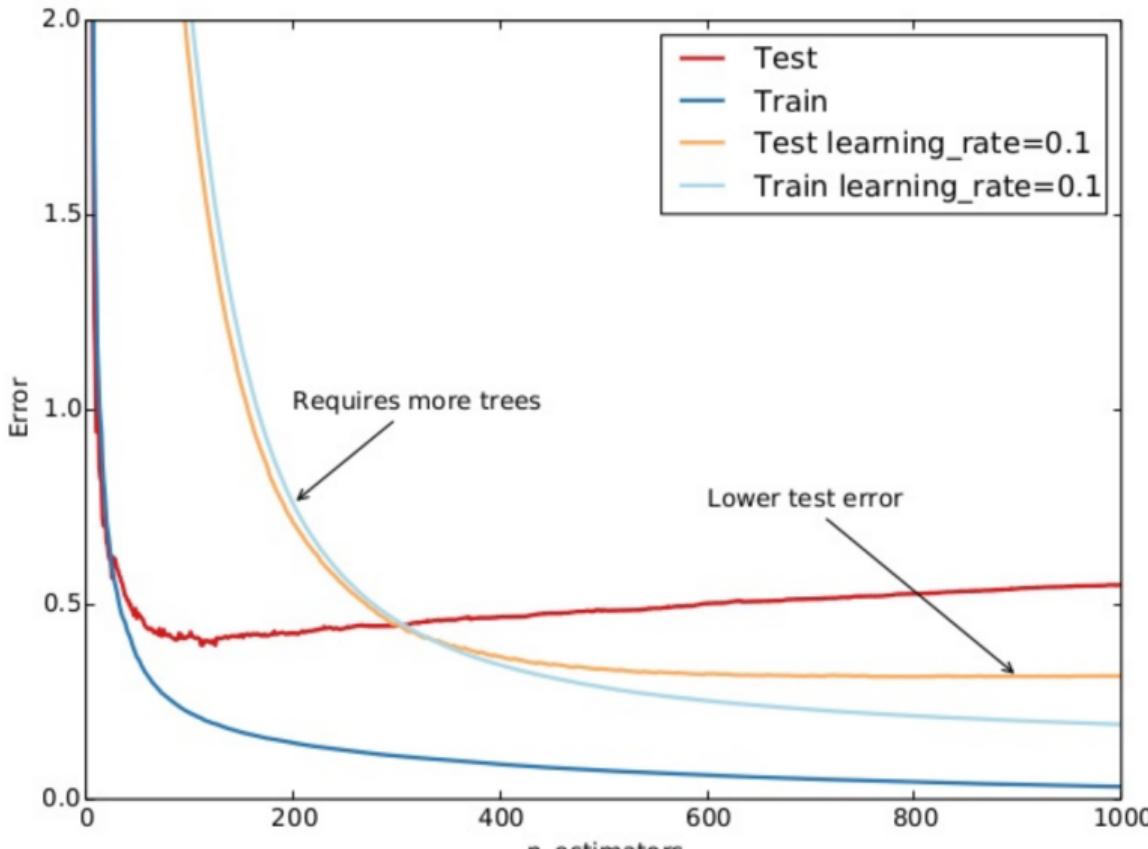
# A co z regresją kwadratową?



# A co z regresją kwadratową?



# Gradient Boosting Trees - wybór parametru $\eta$



# Gradient Boosting - kilka ważnych pytań

## Problem

*Jak tworzymy drzewa decyzyjne dla problemu regresji z MSE?*

## Problem

*Jak zaprojektować boosting dla MAE?*

## Problem

*Jak działa algorytm spadku wzdłuż gradientu (ang. gradient descent)?*

# Drzewa Wzmacniane Gradientowo – postać ogólna

---

**Algorithm 10.3** *Gradient Tree Boosting Algorithm.*

---

1. Initialize  $f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N L(y_i, \gamma)$ .

2. For  $m = 1$  to  $M$ :

(a) For  $i = 1, 2, \dots, N$  compute

$$r_{im} = - \left[ \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f=f_{m-1}}.$$

(b) Fit a regression tree to the targets  $r_{im}$  giving terminal regions

$$R_{jm}, \quad j = 1, 2, \dots, J_m.$$

(c) For  $j = 1, 2, \dots, J_m$  compute

$$\gamma_{jm} = \arg \min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma).$$

(d) Update  $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$ .

3. Output  $\hat{f}(x) = f_M(x)$ .

---

# Drzewa Wzmacniane Gradientowo dla MAE

Proponujemy następujący algorytm boostingowy optymalizujący błąd bezwzględny tj.

$$\ell(y, \hat{y}) = |y - \hat{y}|$$

**Algorithm:**  $l1boost(X, y, M, \eta)$  returns model  $F_M$

Let  $F_0(X) = median(y)$

**for**  $m = 1$  **to**  $M$  **do**

    Let  $\mathbf{r}_{m-1} = \mathbf{y} - F_{m-1}(X)$  be the direction vector

    Let  $\mathbf{sign}_{m-1} = sign(\mathbf{r}_{m-1})$  be the sign vector

    Train regression tree  $\Delta_m$  on  $\mathbf{sign}_{m-1}$ , minimizing squared error

**foreach** leaf  $l \in \Delta_m$  **do**

        Alter  $l$  to predict median (not mean) of  $y_i - F_{m-1}(x_i)$  for obs.  $i$  in  $l$

**end**

$F_m(X) = F_{m-1}(X) + \eta \Delta_m(X)$

**end**

**return**  $F_M$

# Przypomnienie: regresja logistyczna

## Problem

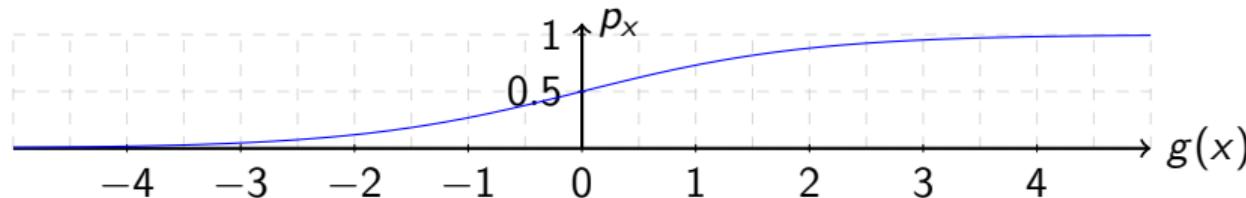
Jak zaprojektować Drzewa Wzmacniane Gradientowo dla problemu klasyfikacji?

Przypomnijmy, że w regresji logistycznej:

$$\text{logit}(p_x) = \ln \frac{p_x}{1 - p_x} = w^T x + b$$

Wyrażając prawdopodobieństwo  $p_x$  w zależności od wyniku wyrażenia liniowego  $g(x) = w^T x + b$  otrzymujemy:

$$p_x = \frac{1}{1 + e^{-g(x)}}$$



# Drzewa Wzmacniane Gradientowo do klasyfikacji

$$\text{logit}(p_x) = \ln \frac{p_x}{1 - p_x} = f_0(x) + \sum_{i=1}^M \Delta_i(x)$$

$$\ell(\hat{y} - y) = y \log \hat{y} + (1 - y) \log(1 - \hat{y})$$

## Problem

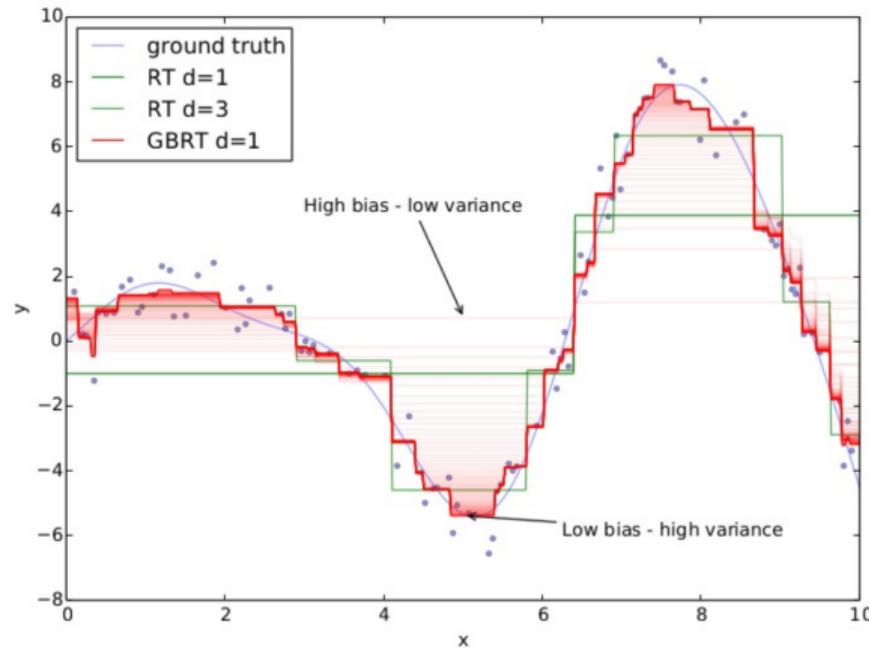
W jaki sposób obliczyć pseudo-rezydua?

## Problem

W jaki sposób obliczyć wartość liścia w GBT?

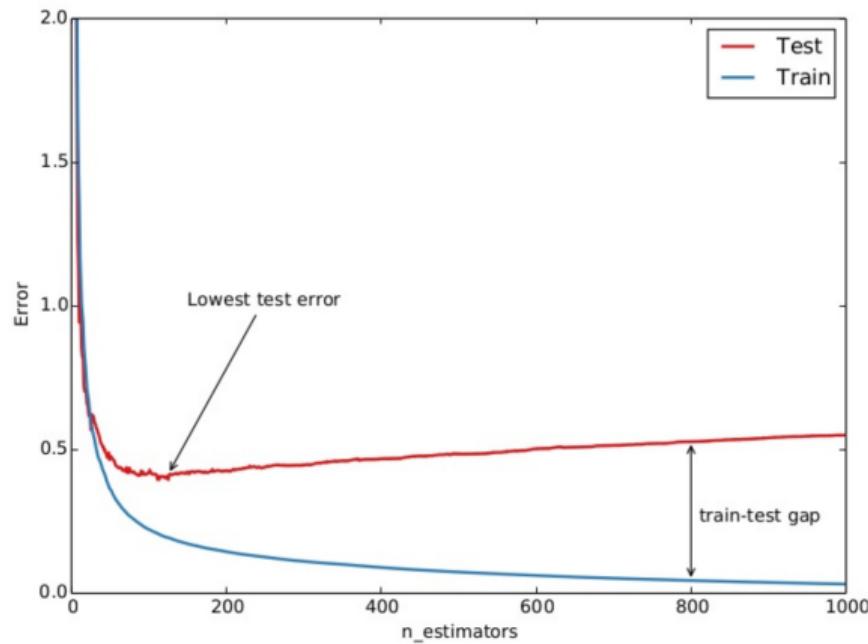
# Drzewa Wzmacniane Gradientowo mogą się przeuczyć...

```
from sklearn.ensemble import GradientBoostingRegressor  
est = GradientBoostingRegressor(n_estimators=2000, max_depth=1).fit(X, y)  
for pred in est.staged_predict(X):  
    plt.plot(X[:, 0], pred, color='r', alpha=0.1)
```

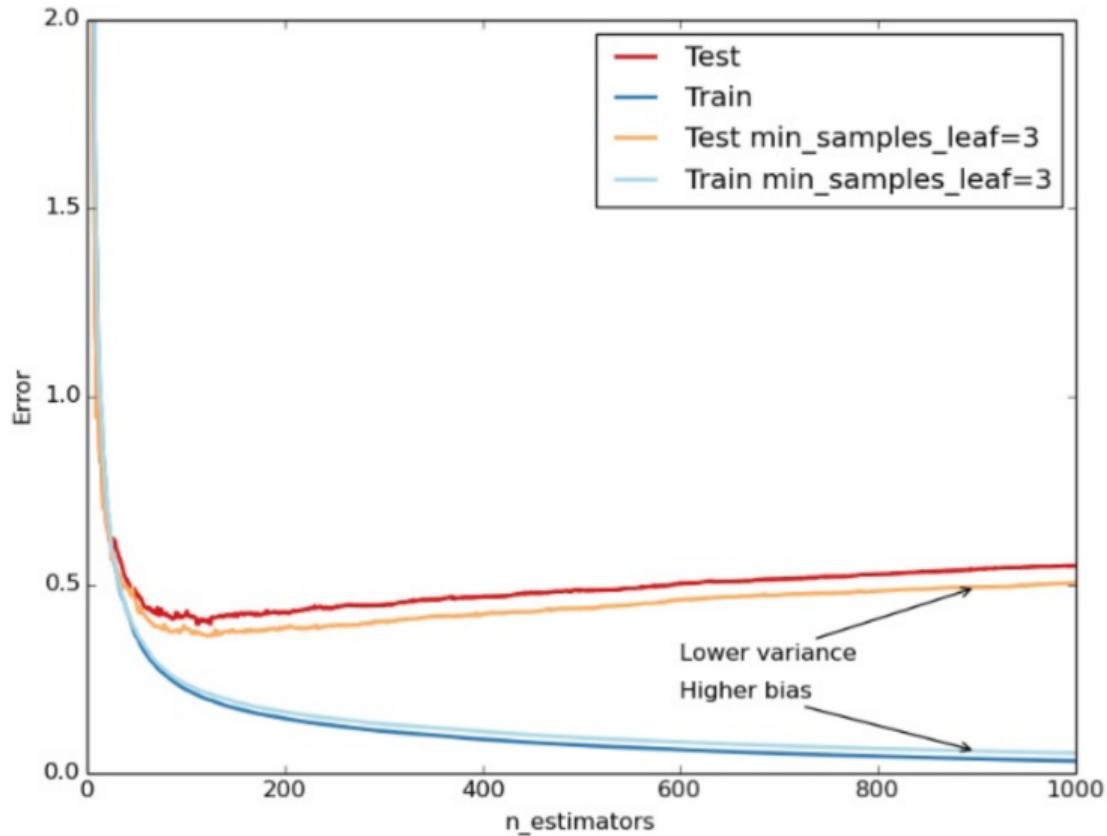


# Drzewa Wzmacniane Gradientowo: zasada dobierania $M$

```
test_score = np.empty(len(est.estimators_))
for i, pred in enumerate(est.staged_predict(X_test)):
    test_score[i] = est.loss_(y_test, pred)
plt.plot(np.arange(n_estimators) + 1, test_score, label='Test')
plt.plot(np.arange(n_estimators) + 1, est.train_score_, label='Train')
```



# Drzewa Wzmacniane Gradientowo: strojenie parametrów pruningu drzewa



# Drzewa Wzmacniane Gradientowo - jak wybrać rozmiar drzewa?

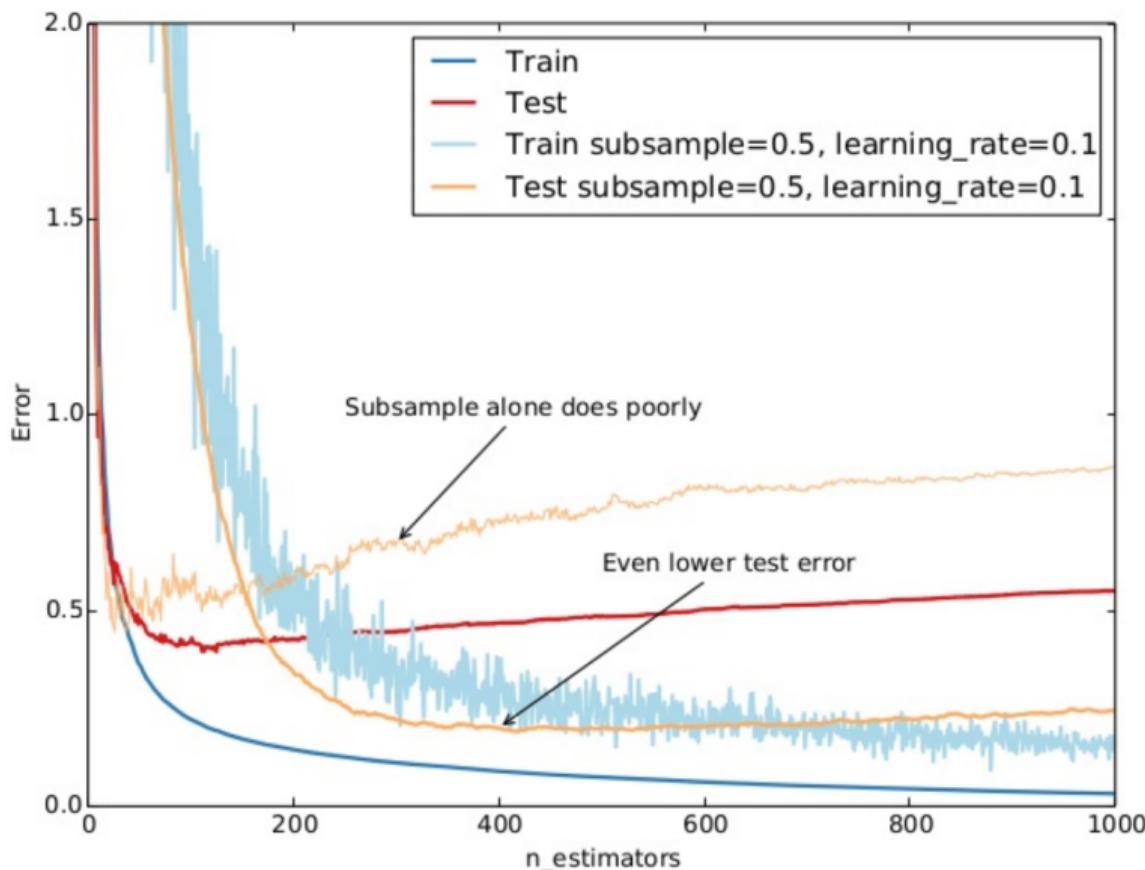
Przykładowy model oparty o decision stump:

$$\begin{aligned}F_M(x) &= \sum_{m=1}^M w_m f_m(x; \theta_m) \\&= w_1 f(x_5) + w_2 f(x_1) + w_3 f(x_3) + w_4 f(x_5) + \dots \\&= w_2 f(x_1) + \dots + w_3 f(x_3) + \dots + \underbrace{w_1 f(x_5) + w_4 f(x_5) + \dots}_{\eta_5(x_5)} \\&= \sum_{j=1}^d \eta_j(x_j)\end{aligned}$$

# Drzewa Wzmacniane Gradientowo - jak wybrać rozmiar drzewa?

$$\begin{aligned}F_M(x) &= \sum_{m=1}^M w_m f_m(x; \theta_m) \\&= \sum_{j=1}^d \eta_j(x_j) + \sum_{j=1}^d \sum_{k=1}^d \eta_{jk}(x_j, x_k) + \sum_{j=1}^d \sum_{k=1}^d \sum_{l=1}^d \eta_{jkl}(x_j, x_k, x_l) + \dots\end{aligned}$$

# Stochastyczne Drzewa Wzmacniane Gradientowo



# XGBoost - Konstrukcja drzewa (regresja)

- Kryterium podziału

$$Structure\_Score = \frac{(\sum r_i)^2}{n}$$

$$Gain = Structure\_Score_L + Structure\_Score_R - Structure\_Score_P$$

- Wartość w liściu

$$\theta = \frac{\sum r_i}{n}$$

- Pre-prunning:

$$n < \tau$$

- Post-prunning:

$$Gain < \gamma$$

# XGBoost - Konstrukcja drzewa (klasyfikacja)

- Kryterium podziału

$$\text{Structure\_Score} = \frac{(\sum r_i)^2}{\sum p_i(1 - p_i)}$$

$$Gain = \text{Structure\_Score}_L + \text{Structure\_Score}_R - \text{Structure\_Score}_P$$

- Wartość w liściu

$$\theta = \frac{\sum r_i}{\sum p_i(1 - p_i)}$$

- Pre-prunning:

$$\sum p_i(1 - p_i) < \tau$$

- Post-prunning:

$$Gain < \gamma$$

# XGBoost - Konstrukcja drzewa z regularyzacją

$$\sum_{i=1}^N L(y_i, \hat{y}_i) + \frac{1}{2}\lambda \sum \theta + \gamma T$$

- Kryterium podziału

$$Structure\_Score = \frac{(\sum r_i)^2}{\lambda + \sum p_i(1 - p_i)}$$

- Wartość w liściu

$$\theta = \frac{\sum r_i}{\lambda + \sum p_i(1 - p_i)}$$

# Gradient Boosting Trees: Strategia wyboru parametrów

- ➊ Wybierz stosunkowo wysokie  $\eta$  i określ optymalne  $M$ 
  - Czasem lepiej: ustaw  $M$  na odpowiednią wartość (czas!) i wybierz do tego  $\eta$  razem z parametrami
- ➋ Dostrój parametry drzewa
- ➌ Dostrój regularyzację
- ➍ Ustaw  $M$  tak duże jak to możliwe i dostrój  $\eta$

# Wiele różnych implementacji

- sklearn
- XGBoost
- H2O
- LightGBM
- Catboost
- ...

Dziękuję za uwagę!



**Fundusze  
Europejskie**  
Polska Cyfrowa



**Rzeczpospolita  
Polska**

**Unia Europejska**  
Europejski Fundusz  
Rozwoju Regionalnego

