

XGBoost

1. Cara kerja Algoritma


XGBoost bekerja dengan menggunakan beberapa pohon/tree yang dibuat secara berurutan/sequential. Pohon dibuat dengan tujuan memperbaiki error dari pohon sebelumnya.

Pada permulaan, nilai prediksi diatur dengan sembarang angka, biasanya bernilai 0.5. Kemudian, pseudo residual dihitung dengan menghitung pengurangan antara data label sebenarnya dengan nilai prediksi. Pembagian cabang pada pohon akan berpusat pada pseudo residual ini.

Untuk membentuk pohon, dilakukan langkah berikut:

- 1) Dimulai dengan semua residual sebagai sebuah daun sekaligus root.
- 2) Similarity score dihitung pada setiap daun dengan rumus:

$$\text{Similarity Score} = \frac{(\sum residuals)^2}{len(residuals) + \lambda}$$


Regularization parameter

Source: <https://medium.com/@prathameshsonawane/xgboost-how-does-this-work-e1cae7c5b6cb>

- 3) Untuk semua fitur pada setiap anggota daun dilakukan:
 - a. Diurutkan dari terkecil ke terbesar.
 - b. Dihitung rata-rata antara nilai anggota satu dengan anggota setelahnya.
 - c. Pilih satu nilai rata-rata lalu bagi anggota menjadi dua node berdasarkan nilai fitur, satu node untuk yang lebih kecil dari rata-rata, sedangkan node lain untuk yang lebih dari rata-rata.
 - d. Hitung gain dari node yang terbentuk dengan rumus:

$$\text{Gain} = \text{ss}(\text{left_child}) + \text{ss}(\text{right_child}) - \text{ss}(\text{Root})$$

Source: <https://medium.com/@prathameshsonawane/xgboost-how-does-this-work-e1cae7c5b6cb>

- 4) Dari semua kemungkinan splitting berdasarkan langkah 3, dipilih satu dengan nilai gain terbesar.
- 5) Langkah 3 dan 4 diulangi untuk setiap node yang terbentuk hingga pohon mencapai maksimal depth/kedalaman yang diinginkan (biasanya 6) atau jumlah anggota pada masing-masing node akhir bernilai satu atau sama dengan minimal jumlah anggota pada satu daun.

Setelah pohon terbentuk dibentuk, pohon selanjutnya dibuat dengan residual yang sudah diupdate dengan:

- 1) Dikalkulasi nilai output untuk masing-masing node daun pada pohon dengan rumus:

$$\text{Output Value} = \frac{\sum residuals}{len(residuals) + \lambda}$$

\downarrow
Regularization parameter
(default 0)

Source: <https://medium.com/@prathameshsonawane/xgboost-how-does-this-work-e1cae7c5b6cb>

- 2) Pohon ditelusuri hingga didapatkan node daun yang bersesuaian dengan data.
- 3) Nilai prediksi saat ini ditambahkan dengan learning rate dikali dengan output node.
- 4) Ulangi langkah 1 sampai 3 untuk setiap data.

Prediksi dilakukan dengan cara: Nilai prediksi awal ditambah dengan jumlah nilai pada masing-masing pohon. Nilai yang dimaksud adalah output pada node yang bersesuaian dikali dengan learning rate.

2. Perbandingan Hasil Evaluasi Model

```
Custom R2 Score: 0.5160388317136779
Sklearn R2 Score: 0.5232801472673129
Custom Mean Squared Error: 18006.4230991446
xgboost lib Mean Squared Error: 17737.00026070723
```

Score yang dihasilkan model saya sedikit lebih buruk daripada model oleh XGBoost library. Hal ini mungkin disebabkan oleh beberapa faktor:

a. Detail implementasi Pembangunan Pohon

Cara pohon dibangun, cara pemisahan yang digunakan, dan cara nilai dihitung pada daun mungkin sedikit berbeda antara implementasi saya dan XGBoost library. XGBoost library telah dioptimalkan selama bertahun-tahun dan mungkin menyertakan heuristik dan pengoptimalan tambahan yang tidak ada dalam model saya.

b. Presisi Kalkulasi

Pustaka XGBoost dapat menggunakan strategi yang berbeda untuk meningkatkan presisi dalam kalkulasi, yang berpotensi menyebabkan sedikit perbedaan dalam hasil karena perbedaan presisi numerik tersebut.

c. Perbedaan Parameter

XGBoost library memiliki parameter yang jauh lebih banyak daripada model saya dengan default parameternya. Hal ini bisa menyebabkan perbedaan pohon yang dihasilkan.

d. Subsampling

Walaupun menggunakan presentase subsampling yang sama, sample yang dipilih bisa berbeda karena menggunakan teknik randomize yang berbeda.

Perbedaan ini juga bisa menyebabkan berbedanya pohon yang dihasilkan

3. Improvement

Untuk mempercepat waktu kalkulasi, model perlu menggunakan sparse matriks.

Selain itu, penambahan hyperparameter yang lebih bervariasi juga bisa menambah fleksibilitas model dan mempermudah hyperparameter tuning untuk mendapatkan score yang lebih baik. Penambahan random seed untuk konsistensi hasil.