Nama : Akmal Zuhdy Prasetya
NIM : H071191035
Kelas : Machine Learning

Resume Pertemuan 4

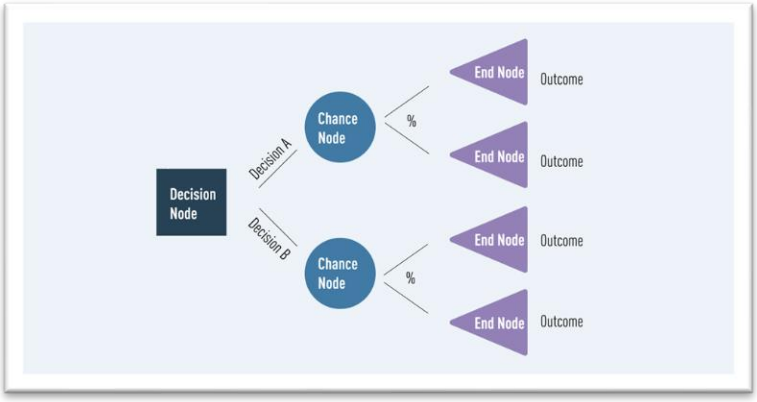
A. Decision Tree

Dalam bentuknya yang paling sederhana, pohon keputusan adalah jenis diagram alur yang menunjukkan jalur yang jelas menuju suatu keputusan. Dalam hal analitik data, ini adalah jenis algoritma yang menyertakan pernyataan “kontrol" bersyarat untuk mengklasifikasikan data. Sebuah pohon keputusan dimulai pada satu titik/simpul (node) yang kemudian bercabang (branch) dalam dua arah atau lebih. Setiap cabang menawarkan kemungkinan hasil yang berbeda, menggabungkan berbagai keputusan dan peristiwa kebetulan sampai hasil akhir tercapai. Ketika ditampilkan secara visual, penampilannya akan terlihat seperti pohon, dengan demikian ia disebut dengan Decision Tree.

Pohon keputusan sangat berguna untuk analisis data dan machine learning karena ia memecah data yang kompleks menjadi bagian-bagian yang lebih mudah dikelola. Ia sering digunakan di bidang ini untuk analisis prediksi, klasifikasi data, dan regresi. Pohon keputusan dapat menangani data yang kompleks, yang merupakan bagian dari apa yang membuatnya berguna. Namun, ini tidak berarti bahwa mereka sulit untuk dipahami. Pada intinya, semua pohon keputusan pada akhirnya hanya terdiri dari tiga bagian utama, yaitu:

- Decision nodes: Mewakili keputusan (biasanya ditunjukkan dengan kotak)
- Chance nodes: Mewakili probabilitas atau ketidakpastian (biasanya dilambangkan dengan lingkaran)
- End nodes: Mewakili hasil (biasanya ditunjukkan dengan segitiga)

Menghubungkan node yang berbeda ini adalah apa yang kita sebut sebagai “cabang”. Node dan cabang dapat digunakan berulang kali dalam sejumlah kombinasi untuk membuat pohon dengan berbagai kompleksitas. Berikut adalah contoh bagaimana bentuk dari bagian-bagian ini sebelum terdapat data yang kita masukkan.



Untungnya, banyak terminology di dalam pohon keputusan yang mengikuti analogi dari sebuag pohon itu sendiri, yang membuatnya lebih mudah untuk diingat. Beberapa istilah lain yang mungkin ditemui dalam decision tree adalah sebagai berikut.

1. Root Nodes (Simpul Akar)

   Dalam diagram di atas, simpul keputusan (Decision Node) yang berwarna biru adalah apa yang kita sebut sebagai “simpul akar”. Simpul ini selalu merupakan simpul pertama di dalam sebuah pohon keputusan. Ia merupakan simpul dari mana semua keputusan, peluang, dan simpul akhir lainnya melakukan percabangan.

2. Leaf Nodes (Simpul Daun)

   Dalam diagram di atas, simpul ujung (End Node) yang berwarna ungu adalah apa yang kita sebut sebagai “simpul daun”. Simpul ini menunjukkan akhir dari sebuah jalur keputusan atau hasil keputusan. Kita selalu dapat mengidentifikasi simpul daun karena simpul ini tidak membelah atau bercabang, seperti halnya dengan daun pada umumnya.

3. Internal Nodes (Simpul Dalam)

   Antara simpul akar dan simpul daun, kita dapat memiliki beberapa simpul dalam atau internal. Simpul ini dapat mencakup simpul keputusan dan simpul probabilitas. Sangat mudah untuk mengidentifikasi simpul internal karena simpulnya masing-masing memiliki cabang sendiri dan simpul ini juga terhubung ke node sebelum dan setelahnya.

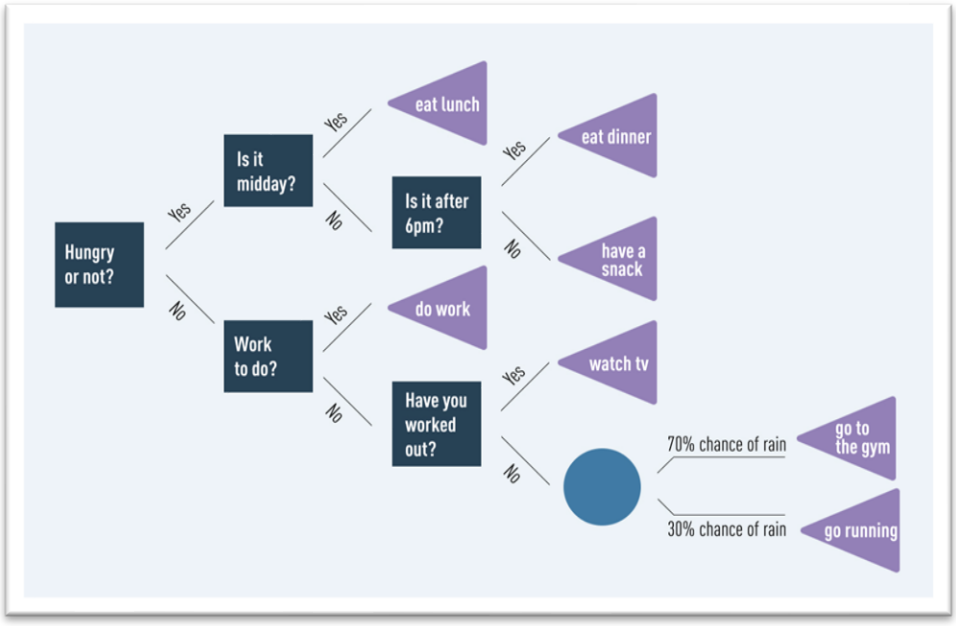
4. Splitting (Percabangan)

   Percabangan atau pemisahan adalah apa yang kita sebut ketika setiap node membagi menjadi dua atau lebih sub-node. Sub-node ini dapat berupa node internal lain, atau mereka dapat mengarah pada sebuah atau beberapa end node.

5. Pruning (Pemangkasan)

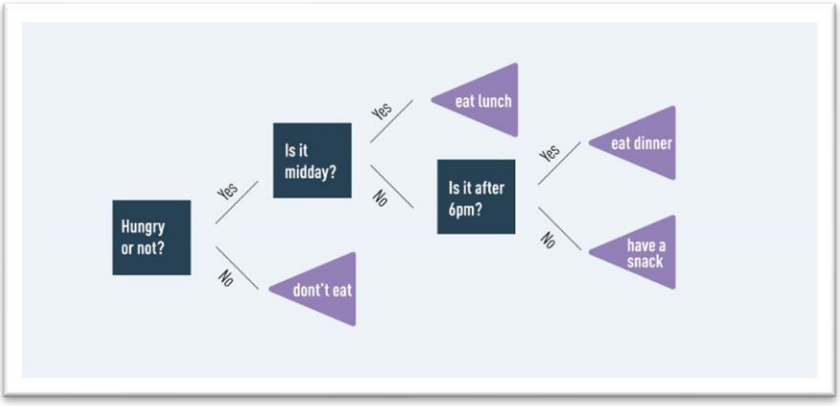
   Terkadang pohon keputusan dapat tumbuh cukup kompleks. Dalam kasus ini, ia dapat memberikan terlalu banyak bobot pada data yang tidak relevan. Untuk menghindari masalah ini, kita dapat menghapus node tertentu menggunakan proses yang dikenal sebagai “pruning”. Pemangkasan seperti namanya merupakan proses pemotongan pada pohon keputusan ketika pohon tersebut menumbuhkan sebuah cabang yang tidak relevan atau tidak kita butuhkan.

Setelah mengetahui fundamental dan beberapa istilah yang terdapat di dalam pohon keputusan, di bawah ini adalah salah satu contoh tentang bagaimana bentuk dari pohon keputusan yang sesunggugnya.



Dalam diagram ini, pilihan yang berbeda ditata dengan cara visual yang jelas. Decision node berwarna biru gelap, chance node berwarna biru muda, dan end node berwarna ungu. Dapat dilihat bahwa diagram ini sangat mudah bagi siapa saja untuk dipahami.

Namun jangan lupa, tujuan kita adalah mengklasifikasikan apa yang harus dilakukan jika lapar. Dengan menyertakan opsi untuk apa yang harus dilakukan jika keputusannya lapar atau tidak lapar, kita telah memperumit pohon keputusan ini. Menata pohon keputusan dengan cara ini adalah sebuah masalah umum, terutama ketika berhadapan dengan jumlah data yang besar. Hal ini sering menghasilkan algoritma yang dapat mengekstrak makna dari informasi yang tidak relevan. Peristiwa ini dikenal sebagai overfitting. Salah satu opsi untuk memperbaiki overfitting adalah dengan melakukan Pruning terhadap pohon keputusan sehingga menjadi seperti berikut.



Seperti yang kita lihat, fokus dari pohon keputusan yang sekarang jauh lebih jelas. Dengan menghapus informasi yang tidak relevan (yaitu apa yang harus dilakukan jika kita tidak lapar), kita akan lebih fokus pada tujuan yang kita tuju.

B. Attribute Selection

Dalam pohon keputusan tantangan utama kita adalah untuk mengidentifikasi atribut untuk root node pada setiap tingkat atau level di dalam pohon keputusan. Proses ini dikenal sebagai pemilihan atribut. Pohon keputusan memiliki dua ukuran pemilihan atribut yang umum, yaitu Information Gain dan Gini Index.

1. Information Gain

Information Gain mengukur seberapa banyak informasi yang diberikan oleh fitur individual tentang kelasnya. Ia bertindak sebagai kunci utama untuk membangun pohon keputusan. Atribut dengan information gain tertinggi akan dipecah terlebih dahulu. Jadi, pohon keputusan selalu memaksimalkan information gain ini. Ketika kita menggunakan sebuah node untuk mempartisi instance menjadi subset yang lebih kecil, maka entropi berubah.

Entropi itu sendiri merupakan ukuran ketidakpastian atau ketidakmurnian variabel acak. Entropi memutuskan bagaimana sebuah pohon keputusan membagi data menjadi himpunan bagian. Adapun cara menghitung entropi adalah sebagai berikut:

$$Entropy(S) = -\sum_{i=1}^{k} p_i \log_2 p_i \text{ , dengan } p_i = \frac{|S_i|}{|S|}$$

Dengan keterangan:

S = Himpunan Kasus (Dataset)
k = Jumlah partisi dari S

Information Gain didasarkan pada penurunan entropi setelah kumpulan data dipecah menjadi atribut. Membangun pohon keputusan adalah tentang bagaimana kita menemukan atribut yang mengembalikan nilai Information Gain tertinggi (yaitu cabang yang paling homogen). Cara menghitungnya adalah sebagai berikut:

$$Info(S) = Entropy(S) \quad Info_A(S) = \sum_{i=1}^{k} \frac{|S_i|}{|S|} Info(S_i)$$

$$Gain(A) = Info(S) - Info_A(S)$$

Dimana $Info_A(S)$ merupakan informasi yang diharapkan akan diperlukan untuk menghasikan tuple dari set S berdasarkan partisi dari A. Atribut yang memiliki nilai Information Gain tertinggi akan dipilih menjadi atribut untuk dilakukan splitting atau percabangan nantinya.

2. Gini Index

Indeks Gini adalah metrik yang memutuskan seberapa sering elemen yang dipilih secara acak akan salah diidentifikasi. Hal ini dengan jelas menyatakan bahwa atribut dengan indeks Gini yang rendah akan diberikan preferensi pertama. Seperti sifat entropi, indeks Gini bervariasi antara nilai 0 dan 1, di mana 0 menyatakan kemurnian klasifikasi, yaitu semua elemen milik kelas tertentu atau hanya satu kelas yang ada di sana. Dan 1 menunjukkan distribusi acak elemen di berbagai kelas. Nilai 0,5 Gini Index menunjukkan distribusi elemen yang merata pada beberapa kelas.

Indeks Gini ditentukan dengan mengurangi jumlah kuadrat dari probabilitas setiap kelas mulai dari satu, secara matematis, Indeks Gini dapat dinyatakan sebagai:

$$Gini(S) = 1 - \sum_{i=1}^{k} p_i^2$$

Dimana p_i menunjukkan probabilitas suatu elemen yang diklasifikasikan untuk kelas yang berbeda. Atribut yang memiliki nilai indeks Gini terendah akan dipilih menjadi atribut untuk dilakukan splitting atau percabangan nantinya.

3. Gain Ratio

Gain Ratio atau Uncertainty Coefficient digunakan untuk menormalkan Information Gain suatu atribut terhadap seberapa banyak entropi yang dimiliki atribut tersebut. Gain ratio adalah modifikasi dari Information Gain yang mengurangi biasnya. Gain ration mengatasi masalah Information Gain dengan memperhitungkan jumlah cabang yang akan dihasilkan sebelum melakukan splitting atau pemangkasan. Ia mengoreksi Information Gain dengan memperhitungkan informasi intrinsik dari proses pemangkasan. Berikut adalah cara menghitung Gain Ratio.

$$SplitInfo_A(S) = Entropy(S)$$

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(S)}$$

Atribut dengan nilai Gain Ratio tertinggi akan dipilih menjadi atribut untuk dilakukan splitting atau percabangan nantinya.

C. ID3

ID3 adalah singkatan dari Iterative Dichotomiser 3, dinamai demikian karena algoritmanya dilakukan secara berulang kali (iterative) membagi (dichotomizes) fitur menjadi dua atau lebih grup pada setiap langkahnya.

Diciptakan oleh Ross Quinlan, ID3 menggunakan pendekatan top-down greedy untuk membangun pohon keputusan. Dengan kata sederhana, pendekatan top-down berarti bahwa kita

mulai membangun pohon dari atas dan pendekatan greedy berarti bahwa pada setiap iterasi kita memilih atribut terbaik saat itu untuk membuat node. Umumnya ID3 hanya digunakan untuk masalah klasifikasi dengan fitur nominal saja.

Algoritma ini menggunakan Information Gain untuk memutuskan atribut mana yang akan digunakan untuk mengklasifikasikan subset data saat iterasinya. Untuk setiap tingkat pohon, Information Gain dihitung untuk data yang tersisa secara rekursif.

Berikut merupakan tahapan konstruksi pohon keputusan menggunakan algoritma ID3:

1. Hitung Information Gain dari setiap fitur atau atribut.
2. Karena diketahui bahwa semua baris tidak termasuk dalam kelas yang sama, pisahkan dataset S menjadi subset menggunakan atribut yang memiliki Information Gain tertinggi.
3. Buat simpul pohon keputusan menggunakan atribut dengan Information Gain tertinggi.
4. Jika semua baris termasuk dalam kelas yang sama, buat simpul saat ini sebagai simpul daun dengan kelas sebagai labelnya.
5. Ulangi untuk semua atribut yang tersisa sampai kita kehabisan atribut atau pohon keputusan sudah memiliki semua simpul daunnya.

Berikut adalah contoh implementasi dari algoritma ID3 pada dataset S mengenai infeksi COVID-19. Berikut adalah datasetnya:

ID	Fever	Cough	Breathing Issues	Infected
1	No	No	No	No
2	Yes	Yes	Yes	Yes
3	Yes	Yes	No	No
4	Yes	No	Yes	Yes
5	Yes	Yes	Yes	Yes
6	No	Yes	No	No
7	Yes	No	Yes	Yes
8	Yes	No	Yes	Yes
9	No	Yes	Yes	Yes
10	Yes	Yes	No	Yes
11	No	Yes	No	No
12	No	Yes	Yes	Yes
13	No	Yes	Yes	No
14	Yes	Yes	No	No

Kolom-kolom yang digunakan untuk membuat simpul keputusan adalah “Demam”, “Batuk”, dan “Masalah Pernapasan” disebut sebagai kolom atribut atau atribut dan kolom yang digunakan untuk simpul daun yaitu “Terinfeksi” disebut sebagai kolom target atau kelas.

Implementasi algoritma pada dataset:

Seperti yang dinyatakan di bagian sebelumnya, langkah pertama adalah menemukan atribut terbaik, yaitu atribut yang memiliki Information Gain tertinggi. Kita akan menghitung Information

Gain untuk masing-masing atribut sekarang, tetapi untuk itu pertama-tama kita perlu menghitung entropi S. Dari total 14 baris di dataset S, ada 8 baris dengan nilai target YES dan 6 baris dengan nilai target NO. Entropi S dihitung sebagai:

$$Entropy(S) = -\left(\frac{8}{14}\right) \cdot \log_2\left(\frac{8}{14}\right) - \left(\frac{6}{14}\right) \cdot \log_2\left(\frac{6}{14}\right) = 0.99$$

Sekarang kita akan menghitung Information Gain untuk setiap atribut yang ada:

Pada atribut “Fever” ini terdapat 8 baris bernilai YES dan 6 baris bernilai NO.

Seperti yang ditunjukkan di bawah ini, dalam 8 baris dengan YES untuk “Fever”, ada 6 baris yang memiliki nilai target YES dan 2 baris memiliki nilai target NO.

Fever	Cough	Breathing Issues	Infected
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	Yes	Yes
Yes	Yes	Yes	Yes
Yes	No	Yes	Yes
Yes	No	Yes	Yes
Yes	Yes	No	Yes
Yes	Yes	No	No

Seperti yang ditunjukkan di bawah ini, dalam 6 baris dengan NO, ada 2 baris yang memiliki nilai target YES dan 4 baris memiliki nilai target NO.

Fever	Cough	Breathing Issues	Infected
No	No	No	No
No	Yes	No	No
No	Yes	Yes	Yes
No	Yes	No	No
No	Yes	Yes	Yes
No	Yes	Yes	No

Berdasarkan fakta di atas, maka perhitungan Information Gain untuk atribut “Fever” adalah sebagai berikut:

$$|S| = 14$$

$$Untuk\ v = YES, |S_v| = 8, maka\ Entropy(S_v) = -\left(\frac{6}{8}\right) \cdot \log_2\left(\frac{6}{8}\right) - \left(\frac{2}{8}\right) \cdot \log_2\left(\frac{2}{8}\right) = 0.81$$

$$Untuk\ v = NO, |S_v| = 6, maka\ Entropy(S_v) = -\left(\frac{2}{6}\right) \cdot \log_2\left(\frac{2}{6}\right) - \left(\frac{4}{6}\right) \cdot \log_2\left(\frac{4}{6}\right) = 0.91$$

$$InformationGain(S, Fever) = Entropy(S) - \left(\frac{|S_{YES}|}{|S|}\right) \cdot Entropy(S_{YES}) - \left(\frac{|S_{NO}|}{|S|}\right) \cdot Entropy(S_{NO})$$

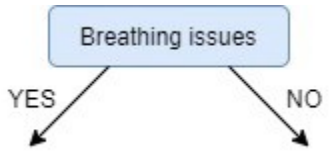
$$InformationGain(S, Fever) = 0.99 - \left(\frac{8}{14}\right) \cdot 0.81 - \left(\frac{6}{14}\right) \cdot 0.91 = 0.13$$

Selanjutnya, kita menghitung nilai Information Gain untuk atribut “Cough” dan “Breathing Issues”. Dengan menggunakan langkah yang sama diperoleh hasil:

$$InformationGain(S, Cough) = 0.04$$

$$InformationGain(S, BreathingIssues) = 0.40$$

Karena atribut “Breathing Issues” memiliki nilai Information Gain tertinggi, atribut inilah yang akan digunakan untuk membuat root node dari pohon keputusan. Oleh karena itu, setelah langkah awal ini, pohon kita terlihat seperti ini:



Selanjutnya, dari sisa dua atribut yang tidak digunakan, yaitu “Fever” dan “Cough, kita akan memutuskan mana yang terbaik untuk cabang kiri dari root node. Karena cabang kiri “Breathing Issues” menunjukkan YES, kita akan bekerja dengan subset dari data asli yaitu kumpulan baris yang memiliki YES sebagai nilai di kolom “Breathing Issues”. Ditemukan 8 baris yang ditunjukkan seperti di bawah ini:

Fever	Cough	Breathing Issues	Infected
Yes	Yes	Yes	Yes
Yes	No	Yes	Yes
Yes	Yes	Yes	Yes
Yes	No	Yes	Yes
Yes	No	Yes	Yes
No	Yes	Yes	Yes
No	Yes	Yes	Yes
No	Yes	Yes	No

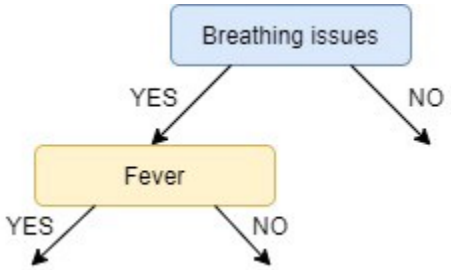
Selanjutnya kita hitung Information Gain untuk atribut “Fever” dan “Cough” menggunakan subset S_{BY} (Set Breathing Issues Yes) yang ditunjukkan pada tabel di atas. Dengan menggunakan langkah yang sama kita peroleh:

$$InformationGain(S_{BY}, Fever) = 0.20$$

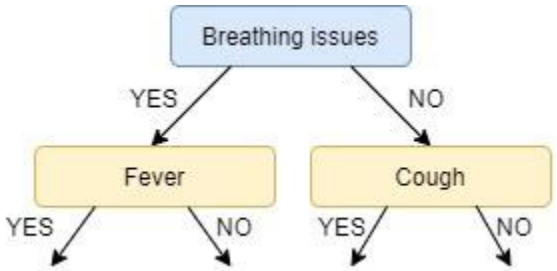
$$InformationGain(S_{BY}, Cough) = 0.09$$

Information Gain “Fever”lebih besar daripada “Cough”, jadi kita akan memilih “Fever” sebagai cabang kiri dari root node “Breathing Issues”:

Maka pohon keputusan kita sekarang akan terlihat seperti ini:



Selanjutnya, kita akan mencari atribut dengan Information Gain tertinggi untuk cabang kanan dari root node. Tetapi, karena hanya tersisa satu atribut yang tidak digunakan, kita tidak punya pilihan lain selain menjadikannya cabang kanan dari root node. Jadi pohon keputusan kita sekarang akan terlihat seperti ini:



Tidak ada lagi atribut yang tidak digunakan, jadi kita berhenti di sini dan menuju ke langkah terakhir yaitu membuat simpul daun atau end node. Untuk simpul daun kiri dari “Fever”, kita dapat lihat subset baris dari set data asli yang memiliki “Breathing Issues” dan “Fever” dimana keduanya bernilai YES. Ditemukan 5 baris yang ditunjukkan seperti di bawah ini:

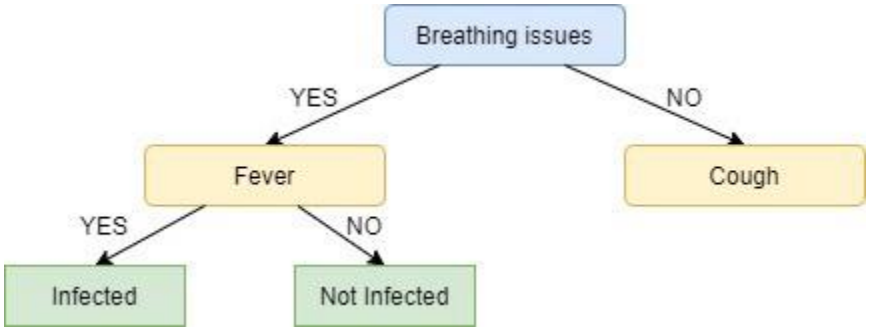
Fever	Cough	Breathing Issues	Infected
Yes	Yes	Yes	Yes
Yes	No	Yes	Yes
Yes	Yes	Yes	Yes
Yes	No	Yes	Yes
Yes	No	Yes	Yes

Karena semua nilai dalam kolom target adalah YES, kita memberi label simpul daun kiri sebagai YES, tetapi untuk membuatnya lebih logis, kita memberi label “Infected”.

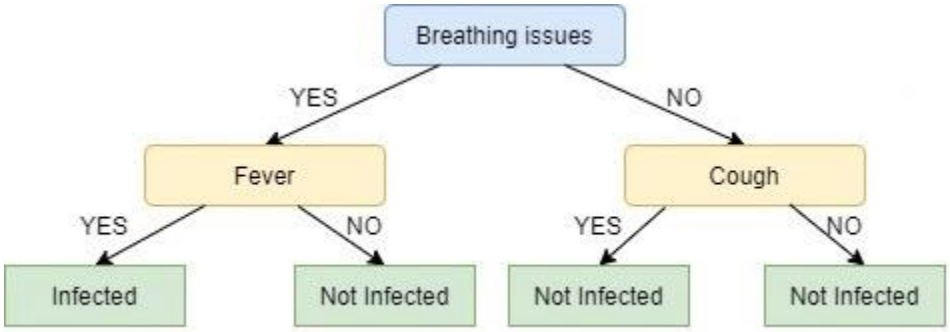
Demikian pula, untuk node kanan “Fever” kita melihat subset baris dari set data asli yang memiliki nilai “Breathing Issues” sebagai YES dan “Fever” sebagai NO. Ditemukan 3 baris yang ditunjukkan seperti di bawah ini:

Fever	Cough	Breathing Issues	Infected
No	Yes	Yes	Yes
No	Yes	Yes	Yes
No	Yes	Yes	No

Di sini tidak semua tetapi sebagian besar nilainya adalah NO, maka NO atau “Not Infected” menjadi simpul daun kanan kita. Pohon keputusan kita sekarang akan terlihat seperti ini:



Kita terus mengulangi proses yang sama untuk simpul “Cough”, namun pada kasus ini kedua daun kiri dan kanan ternyata sama yaitu NO atau “Not Infected” seperti yang ditunjukkan di bawah ini:



Dan inilah pohon keputusan yang telah kita konstruksi menggunakan algoritma ID3 menggunakan dataset infeksi COVID-19 di atas. Hasilnya mungkin terlihat aneh karena simpul kanan dari “Breathing Issues” sama baiknya dengan hanya simpul daun dengan kelas “Not Infected”. Ini adalah salah satu Kekurangan ID3 yaitu algoritmanya tidak melakukan pruning atau pemangkasan.

Dimana telah kita ketahui bahwa pemangkasan adalah mekanisme yang mengurangi ukuran dan kompleksitas pohon keputusan dengan menghapus node yang tidak relevan. Kelemahan lain dari ID3 adalah overfitting atau variansinya tinggi, yaitu mempelajari dataset yang digunakan dengan sangat baik sehingga gagal untuk menggeneralisasi pada data baru.

D. C4.5

C4.5 merupakan salah satu algoritma dalam klasifikasi menggunakan pohon keputusan. Algoritma ini merupakan penerus dari algoritma ID3. Algoritma ini menggunakan baik Information Gain atau Gain Ratio untuk memutuskan atribut pengklasifikasian. Ia merupakan peningkatan langsung dari algoritma ID3 karena dapat menangani nilai atribut yang terus menerus dan hilang.

E. Kelebihan dan Kekurangan Decision Tree

Beberapa kelebihan dari klasifikasi pohon keputusan adalah:

- Komprehensif, pohon keputusan mempertimbangkan setiap hasil yang mungkin dari suatu keputusan dan menelusuri setiap simpul ke kesimpulan yang sesuai.
- Spesifik, pohon keputusan menetapkan nilai spesifik untuk setiap masalah, keputusan, dan hasil. Ini mengurangi ketidakpastian dan ambiguitas dan juga meningkatkan kejelasan.
- Kesederhanaan, pohon keputusan adalah salah satu algoritma yang lebih mudah dan andal karena tidak memiliki rumus atau struktur data yang rumit. Hanya statistik dan matematika sederhana yang diperlukan untuk perhitungan.
- Serbaguna, pohon keputusan dapat dibangun secara manual menggunakan matematika dan juga digunakan dengan program komputer lainnya.

Beberapa kekurangan dari klasifikasi pohon keputusan adalah:

- Pohon keputusan kurang tepat untuk estimasi dan tugas keuangan di mana kita membutuhkan nilai yang sesuai.
- Pohon keputusan merupakan algoritma klasifikasi yang rawan terdapat kesalahan dibandingkan dengan algoritma komputasi lainnya.
- Pohon keputusan mahal secara komputasi. Pada setiap node, kandidat split harus disortir sebelum memastikan yang terbaik. Ada alternatif lain yang diikuti oleh banyak entitas bisnis untuk tugas keuangan karena pohon keputusan terlalu mahal untuk dievaluasi.
- Saat bekerja dengan variabel kontinu, pohon keputusan tidak cocok sebagai solusi terbaik karena cenderung kehilangan informasi saat mengkategorikan variabel.
- Terkadang tidak stabil karena variasi kecil dalam kumpulan data dapat menyebabkan pembentukan pohon baru.

F. Kesimpulan

Sebagai salah satu algoritma yang paling penting dan diawasi, Decision Tree atau Pohon Keputusan memainkan peran penting dalam analisis keputusan dalam kehidupan nyata. Sebagai model prediktif, digunakan di banyak bidang untuk pendekatan split yang membantu dalam mengidentifikasi solusi berdasarkan kondisi yang berbeda baik dengan metode klasifikasi ataupun regresi.