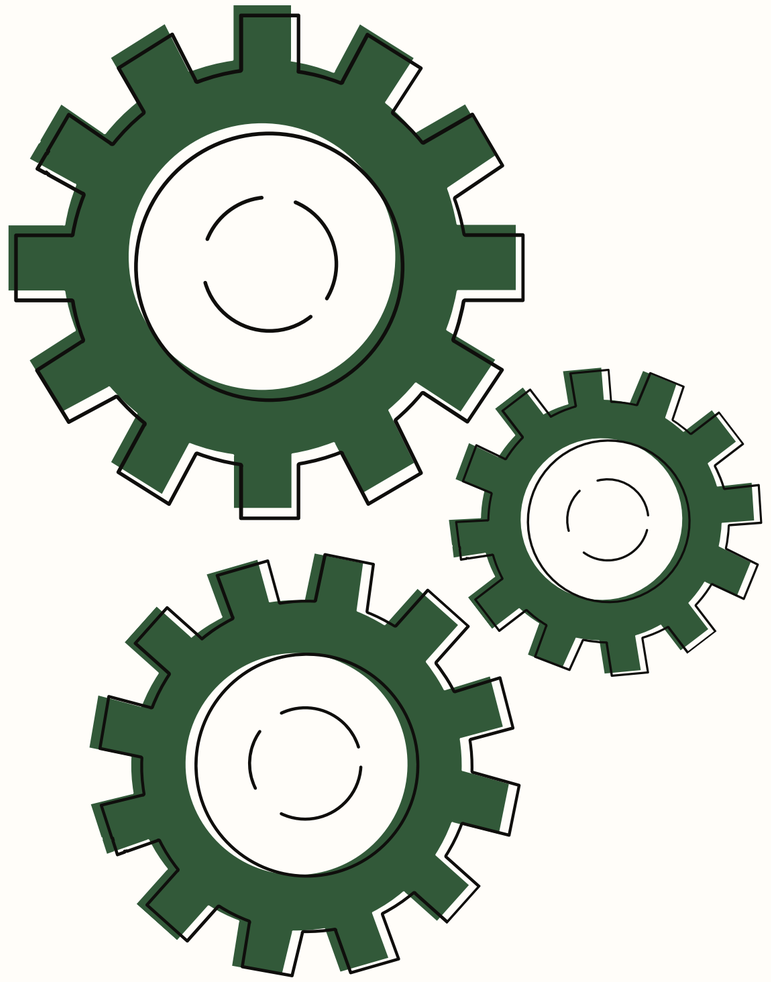
Decision Tree Learning

Alwi Sihabudin Awara (234311006)
Enjang Margha Sasana (234311013)
Rengga Adhiva Fajar P (234311023)

Decision Tree

Decision Tree merupakan sebuah algoritma Supervised Learning yang dapat digunakan untuk klasifikasi maupun regresi. Algoritma ini bekerja dengan memecah data menjadi himpunan bagian yang lebih kecil, membentuk struktur seperti pohon dengan keputusan di setiap cabangnya.



Konsep Dasar

Konsep dasar pada Decision Tree umumnya ada 4 komponen utama yang menyusun strukturnya:

- Root Node (Akar): Node paling atas yang mewakili seluruh populasi atau sampel data. Ini adalah titik awal keputusan.
- Internal Node (Cabang): Node percabangan yang mewakili fitur atau atribut yang sedang diuji (misalnya: mean radius > 14.5?).
- Leaf Node (Daun): Node akhir yang mewakili hasil keputusan atau label kelas (misalnya: jinak atau Ganas).
- Splitting: Proses membagi node menjadi dua atau lebih sub-node berdasarkan kondisi tertentu.

Project Timeline

Rumus Perhitungan (Enthropy)
Decision Tree menggunakan Impurity Measure untuk menentukan pemisahan terbaik. Dua rumus yang paling umum adalah Entropy.

Rumus Enthropy : $$H(S) = -\sum p_i \log_2(p_i)$$

Penjelasan Singkat:

- *H(S)* : Nilai Enthropy (ketidakmurnian) dari set data *S*.
- *pi* : Probabilitas dari kelas *i* dalam node tersebut.
- Algoritma akan mencari pemisahan (split) yang menghasilkan penurunan Enthropy terbesar (Information Gain tertinggi), agar data di node anak (child node) semakin homogen (murni).

Penejelasan Rumus

$$H(S) = -\sum p_i \log_2(p_i)$$

- H(S) adalah Entropy dari suatu himpunan data S.
- ∑ (sigma / penjumlahan) adalah penjumlahan semua
- pi adalah proporsi (probabilitas) data yang termasuk pada kelas ke-i.
- log2(pi) menunjukkan tingkat ketidakpastian atau informasi dari setiap kelas.
- Tanda negatif (−) memastikan nilai entropy selalu positif.
- Entropy tinggi berarti data sangat tercampur dan tidak murni.
- Entropy rendah (mendekati 0) menunjukkan data murni, biasanya hanya terdiri dari satu kelas.

Project Timeline

Rumus Perhitungan (Gini)
Decision Tree juga dapat menggunakan Gini Index sebagai ukuran impurity, terutama pada algoritma CART.

Rumus Gini:

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

Penjelasan Singkat:

- Gini(S) : Nilai Gini (ketidakmurnian) dari set data S.
- pi : Probabilitas dari kelas ke-i dalam node tersebut.
- Semakin kecil nilai Gini, maka node semakin murni.
- Algoritma akan mencari pemisahan (split) yang menghasilkan penurunan Gini terbesar, sehingga node anak (child node) menjadi lebih homogen.

Penejelasan Rumus

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

- Gi = nilai Gini pada node ke-i.
- pi,k = proporsi data kelas ke-k di node i.
- p2i,k = probabilitas kelas ke-k yang dikuadratkan.
- Sigma menjumlahkan seluruh kelas dalam node.

Project Timeline

Rumus Perhitungan (Gain Ratio)

Gain Ratio digunakan pada algoritma C4.5 untuk mengatasi kelemahan Information Gain yang bias terhadap fitur dengan banyak nilai unik.

Rumus Gain Ratio:

$$GainRatio(S, A) = \frac{Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)}{- \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log_2 \left(\frac{|S_v|}{|S|} \right)}$$

Penjelasan Singkat:

- Entropy(S) : Tingkat ketidakteraturan (impurity) pada seluruh data S.
- Values(A) : Semua nilai unik atribut A (misal: “Tinggi”, “Rendah”, atau “Ya”, “Tidak”).
- S_v : Subset data S yang memiliki nilai atribut A = v.
- $|S_v| / |S|$: Proporsi jumlah data pada subset S_v terhadap total data S.
- Entropy(S_v) : Entropy pada subset S_v.
- Log base 2 karena satuan informasi adalah "bit"

Penejelasan Rumus

$$GainRatio(S, A) = \frac{Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)}{- \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log_2 \left(\frac{|S_v|}{|S|} \right)}$$

- Gain(S, A) = Penurunan entropy pada dataset S setelah dilakukan split menggunakan atribut A.
- Entropy(S) = Tingkat ketidakmurnian (impurity) sebelum dilakukan split.
- Entropy(Sv) = Tingkat ketidakmurnian pada subset hasil split ke-v dari atribut A.
- | Sv | / | S | = Proporsi jumlah data pada subset ke-v terhadap jumlah data total.
- SplitInfo(A) = Ukuran seberapa besar atribut A membagi data ke dalam beberapa subset.
- $\frac{|S_v|}{|S|} \log_2 \left(\frac{|S_v|}{|S|} \right)$ = Informasi yang menggambarkan "sebaran" data ke dalam setiap subset v.
- GainRatio(S, A) = Nilai akhir yang menunjukkan kualitas suatu atribut A sebagai pemisah; dihitung dengan membagi Information Gain dengan Split Information.
- Sigma (Σ) = Menjumlahkan seluruh subset hasil pemisahan atribut A.

Hyperparameter

Hyperparameter Decision Tree, parameter yang mengatur proses belajar agar tidak overfitting (terlalu menghafal data):

1. Max Depth: Menentukan kedalaman maksimum pohon.
    - Nilai besar : Pohon sangat detail, risiko overfitting.
    - Nilai kecil : Pohon sederhana, risiko underfitting.
2. Min Samples Split: Jumlah minimum sampel data yang diperlukan untuk membagi node internal.
3. Min Samples Leaf: Jumlah minimum sampel data yang harus ada di node daun (leaf).
4. Criterion: Metode untuk mengukur kualitas split (pilihannya: "Gini" atau "Entropy").

Hypothesis Fungction

- Hipotesis Nol (H_0):
  - Decision Tree tidak mampu memberikan prediksi yang signifikan berdasarkan atribut-atribut pada dataset.

- Hipotesis Alternatif (H_1):
  - Decision Tree dapat memprediksi kelas dengan tingkat akurasi yang signifikan berdasarkan atribut-atribut pada dataset.

- Dasar Penyusunan Hipotesis:
  - Decision Tree efektif digunakan pada data medis karena mampu memetakan pola dari fitur-fitur numerik.
  - Dataset Breast Cancer memiliki fitur yang relevan secara klinis (radius, area, smoothness, dll) sehingga diharapkan model dapat membuat pemisahan (split) yang tepat.
  - Penggunaan impurity (Gini/Entropy) dan hyperparameter (max depth, min samples) mendukung model agar menghasilkan prediksi yang akurat.

STUDI KASUS

Cost Function

Cost Function dalam Decision Tree adalah meminimalkan Impurity (ketidakmurnian).

- *Gi*: Nilai Gini Impurity untuk node i.
  - Jika *G=0* : Node Murni (semua data adalah satu kelas, misal semua "Jinak").
  - Jika *G>0* : Node Tidak Murni (data bercampur antara "Jinak" dan "Ganas").
- *pi,k* : Rasio sampel kelas *k* di dalam node *i*.
  - Contoh: Jika di node ada 100 pasien, 70 Jinak dan 30 Ganas.
  - Maka *pjinak* = 0.7 dan *pganas* = 0.3.

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

Penejelasan Rumus

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

- Gi = nilai Gini pada node ke-i.
- pi,k = proporsi data kelas ke-k di node i.
- p2i,k = probabilitas kelas ke-k yang dikuadratkan.
- Sigma menjumlahkan seluruh kelas dalam node.
- Nilai Gini mendekati 0 → node murni (satu kelas).
- Nilai Gini mendekati 0.5 → node sangat campur.
- Decision Tree memilih split yang meminimalkan Gini, agar node anak lebih homogen.

Bagaimana Dataset Diterapkan

Dataset Breast Cancer (Wisconsin)

Dataset ini digunakan untuk melatih Decision Tree membedakan sel tumor.

- Features (X): Terdapat 30 fitur pengukuran sel[2].
  - Contoh: Radius, Texture, Area, Smoothness.
- Target (Y): Label diagnosa akhir.
  - 0 = Ganas (Malignant) - Kanker.
  - 1 = Jinak (Benign) - Aman.
- Jumlah Data: 569 Pasien total.

Hyperparameter Decision Tree

Konfigurasi Model (Hyperparameter) Dalam kode ini, kita mengatur parameter agar pohon tidak terlalu rumit (Overfitting):

1. Criterion = 'Gini'
    - Metode untuk mengukur ketidakmurnian (impurity) data. Mirip dengan bagaimana Q-Learning memilih Action terbaik, Gini memilih pemisahan (split) yang paling efektif membersihkan data.
2. Max Depth = 3
    - Apa itu? Membatasi kedalaman pohon maksimal 3 tingkat ke bawah.
    - Tujuannya? Agar model tetap sederhana dan bisa memprediksi data baru dengan baik, tidak hanya "menghafal" data latihan. (Konsep ini mirip dengan Epsilon Decay untuk menyeimbangkan eksplorasi).

Training & Testing Split

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Proses Pembelajaran (Training) Agar evaluasi fair dan tidak curang:

- Train Set (80%): 455 Pasien digunakan untuk belajar pola. Komputer mencari aturan "If-Then" terbaik dari data ini.
- Test Set (20%): 114 Pasien disembunyikan untuk ujian. Model dilarang melihat kunci jawaban data ini saat belajar.

Hasil Evaluasi (Accuracy)

```
# Sekarang kita uji dengan 114 pasien sisa (20% data) yang belum pernah dilihat model
prediksi = model.predict(X_test)

# Hitung Akurasi: Seberapa pintar model kita?
akurasi = accuracy_score(y_test, prediksi)

print(f"Akurasi Model: {akurasi * 100:.2f}%")

# Contoh Prediksi Satu Pasien (Ambil data pertama dari test set)
contoh_pasien = X_test[0].reshape(1, -1)
hasil = model.predict(contoh_pasien)
diagnosis = "JINAK (Aman)" if hasil[0] == 1 else "GANAS (Kanker)"

print(f"\nContoh Diagnosa Pasien X: {diagnosis}")
```

Evaluasi Akurasi Seberapa pintar model kita setelah belajar?

- Metrik: Accuracy Score.
- Hasil: *Akurasi Model: XX.XX%* (Akan muncul saat kode dijalankan, biasanya sekitar 93-96% untuk dataset ini).
- Penjelasan: Ini adalah persentase seberapa sering model menebak diagnosa (Jinak/Ganas) dengan benar pada pasien yang belum pernah dilihat sebelumnya.

Prediksi Pasien (Policy)

Contoh Diagnosa (Real-time Prediction)
Setelah model terbentuk, kita menggunakannya untuk pasien baru (seperti Policy menentukan aksi terbaik):

- Input: Data pengukuran sel pasien X (Radius, Texture, dll).
- Proses: Data dijatuhkan ke dalam pohon, melewati pertanyaan-pertanyaan cabang.
- Output:
  - Jika hasil = 1 - JINAK (Aman).
  - Jika hasil = 0 - GANAS (Kanker).

```
Akurasi Model: 93.86%

Contoh Diagnosa Pasien X: JINAK (Aman)
```

RIMBERIO

Thank you!