CatBoost

Agata Kopyt, Zuzanna Kotlińska, Szymon Matuszewski, Patryk Słowakiewicz

Contents

1. Introduction & Idea

Introduction

- **What is CatBoost?**

  It's a high-performance open-source library for gradient boosting on decision trees developed by Yandex researchers and engineers in 2017.

Easy-to-use **Practical** **Accurate**

https://youtu.be/s8Q_orF4tcI

Usage

- Recommendation systems
- Personal assistant
- Self-driving cars
- Weather prediction

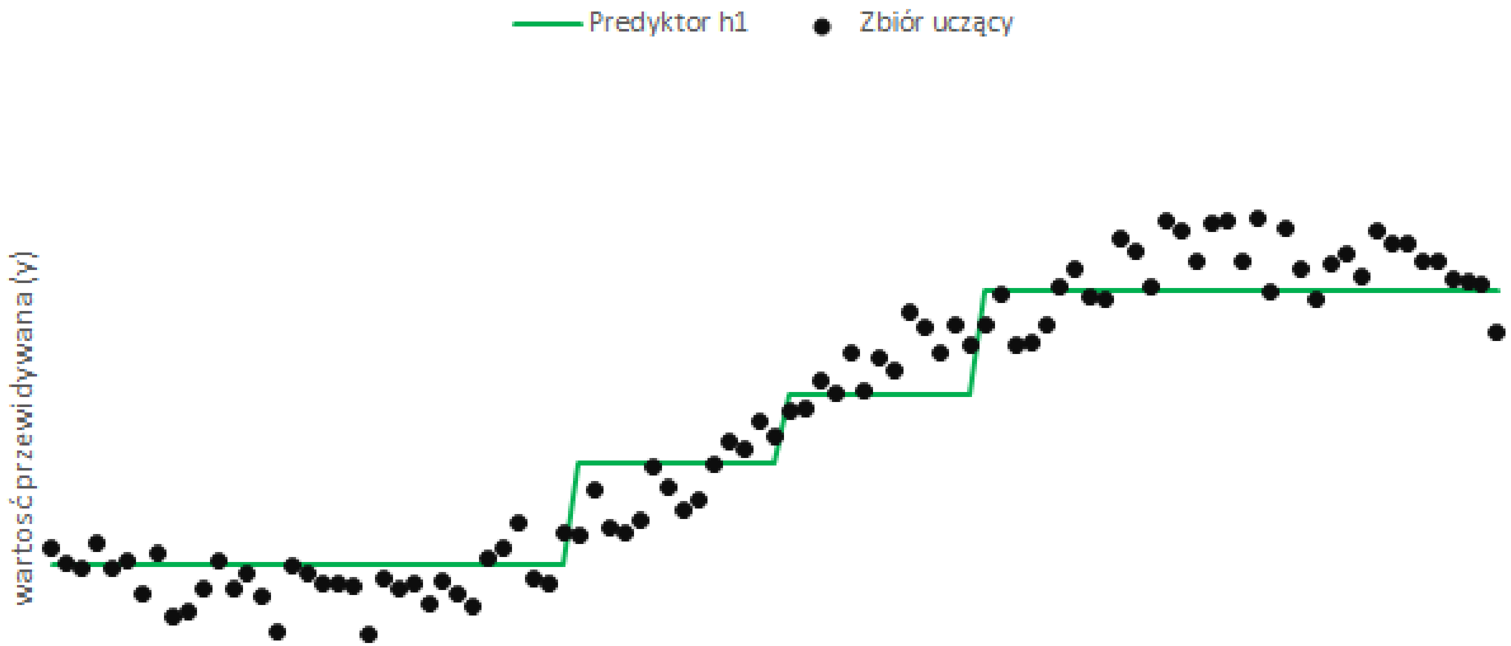
Idea

- **Gradient Boosting**
- **Gradient** – the vector field denoting the direction of greatest change of a scalar function.
- **Gradient** is used to show the direction, in which our model should adjust.
- **Boosting** means correcting our mistakes in each step.

Step 0

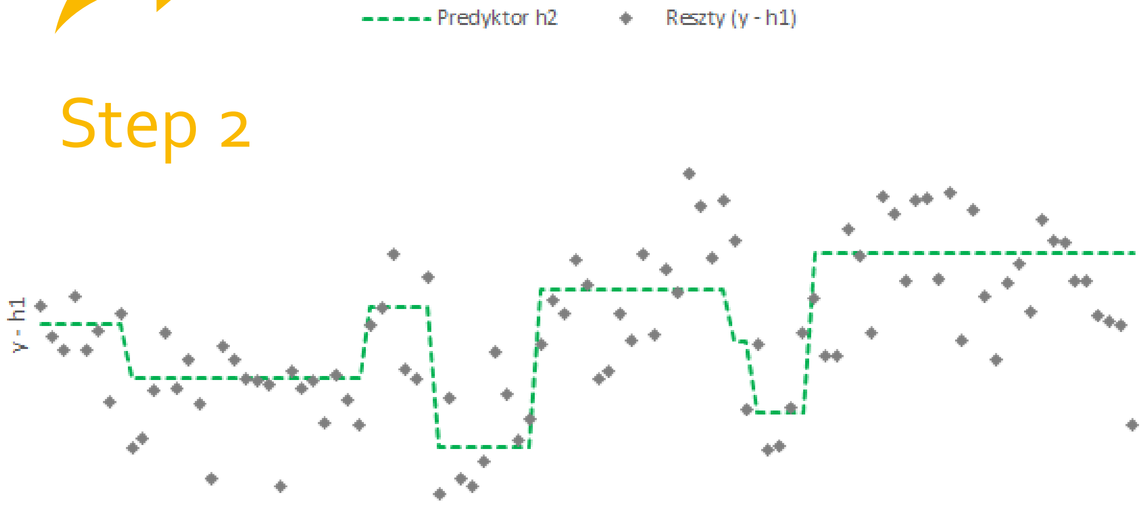
Gradient boosting

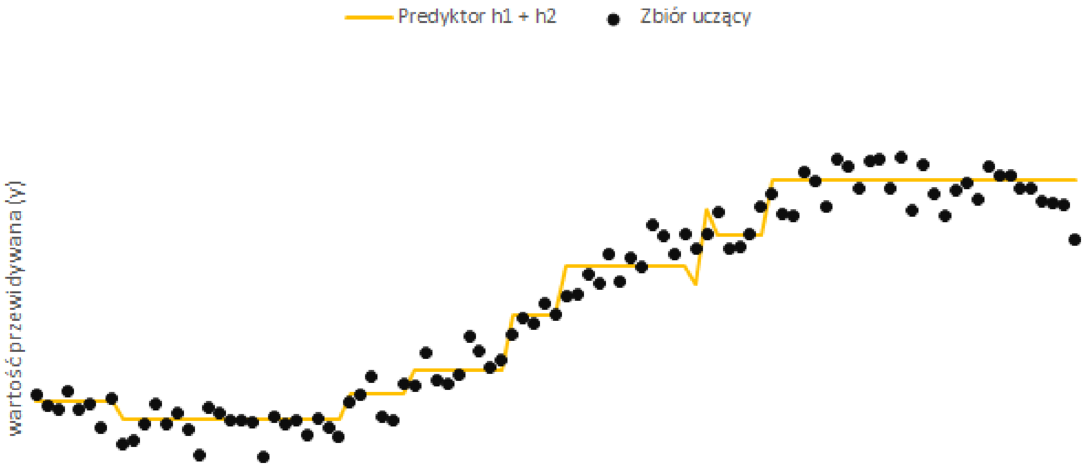
Step 1



First simple decision tree

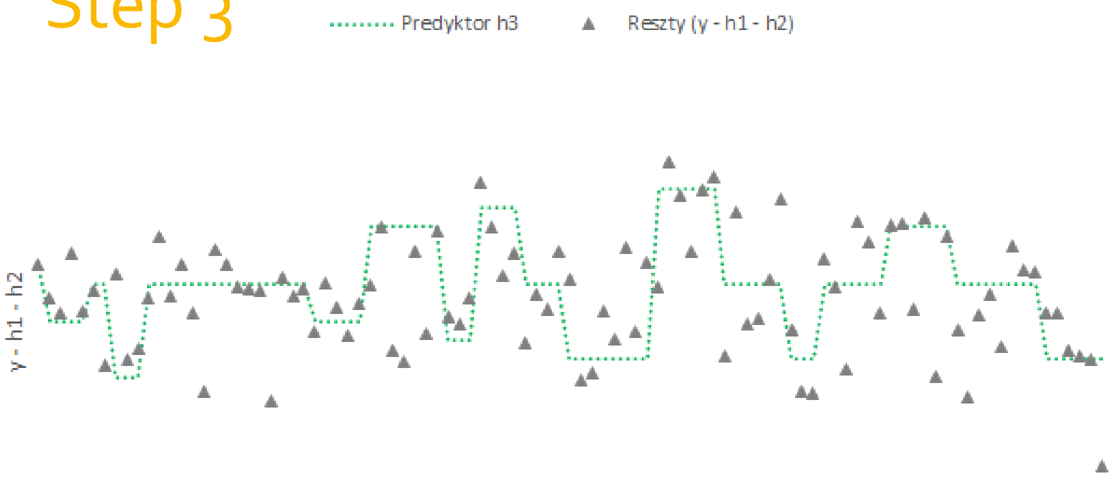
Step 2

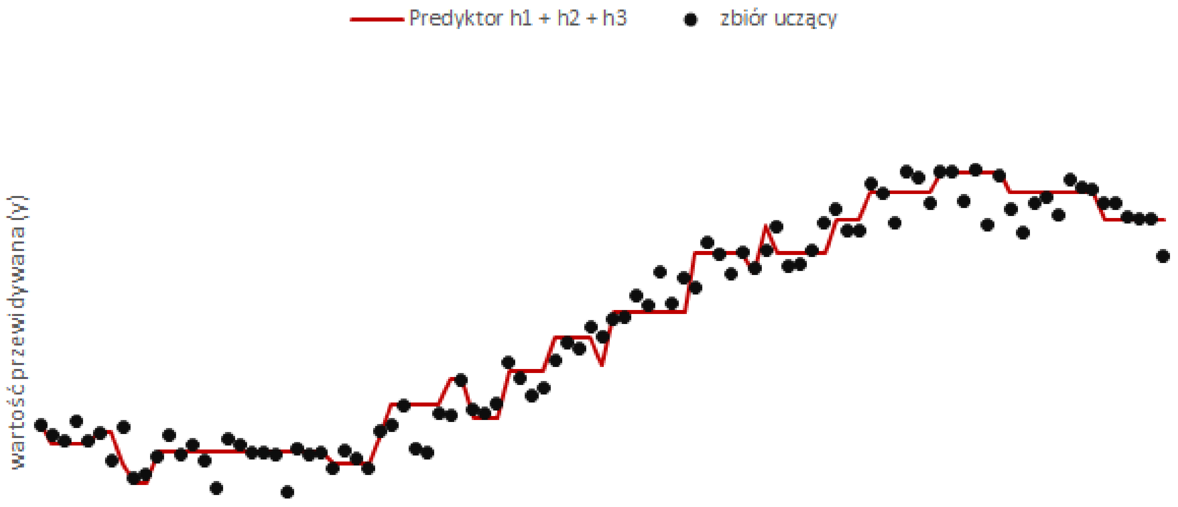




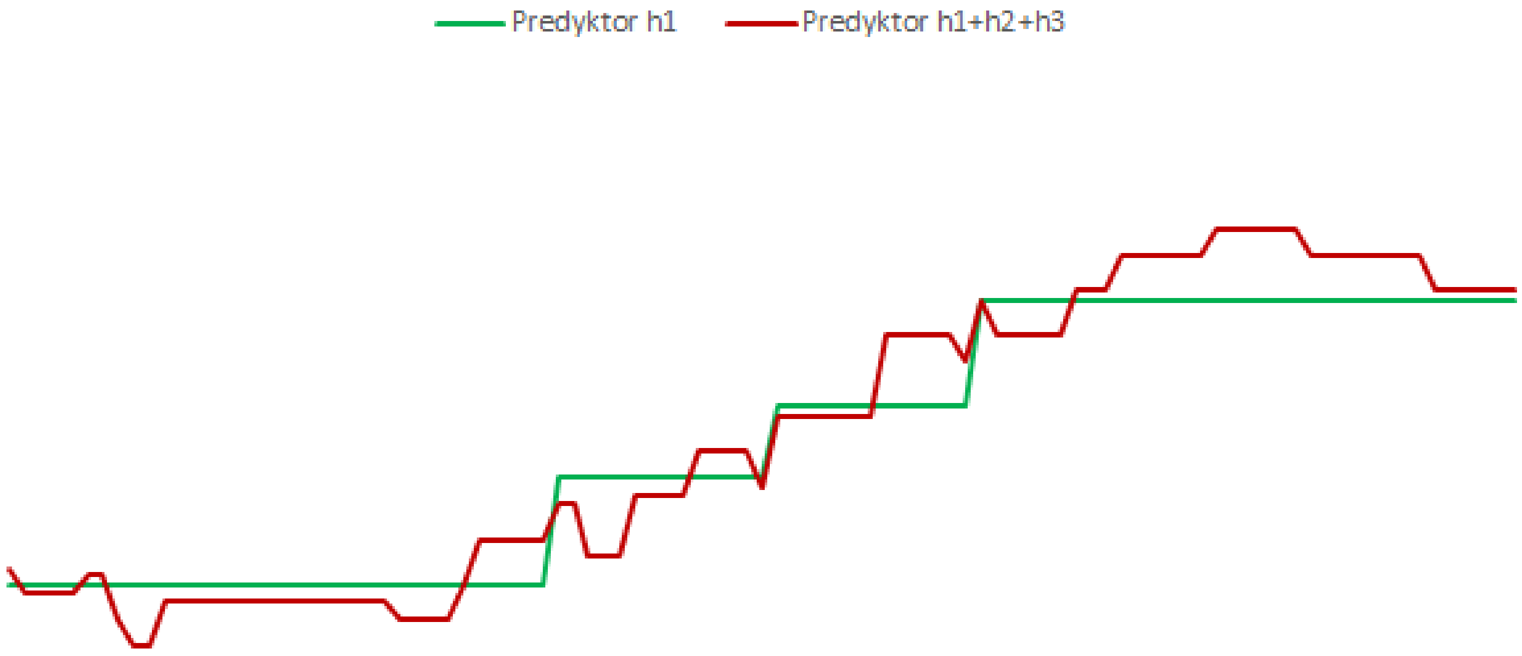
Correcting tree’s previous mistakes

Step 3





Repeating the second step



The difference between 1st and 3rd step

2. Materials & Algorithms

Preprocesing

1. Permutation
2. Labels: float to int:
    1. Regresion problems: divide into k+1 bucets
    2. Classification to 0 and 1
    3. Multiclasification 0,1,2,..
3. Categorical to numerical (prior = 0.05)

$$avg\_target = \frac{countInClass + prior}{totalCount + 1}$$

Object #	f_1	f_2	...	f_n	Function value
1	4	53	...	rock	0
2	3	55	...	indie	0
3	2	40	...	rock	1
4	5	42	...	rock	1
5	5	34	...	pop	1
6	2	48	...	indie	1
7	2	45	...	rock	0
...					

Object #	f_1	f_2	...	f_n	Function value
1	4	53	...	0,05	0
2	3	55	...	0,05	0
3	2	40	...	0,025	1
4	5	42	...	0,35	1
5	5	34	...	0,05	1
6	2	48	...	0,025	1
7	2	45	...	0,5125	0
...					

Preprocesing

- It's possible to do just simple One-Hot encoding on categorical data by giving parameter: `one_hot_max_size` it's determine maximum number of unique value in categorical data that One-Hot will by perform on

Text processing:

1. Tokenization by space " "
2. Dictionary of words/letters(`token_level_type`)
3. Bag of words (check if word appear in sentence)

`gram_order` gives us opportunity to prerform n-gram method

With parameters `occurence_lower_bound` and

`max_dictionary_size` we can control size of dictionary to avoid considering very rare words

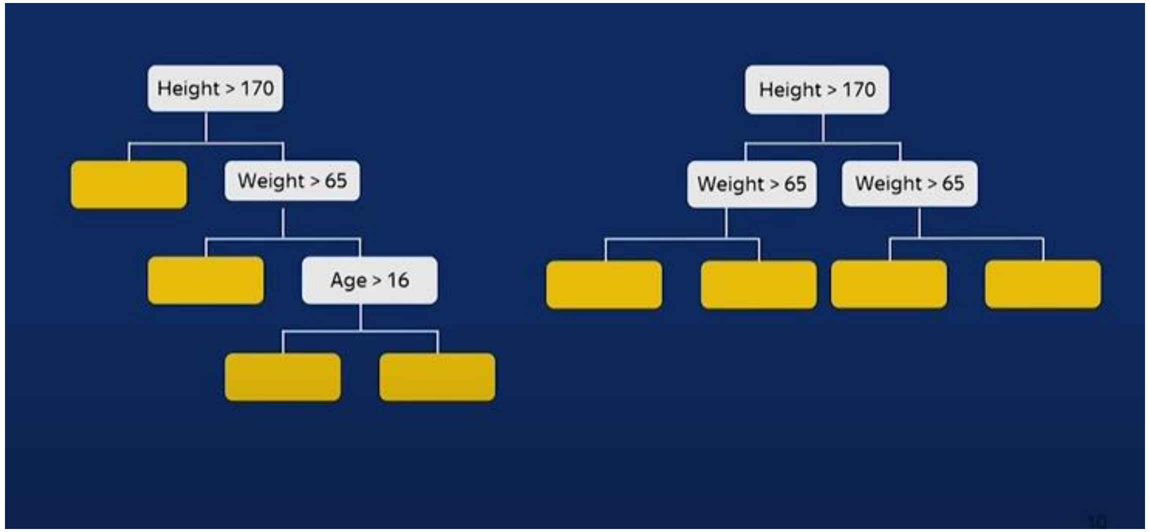
Creating trees

Symetric trees

shallow trees

avoid overfitting

improve speed



To aviod problem with bias during creating trees, (as like it's appearing on basic XGBoost models) Catboost crating log(n) models based only on observation that has been used in the past

Pareameters

- `iterations` – max number of trees (too small might couse underfitting, too many overfitting)
- `use-best-model` – save best model on validation set according to eval-metric
- `eval-metric` – metric we want to use for validation (example: `eval-metric = 'R2'`, `eval-metric = 'Quantile:alpha=0.3'`)
- grow_policy - form ("SymmetricTree ", "Depthwise " untill it's worth to divide , "Lossguide " - until proper amount of leaves)
- max_leaves, min_data_in_leaf – menaging tree size

Pareameters

- `learning_rate – indicate how big steps each itteration does`
- `depth – of singe tree (4-10 recomended)`
- l2_leaf_reg – value of l2 multiplayer
- random_strength – adds more randomnes into splits (to avoid overfitting)
- has-time – use if order is important so not gonna be suffle

Golden features

If the dataset has a feature, which is a strong predictor of the result, the pre-quantisation of this feature may decrease the information that the model can get from it. It is recommended to use an increased number of borders (1024) for this feature.

```
per_float_feature_quantization=['0:border_count=1024', '1:border_count=1024']
```

Main advantages

- Works great for data **from different sources** and when **there's not much** training data.
- **No need** to transform your data into numeric type.
- Offers Python interfaces integrated with scikit, as well as R and command-line interfaces.

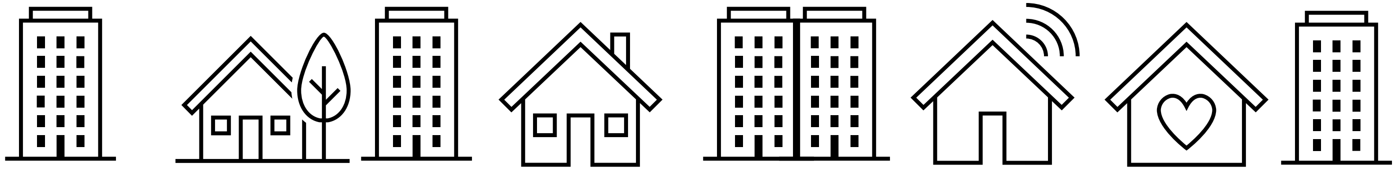
3. Usage Example

Preprocessing for CatBoost model

- Data format should be **data.frame** or **matrix** with features.
- Catboost is **resistant** to skewed and/or correlated data
- This model is also handles well categorical variables
- One hot encoding during preprocessing is **not recommended**
- CatBoost can handle missing values internally

What to do:

- Combine factor levels with low frequency
- Remove no variance predictors



Step 1 - load package

```
library(catboost)
```

Step 2 – load dataset

```
# load libraries
library(mlbench)

# attach the BostonHousing dataset
data(BostonHousing)
```

Step 3 – split the dataset

```
#caret library
library(caret)


# Split out validation dataset
# create a list of 80% of the rows in the original dataset we can use
for training
set.seed(7)
validation_index <- createDataPartition(BostonHousing$medv, p=0.80,
list=FALSE)
# select 20% of the data for validation
validation <- BostonHousing[-validation_index,]
# use the remaining 80% of data to training and testing the models
dataset <- BostonHousing[validation_index,]
```

Step 4 – split the predicators and dependent variable

```
library(dplyr)
y_train <- unlist(dataset[c('medv')])
X_train <- dataset %>% select(-medv)


y_valid <- unlist(validation[c('medv')])
X_valid <- validation %>% select(-medv)
```

Step 5 – convert to CatBoost specific format

```
train_pool <- catboost.load_pool(data = X_train, label = y_train)
test_pool <- catboost.load_pool(data = X_valid, label = y_valid)
```

Step 6 – specify parameters for CatBoost regression

```
params <- list(iterations=500,
learning_rate=0.01,
depth=10,
loss_function='RMSE',
eval_metric='RMSE',
random_seed = 55,
od_type='Iter',
metric_period = 50,
od_wait=20,
use_best_model=TRUE)
```

Step 7 – train the model

```
model <- catboost.train(learn_pool = train_pool,params = params)
```

Step 8 – predict the output

```
y_pred=catboost.predict(model,test_pool)
```

Step 9– calculate the error

```
#calculate error metrics
postResample(y_pred,validation$medv)


#output
RMSE      Rsquared      MAE
3.1027671 0.8670278 2.2757869
```

Received RMSE value is **3.1**, while RMSE for random forest regression would be **3.88**

4. Summary Reminder

What is the type of CatBoost Algorithm?

CLASSIFICATION

Main features of CatBoost

CatBoost

plain or ordered boosting
(modification of standard gradient boosting algorithm)

processing categorical features

outperforming gradient boosted decision trees

open-source

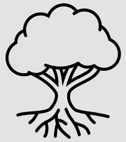
FUN FACT: Plain Boosting vs Ordered Boosting

- PLAIN BOOSTING:
- Bigger datasets
- Used when bias is smaller
- Similar to GBDT procedure

- ORDERED BOOSTING:
- Smaller datasets
- Used when bias is bigger
- Sampling new permutation at each iteration
- Calculating gradient by averaging predecesing coeffs

What should we think when we hear 'CatBoost?'

1. Gradient boosted **binary decision trees**



2. Classification

3. Minimizing the expected loss with a sequence of iterative approximations

101	001
110	111

4. Grouping all categories to clusters, one-hot encoding

5. TS (target statistic) as numerical

10

'Positive' Aspects

- 1. Works well with not much training data **– no need of collecting a lot of data**
- 2. Straightforward parameters – **quick coding with immediately good score**
- 3. Compatibility with **TensorFlow, Keras, Apple's core ML, Python, R**
- 4. **Easily-accessible tutorials** to learn

Where to read about CatBoost?

Sources

- 1. https://catboost.ai/ - official page of the library with huge documentation
- 2. Tutorials for learning
  https://github.com/catboost/tutorials
  https://medium.com/ampersand-academy/how-to-create-regression-model-using-catboost-package-in-r-programming-6cce3805a5e1
  https://www.r-bloggers.com/2020/08/how-to-use-catboost-with-tidymodels/
- 3. Articles:
- 'CatBoost: unbiased boosting with categorical features', L. Prokhorenkova , G. Gusev , A. Vorobev, A.V. Dorogush, A. Gulin
- 'Performance of CatBoost classifier and other machine learning methods', A. Ibrahim, M. M. Muhammed, S. O. Sowole, R. Raheem, R. O. Abdulaziz

Thanks for listening