Gradient Boosting

Ce ne așteaptă?

1. Esența Gradient Boosting

2. Instrumente Scikit-Learn

1. Esența Gradient Boosting

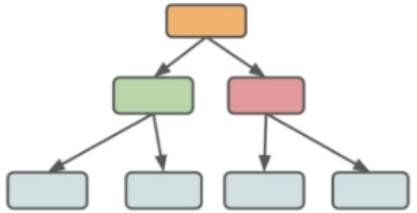
- Gradient Boosting este un mecanism boosting aplicat consecutiv asupra algoritmilor simpli prin utilizarea valorii erorii reziduale și coeficienți de valoarea egalăi
- Pașii algoritmului Gradient Boosting pentru sarcini de regresie

1. Se elaborează algoritmul simplu și se obțin rezultatele de predicții pe datele de training $\hat{y}$

2. Se determină valoarea erorii reziduale e

$$e = y - \hat{y}$$

X_1	X_2	X_3	y	$\hat{y}$	e
200	3	2	500	430	70
190	2	1	462	395	67
230	3	3	565	602	-37



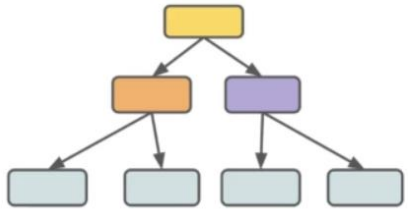
3. Se elaborează algoritmul simplu și se realizează trainingul acestuia considerându-se drept etichetă valoarea erorii reziduale e

4. Se obține rezultatele predicției f_1 a acestui nou algoritm pe datele de training

5. Actualizarea predicției $\hat{y}_1$ algoritmului Mega-Learning considerând predicția primului algoritm simplu și a predicția celui de al doilea înmulțită cu un coeficient al ratei de training (learning rate) $\alpha = (0 \div 1)$

$$\hat{y}_1 = \hat{y} + \alpha * f_1$$

X_1	X_2	X_3	y	$\hat{y}$	e	f_1	$\hat{y}_1(\alpha = 0,1)$
200	3	2	500	430	70	85	438,5
190	2	1	462	395	67	59	400,9
230	3	3	565	602	-37	-40	598



6. Se repetă pașii 2-5 considerându-se noile erori reziduale $e_n = y - \hat{y}_n$

- În cadrul sarcini de clasificarea se vor considera:

1. Valoarea prezisă $\hat{y}$ se va transforma în probabilitatea de prezicere $\hat{p}$:

$$\hat{p} = \frac{e^{\hat{y}}}{1 + e^{\hat{y}}}$$

2. Eroarea reziduală se va determina conform probabilității de prezicere $\hat{p}$:

$$e = y - \hat{p}$$

3. Valoarea predicției f_1 se va transforma în unități logaritmice ale cotei:

$$F_{f_1} = \frac{f_1}{\hat{p} * (1 - \hat{p})}$$

4. Actualizarea predicției $\hat{y}_1$ algoritmului Mega-Learning:

$$\hat{y}_1 = \hat{y} + \alpha * F_{f_1}$$

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

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

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

2. Instrumente Scikit-Learn

- Se importa clasa algoritmului GradientBoostingClassifier

```
from sklearn.ensemble import GradientBoostingClassifier
```

- Se creează un model cu fixarea valori hiper-parametrului:
  - n_estimators - numărul de arbori ai algoritmului (valori posibile = int, implicit =100)
  - max_depth – numarul maxim de divizari a datelro în arbore (valori posibile = int, implicit =3)

```
model = GradientBoostingClassifier(n_estimators =100, max_depth=4)
```

- 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)
```