



Modele oparte na drzewach

Drzewa decyzyjne, lasy losowe, boosting
Laboratorium 6

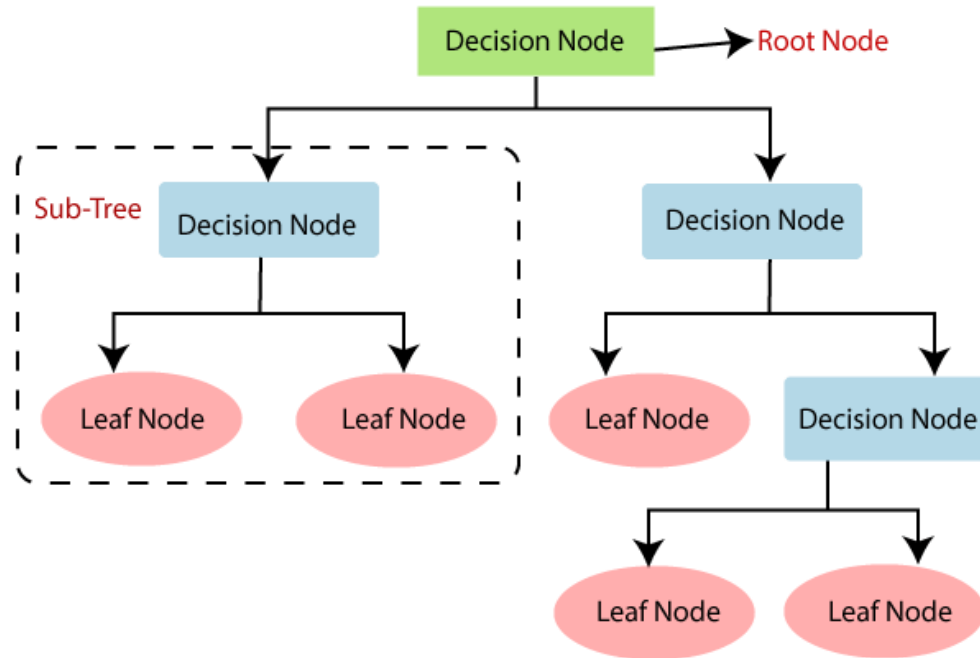
Sebastian Kuzara

KRUK S.A.

Statistical Methods Development Area

Wrocław, 2024

Drzewa decyzyjne



<https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm>

Podział na podstawie:

- Gini index dla modelu klasyfikacyjnego
- RMSE w modelu regresyjnym

Dodatkowy tutorial: <https://towardsdatascience.com/decision-trees-explained-3ec41632ceb6>

Drzewa decyzyjne - hiperparametry



```
rpart::rpart.control(
```

```
# Podstawowe:
```

```
  minsplit = 20, # minimalna liczba obserwacji w węźle
```

```
  minbucket = round(minsplit/3), # minimalna liczba obserwacji w węźle-liściu
```

```
  cp = 0.01, # o ile min. ma się poprawić metryka podziału (Gini/RMSE), aby do tego podziału doszło
```

```
  maxdepth = 30, # maksymalna głębokość drzewa (gdzie root=0), maksymalna liczba poziomów podziału
```

```
# Pozostałe:
```

```
  maxcompete = 4,
```

```
  maxsurrogate = 5,
```

```
  usesurrogate = 2,
```

```
  xval = 10,
```

```
  surrogatestyle = 0, ...)
```

Przykładowe wywołanie modelu drzewa losowego w R:

```
model_tree <- rpart::rpart(
```

```
  formula = Gender~.,
```

```
  data=cases,
```

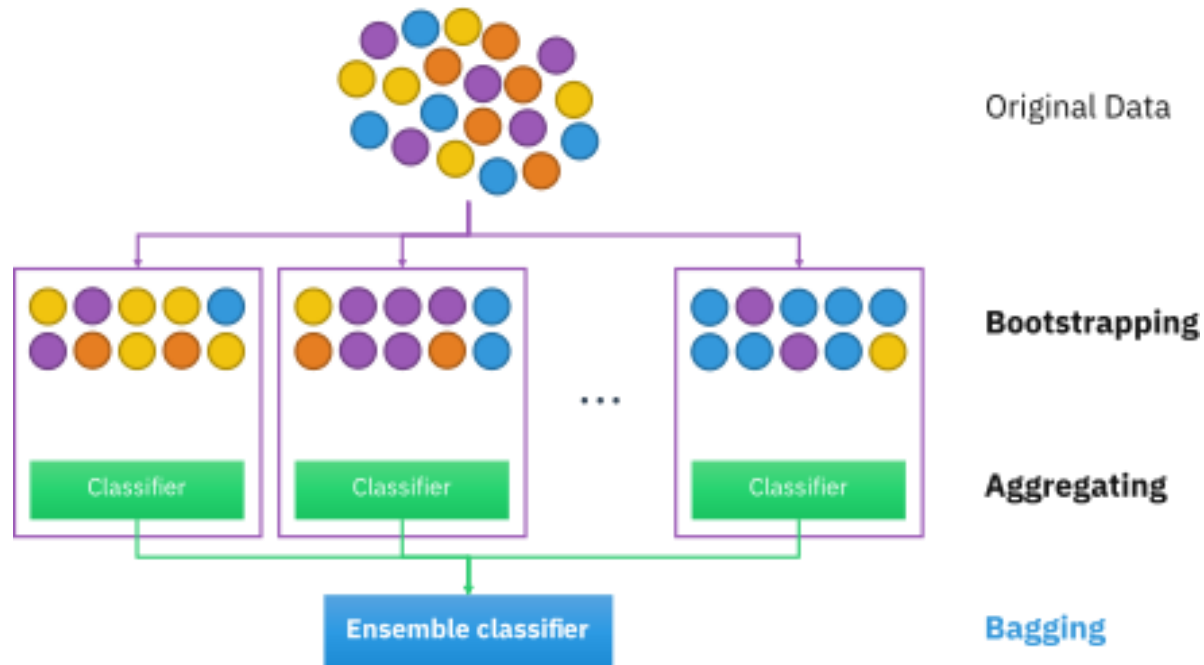
```
  method="class",
```

```
  control=rpart.control(minsplit = 1000, cp=0.0001)
```

```
)
```

Lasy losowe

- przykład ensemble model - predykcje poprzez głosowanie wielu modeli
- Bagging



<https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>

Drzewa decyzyjne - hiperparametry



```
library(randomForest)
```

```
# Podstawowe HP:
```

```
## ntree=500 - liczba submodeli drzew losowych
```

```
## mtry - liczba cech jaka ma być wylosowana do pojedynczego modelu drzewa decyzyjnego
```

```
## sampsize - liczba obserwacji jaka ma być losowana do modelu drzewa decyzyjnego
```

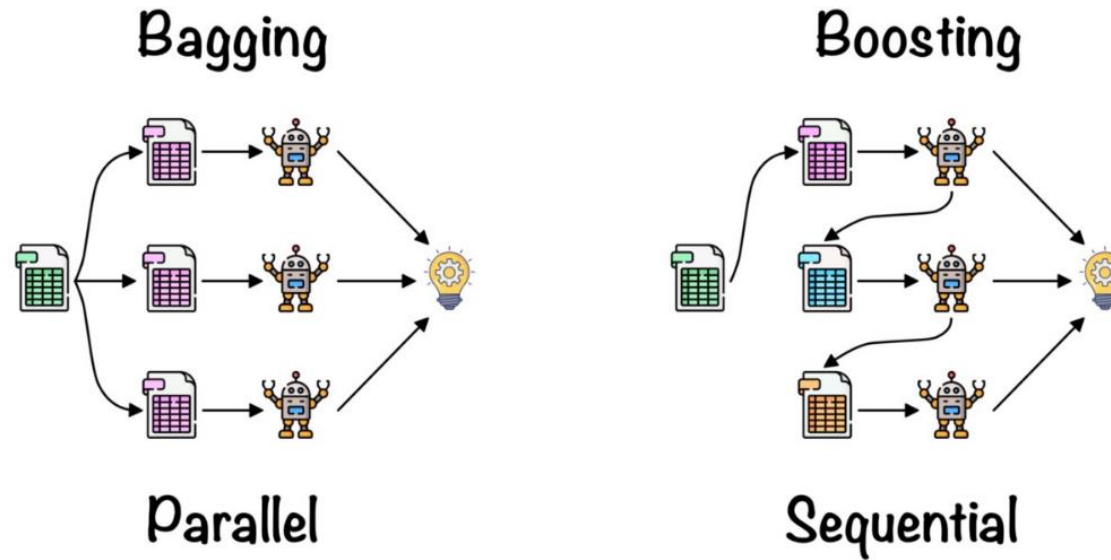
```
## nodesize - minimalna wielkość węzła-liścia w modelu drzewa (odpowiednik minbucket w rpart)
```

```
## maxnodes - maksymalna liczba węzłów liści
```

```
rf <- randomForest(  
  formula=TOA~.,  
  data=cases,  
  nodesize=1000,  
  ntrees=500,  
  maxnodes=2^6,  
  na.action = "na.omit")
```

Boosting

- AdaBoost (tylko klasyfikacja)
- GBM
- LightGBM
- XGBoost



<https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>

Dodatkowy materiał:

<https://towardsdatascience.com/boosting-algorithms-explained-d38f56ef3f30>

GBM - hiperparametry



```
library(gbm)

boost_model <- gbm(
  TOA~Age+Principal+MeanSalary+LastPaymentAmount,
  data=cases,
  distribution = "gaussian", # w zależności czy klasyfikacja czy regresja
  n.trees = 500, # liczba submodeli drzew
  interaction.depth = 6, # max głębokość 1 drzewa
  shrinkage = 0.01, # learning rate, "o ile kolejne modele wpływają na predykcje"
  n.minobsinnode = 1000, # min. liczba obserwacji w węźle-liściu
  bag.fraction = 0.5, # jaki % cech wylosować do pojedynczego submodelu drzewa )
```

Przykładowe postępowanie przy optymalizacji HP:

1. ustalamy “małą” liczbę drzew i stosunkowo “duży” learning rate
2. sprawdzamy różne kombinacje głębokości, minimalnej liczby obserwacji w liściach oraz % cech w submodelu
3. dla najlepszej (najlepszych) kombinacji z punktu 2. testujemy różne warianty liczby drzew i learning rate
4. wybieramy wartości, kiedy model przestaje się istotnie poprawiać na zbiorze walidacyjnym

Metody optymalizacji hiperparametrów



- grid search
- random search
- optymalizacja bayesowska (przykładowa implementacja: rBayesianOptimization)