Adaboost

Ce ne așteaptă?

1. Esența mecanismului boosting
2. Esența Adaboost
3. Instrumente Scikit-Learn

1. Esența mecanismului boosting

- Conceptul de boosting nu este un algoritm ci o metodologie aplicată asupra altor algoritmi Machine Learning
- Algoritm Machine Learning supus metodologiei boosting se va considera un algoritm Meta-Learning
- Algoritmul Meta Learning se va determina suma mai multor algoritmi simpli înmulțiți cu un coeficient

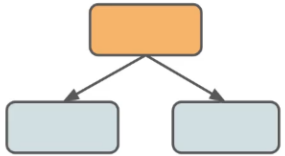
$$F_T(x) = \sum_{t=1}^{T} f_t(x) \quad \text{unde} \quad f_t(x) = \alpha_t h(x)$$

$h(x)$ - algoritm simplu (stump)

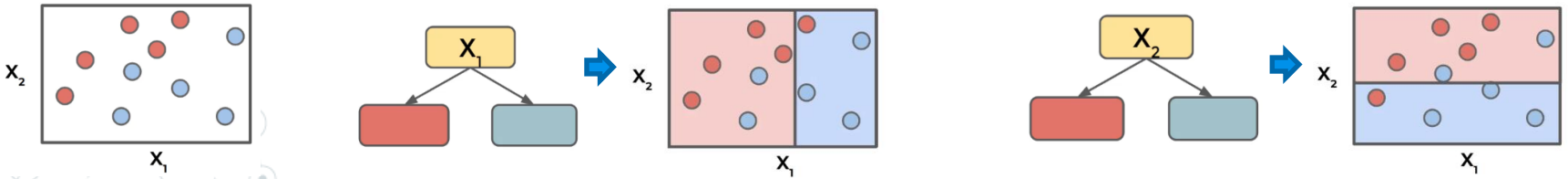
- Algoritm simplu numit și stump, se consideră orice algoritm Machine Learning ce nu poate asigura rezultate satisfăcătoare, în special arbori de decizie

2. Esența Adaboost

- Adaboost (Adaptive Boosting) este un mecanism boosting aplicat consecutiv asupra algoritmilor simple (stump-uri) cu ajustarea datelor greșite în algoritmii anteriori
- Stump se va considera un arbore de decizie format dintr-un nod rădăcina și 2 noduri terminale

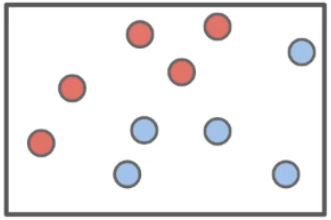


- Algoritmul simplu nu poate asigura rezultate satisfăcătoare



- **Pașii algoritmului Adaboost:**

1. Se analizează datele și se atribuie ponderi tuturor datelor



Numărul datelor $n = 10$

Ponderea datelor $pond_{date} = \frac{1}{n} = \frac{1}{10}$

2. Se elaborează arborele de decizie simplu pentru caracteristica cu cea mai mică impuritate Gini și se aplică asupra datelor

3. Se determina eroarea totală a acestui arbore

$$eroare_{total} = n_{puncte\_eronate} * pond_{date} = 3 * \frac{1}{10} = 0,3$$

4. Se determina performanța acestui arbore

$$perf = \frac{1}{2} ln\left(\frac{1-eroarea_{total}}{eroarea_{total}}\right) = \frac{1}{2} ln\left(\frac{1-0,3}{0,3}\right) = 0,42$$

5. Se actualizează ponderile datelor eronate și a celor corecte

$$pond_{er} = pond_{date} * e^{perf} = 0,1 * e^{0,42} = 0,15$$

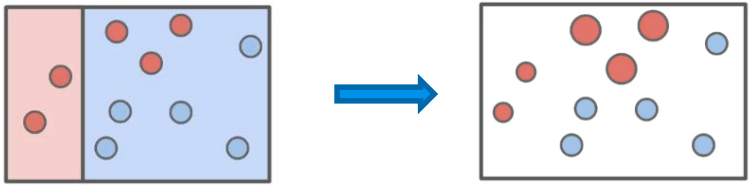
$$pond_{cor} = pond_{date} * e^{-perf} = 0,1 * e^{-0,42} = 0,06$$

6. Se normalizează ponderile actualizate

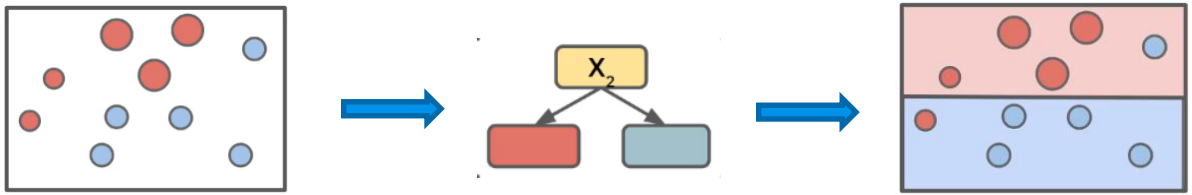
$$pond_{er\_nor} = \frac{pond_{er}}{pond_{er}+pond_{cor}} = \frac{0,15}{0,15+0,06} = 0,71$$

$$pond_{ecor\_nor} = \frac{pond_{cor}}{pond_{er}+pond_{cor}} = \frac{0,06}{0,15+0,06} = 0,29$$

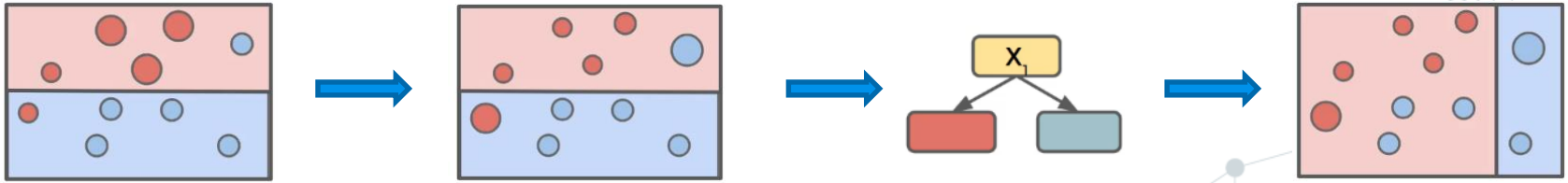
7. Crearea unui nou set din n date selectate aleator din setul inițial ținându-se cont de noile ponderi (datele care au fost greșite vor avea o probabilitate mai marea de includere în set)



8. Se repetă pașii 1-7 pe noul set de date creându-se un nou arbore simplu

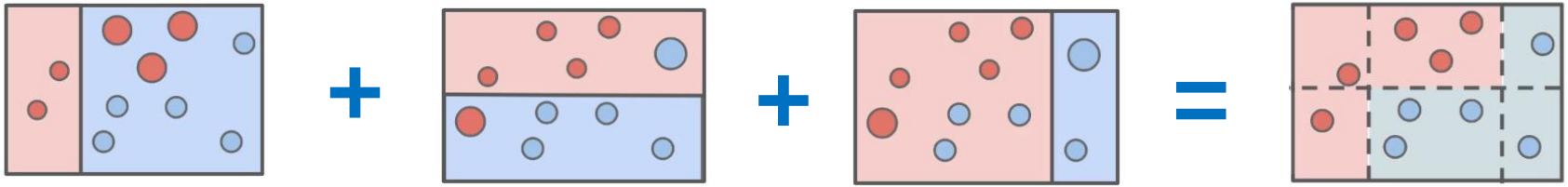


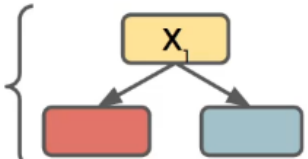
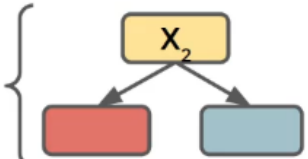
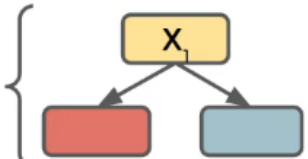
9. Procedura se repetă până la atingerea numărului de arbori specificat



10. Modelul Meta Learning se obține prin sumarea produsului dintre fiecărui arbore și performanța sa

$$F_T(x) = \sum_{t=1}^{T} perf * h(x)\,, \text{unde T} - \text{numarul de arbori}$$



$$F_3(x) = perf_1 * \left\{ \text{  } \right\} + perf_2 * \left\{ \text{  } \right\} + perf_3 * \left\{ \text{  } \right\}$$

3. Instrumente Scikit-Learn

- Se importa clasa algoritmului AdaBoostClassifier

```
from sklearn.ensemble import AdaBoostClassifier
```

- Se creează un model cu fixarea valori hiper-parametrului:
  - n_estimators - numărul de arbori ai algoritmului (valori posibile = int, implicit =50)

```
model = AdaBoostClassifier(n_estimators =18)
```

- Se realizează trainingul modelului pe datele de training

```
model.fit(X_train, y_train)
```

- Se vizualizează importanța fiecărei caracteristici

```
model.feature_importances_
```

- Se realizează predicția pe datele de test

```
y_pred = model.predict(X_test)
```