Algoritma Random Forest

Oleh Kelompok 5

Anggota Kelompok

Alfonsus William Hamonangan Sinaga

234311005

Mohammad Fakhriza Maftukhin

234311018

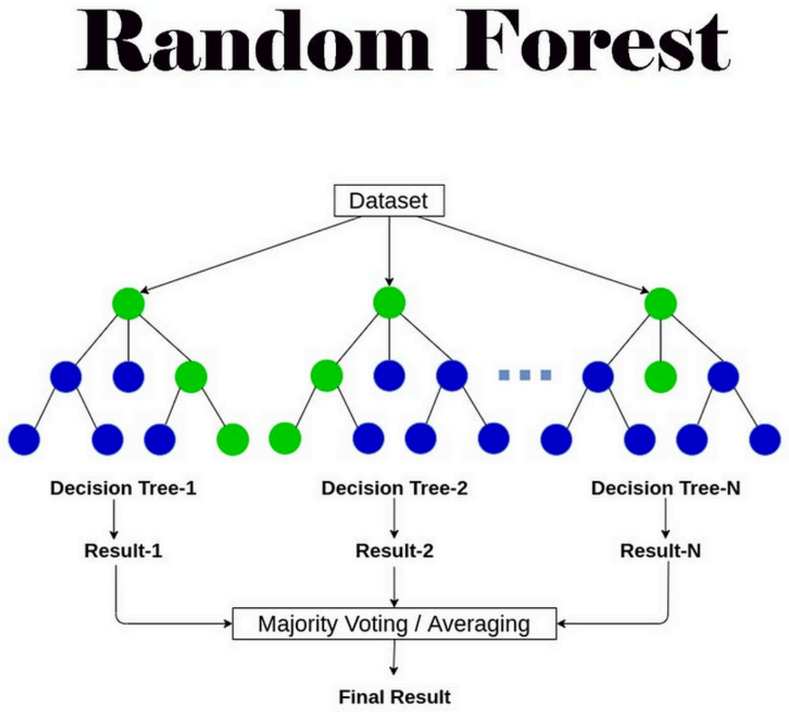
Muhammad Aulia Alfarouq

234311020

Definisi Random Forest

Random Forest adalah algoritma **supervised learning** yang termasuk dalam metode **ensemble learning**. Ensemble learning adalah menggabungkan beberapa model pengklasifikasi yang lebih lemah untuk memecahkan masalah yang lebih kompleks

- Bekerja dengan membangun sejumlah besar *decision trees* (pohon keputusan) pada saat training.
- Untuk **klasifikasi**, hasil akhirnya adalah *voting* (suara terbanyak) dari semua pohon.
- Untuk **regresi**, hasil akhirnya adalah *rata-rata* (average) dari semua pohon.
- Menggabungkan banyak model (pohon) untuk menghasilkan prediksi yang lebih akurat dan stabil.



Konsep Inti

Ensemble Learning

Ide menggabungkan beberapa model (disebut "weak learners") untuk menciptakan satu model ("strong learner") yang lebih kuat dan akurat.

Bagging

(Bootstrap Aggregating). Membuat banyak subset data secara acak dari data training (dengan *replacement*) untuk melatih setiap pohon secara independen.

Random Subspace

Saat membelah *node*, algoritma hanya mempertimbangkan subset fitur acak, bukan semua fitur. Ini memastikan pohon-pohonnya beragam (tidak identik).

Hipothesis Function(H(x))

“Hipotesis function h(x) adalah prediksi yang dibuat oleh satu pohon dalam Random Forest. Ia bekerja berdasarkan aturan IF–THEN yang ada di dalam pohon keputusan. Jadi setiap pohon punya pendapat sendiri dulu, dan pendapat tiap pohon inilah yang nantinya digabungkan.”

Pohon Tunggal (h(x))

Fungsi hipotesis untuk satu *decision tree* adalah serangkaian aturan IF-THEN yang mempartisi ruang fitur.

Forest (H(x))

Hipotesis untuk *Random Forest* adalah agregasi dari semua pohon (K). Ini adalah fungsi non-linear yang kompleks.

Klasifikasi (Voting):

$$H(x) = \text{mode}\{h_1(x), h_2(x), \ldots, h_K(x)\}$$

H(x)
→ Prediksi akhir dari keseluruhan Random Forest untuk input x. Ini adalah hasil gabungan dari semua pohon.
$h_1(x), h_2(x), \ldots, h_k(x)$
→ Prediksi dari masing-masing pohon k dalam hutan (forest). Setiap pohon memberikan satu suara/class prediction untuk data x.
mode{…}
→ Nilai yang paling sering muncul (mayoritas). Dalam klasifikasi, Random Forest mengambil hasil voting terbanyak dari semua pohon.
K
→ Jumlah total pohon dalam forest.

Cost Function (Kriteria Split)

Random Forest tidak memiliki satu "cost function" global. Sebaliknya, algoritma ini menggunakan **impurity measures** (pengukur ketidakmurnian) di setiap *node* untuk memutuskan *split* (pembelahan) terbaik.

Klasifikasi: Gini Impurity

Mengukur seberapa sering elemen yang dipilih secara acak akan salah diklasifikasikan. Tujuannya adalah meminimalkan Gini.

$$G = 1 - \sum_{i=1}^{C} (p_i)^2$$

- C
Jumlah kelas dalam node (misalnya: Churn & Not Churn → C = 2).
- p_i
Proporsi data untuk kelas ke-i pada node tersebut.
Contoh: jika 7 dari 10 data adalah “Not Churn”, maka p = 0.7.
- p_i^2
Probabilitas tersebut dikuadratkan untuk mengukur “kemurnian” kelas.
- Σ (sigma)
Menjumlahkan seluruh p_i^2 dari semua kelas.
- 1 - ...
Mengubah hasilnya menjadi ukuran impurity.
→ Semakin kecil nilai Gini, semakin murni node (lebih sedikit campuran kelas).

Regresi: Mean Squared Error (MSE)

Digunakan untuk regresi. Memilih split yang paling mengurangi varians (atau MSE) dalam node anak.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \&circvers;y)^2$$

- n
Jumlah total data dalam node.
- y_i
Nilai asli (ground truth) pada data ke-i.
- ŷ (y-hat)
Nilai prediksi di node tersebut
(biasanya rata-rata dari nilai-nilai dalam node).
- $(y_i - \hat{y})$
Selisih antara nilai asli dan prediksi.

- $(y_i - \hat{y})^2$
- Selisihnya dikuadratkan agar tidak bernilai negatif
- dan memberi penalti lebih besar untuk error besar.
- Σ (sigma)
- Menjumlahkan seluruh error kuadrat.
- 1/n
- Mengambil rata-rata dari semua error.

Hyperparameter Penting

n_estimators

Jumlah pohon (trees) dalam forest. Semakin banyak, performa semakin baik dan stabil, namun *training* lebih lama.

max_depth

Kedalaman maksimum setiap pohon. Jika terlalu dalam, bisa *overfitting*. Jika terlalu dangkal, bisa *underfitting*.

min_samples_split

Jumlah minimum sampel yang diperlukan untuk membelah (split) sebuah *internal node*. Berguna untuk mengontrol *overfitting*.

max_features

Jumlah fitur yang dipertimbangkan secara acak saat mencari *split* terbaik. Kunci untuk mengurangi korelasi antar pohon.

Kelebihan dan Kekurangan Random Forest

Kelebihan

Akurasi Tinggi: Menggunakan voting dari banyak pohon sehingga hasil lebih stabil dan seringkali lebih unggul dari model lain.
Mencegah Overfitting: Teknik Bagging dan Random Subspace memastikan variasi antar pohon, mencegah model terlalu spesifik pada data latih.
Robust terhadap Noise: Kurang sensitif terhadap data outlier (pencilan) dan data yang noisy (berisik).
Feature Importance: Secara otomatis menyediakan metrik seberapa penting setiap fitur dalam prediksi.

Kekurangan

Komputasi Mahal: Membangun ratusan pohon membutuhkan waktu training yang lebih lama dan sumber daya komputasi yang besar.
Kurang Interpretatif (Black Box): Karena agregasi dari banyak pohon, sulit untuk menjelaskan *mengapa* model membuat prediksi tertentu (dibanding Decision Tree tunggal).
Lambat untuk Real-Time: Waktu prediksi (inference) bisa lebih lambat karena harus melewati semua pohon untuk mendapatkan hasil voting.

Kapan Menggunakan & Menghindari Random Forest

⭐ Waktu Terbaik Penggunaan

Membutuhkan Akurasi Maksimal: Ketika akurasi prediksi adalah prioritas utama dan Anda memiliki sumber daya komputasi yang memadai.

Data Beragam/Noisy: Ketika dataset Anda mengandung banyak fitur (dimensi tinggi), data kategorikal, atau memiliki banyak *noise* atau *outlier*.

Tidak Perlu Normalisasi: Ketika Anda ingin menghindari proses scaling/normalisasi data karena RF tidak sensitif terhadap skala fitur.

Memerlukan Feature Ranking: Ketika Anda perlu mengetahui fitur mana yang paling penting dalam memprediksi target.

⚠ Kapan Harus Menghindari

Interpretasi Mutlak Penting: Ketika model harus bisa dijelaskan secara sederhana ke stakeholder non-teknis (misalnya, di industri finansial atau medis).

Keterbatasan Sumber Daya: Ketika Anda bekerja pada sistem dengan RAM atau CPU yang sangat terbatas, terutama untuk data yang sangat besar.

Kebutuhan Prediksi Real-Time: Untuk aplikasi yang membutuhkan kecepatan prediksi sangat tinggi (latency rendah), karena RF harus menjalankan ratusan prediksi per *instance*.

Data Sangat Langka (Sparse): Untuk data teks atau data yang sebagian besar berisi nol (0), model linier atau algoritma lain mungkin lebih efisien.

Terima Kasih