**BAB II**

**LANDASAN TEORI**

1. **Pengertian *Data Mining***

Dalam perkembangannya data mining memiliki banyak definisi yang cukup beragam sehingga data mining dapat menambah ilmu pengetahuan. Berikut ini adalah beberapa definisi data mining pada umumnya :

“Data mining adalah proses yang mempekerjakan satu atau lebih teknik pembelajaran komputer (*machine learning*) untuk menganalisis dan mengekstraksi pengetahuan (*knowledge*) secara otomatis. Data mining berisi pencarian trend atau pola yang diinginkan dalam database besar untuk membantu pengambilan keputusan di waktu yang akan datang. Pola-pola ini dikenali oleh perangkat tertentu yang dapat memberikan suatu analisa data yang berguna dan berwawasan yang kemudian dapat dipelajari dengan lebih teliti, yang mungkin saja menggunakan perangkat pendukung keputusan lainnya” (Fajar Astuti, 2013:3).

“Data mining adalah proses untuk mendapatkan informasi yang berguna dari gudang basis data yang besar. Data mining juga dapat diartikan sebagai pengekstrakan informasi baru yang diambil dari bongkahan data besar yang membantu dalam pengambilan keputusan. Istilah data mining kadang disebut juga *knowledge discovery*” (Tan, 2006).

“Data mining adalah campuran dari statistik, kecerdasan buatan, dan riset basis data” (Gonunescu, 2011).

Jadi dapat disimpulkan bahwa data mining adalah serangkaian proses untuk menggali nilai tambah berupa informasi yang selama ini tidak diketahui secara manual dari suatu basis data. Informasi yang dihasilkan diperoleh dengan cara mengekstraksi dan mengenali pola yang penting atau menarik dari data yang terdapat dalam basis data yang besar. Di samping itu ada istilah lain yang mempunyai makna yang sama dengan *data mining* yaitu *Knowledge Discovery in*

*Database* (*KDD*). Di dalam *data mining* maupun *KDD* bertujuan untuk memanfaatkan data dalam basis data dengan mengolahnya sehingga menghasilkan informasi baru yang berguna.

*Database*

Informasi

Gambar 2.1 Akar Ilmu *Data Mining* (*Prasetyo, 2014)*

Seperti diilustrasikan pada Gambar 2.1, jika diteliti akar keilmuannya, ternyata *data mining* mempunyai empat akar bidang ilmu sebagai berikut:

1. Statistik

Bidang ini merupakan akar paling tua, tanpa ada statistik maka *data mining* mungkin tidak ada. Dengan menggunakan statistik klasik ternyata data yang diolah dapat diringkas dalam apa yang umum dikenal sebagai *exploraty data analysis* (*EDA)*. *EDA* berguna untuk mengidentifikasi hubungan sistematis antar variabel ketika tidak ada cukup informasi alami yang dibawanya.

1. Kecerdasan Buatan atau A*rtifical Intelligence* (*AI*)

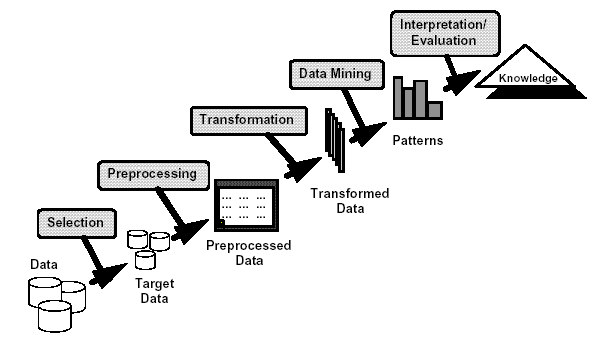
Bidang ilmu ini berbeda dengan statistik. Teorinya dibangun berdasarkan teknik *heuristik* sehingga *AI* berkontribusi terhadap teknik pengolahan informasi berdasarkan pada model penalaran manusia. Salah satu cabang dari *AI*, yaitu pembelajaran mesin atau *machine learning,* merupakan disiplin ilmu yang paling penting yang direpresentasikan dalam pembangun *data mining*.

1. Pengenalan Pola

Sebenarnya *data mining* juga menjadi turunan bidang pengenalan pola, tetapi hanya mengolah data dari basis data. Data yang diambil dari basis data untuk diolah bukan dalam bentuk relasi, melainkan dalam bentuk normal pertama sehingga set data dibentuk menjadi bentuk normal pertama. Akan tetapi *data mining* mempunyai ciri khas yaitu pencarian pola asosiasi dan pola sekuensial.

1. Sistem Basis Data

Akar bidang ilmu yang keempat dari *data mining* yang menyediakan informasi berupa data yang akan digali menggunakan metode-metode yang ada di dalam *data mining*. Tahapan proses dalam penggunaan data mining yang merupakan proses *Knowledge Discovery in Database* (*KDD)* seperti yang terlihat pada Gambar 2.2 dapat diuraikan sebagai berikut:

1. Memahami domain aplikasi untuk mengetahui dan menggali pengetahuan awal serta apa sasaran pengguna.
2. Membuat target data set yang meliputi pemilihan dan fokus pada sub-sub data.
3. Pembersihan dan transformasi data meliputi eliminasi derau, *outliers*, *missing value* serta pemilihan fitur dan reduksi dimensi.
4. Penggunaan algoritma *data mining* yang terdiri dari asosiasi, sekuensial, klasifikasi klasterisasi, dan sebagainya.
5. Interpretasi, evaluasi dan visualisasi pola untuk melihat apakah ada sesuatu yang baru dan menarik dan dilakukan iterasi jika diperlukan.

Gambar 2.2 Proses *Data Mining (Fajar Astuti, 2013)*

Tahap – tahap pada proses *Knowledge Discovery in Database* (*KDD)* sebagai berikut:

1. *Data Selection*

Pemilihan (seleksi) data dari sekumpulan data operasional perlu dilakukan sebelum tahap penggalian informasi dalam *KDD* dimulai. Data hasil seleksi yang akan digunakan untuk proses *data mining*, disimpan dalam suatu berkas, terpisah dari basis data operasional.

1. *Pre-processing/cleaning*

Sebelum proses *data mining* dapat dilaksanakan, perlu dilakukan proses *cleaning* pada data yang menjadi fokus *KDD*. Proses *cleaning* mencakup antara lain membuang duplikasi data, memeriksa data yang inkonsisten, dan memperbaiki kesalahan pada data, seperti kesalahan cetak. Juga dilakukan proses *enrichment*, yaitu proses “memperkaya” data yang sudah ada dengan data atau informasi lain yang relavan dan diperlukan untuk *KDD*, seperti data atau informasi eksternal.

1. *Transformation*

*Coding* adalah proses transformasi pada data yang telah dipilih, sehingga data tersebut sesuai untuk proses *data mining*. Proses *coding* dalam *KDD* merupakan proses kreatif dan sangat tergantung pada jenis atau pola informasi yang akan dicari dalam basis data.

1. *Data mining*

*Data mining* adalah proses mencari pola atau informasi menarik dalam data terpilih dengan menggunakan teknik atau metode tertentu. Teknik, metode, atau algoritma dalam *data mining* sangat bervariasi. Pemilihan metode dan algoritma yang tepat sangat bergantung pada tujuan dan proses *KDD* secara keseluruhan.

1. *Interpretation/evalution*

Pola informasi yang dihasilkan dari proses data mining perlu ditampilkan dalam bentuk yang mudah dimengerti oleh pihak yang berkepentingan. Tahap ini merupakan bagian dari proses *KDD* yang disebut *interpretation*. Tahap ini mencakup pemeriksaan apakah pola atau informasi yang ditemukan bertentangan dengan fakta atau hipotesis yang ada sebelumnya.

**2.1.1 Pekerjaan Dalam *Data Mining***

Pekerjaan yang berkaitan dengan *data mining* dapat dibagi menjadi empat kelompok yaitu :

1. Model Prediksi (*Prediction Modelling*)

Pekerjaan ini berkaitan dengan pembuatan sebuah model yang dapat melakukan pemetaan dari setiap himpunan variabel ke setiap targetnya, kemudian menggunakan model tersebut untuk memberikan nilai target pada himpunan baru yang didapat.

1. Analisis Cluster (*Cluster Analysis*)

Analisis cluster melakukan pengelompokan data kedalam sejumlah kelompok berdasarkan kesamaan karakteristik masing-masing data pada kelompok-kelompok yang ada. Data-data yang masuk dalam batas kesamaan dengan kelompoknya akan bergabung dalam kelompok tersebut, dan akan terpisah dalam kelompok yang berbeda jika keluar dari batas kelompok tersebut.

1. Analisis Asosiasi (*Assosiation Analysis*)

Analisis asosiasi digunakan untuk menemukan pola yang menggambarkan kekuatan hubungan fitur dalam data. Pola yang ditemukan biasanya mempresentasikan bentuk aturan implikasi. Tujuannya adalah untuk menemukan pola yang menarik dengan cara yang efisien.

1. Deteksi Anomali (*Anomaly Detection*)

Deteksi anomali yang baik harus mempunyai laju deteksi yang tinggi dengan jaringan untuk mengetahui pola data yang memasuki jaringan sehingga dapat diketahui dengan adanya penyusupan jika pola kerja data yang datang berbeda.

**2.1.2 Proses *Data Mining***

Secara sistematis, ada tiga langkah utama dalam *data mining* (Gonunescu, 2011).

1. Eksplorasi/pemrosesan awal data

Eksplorasi/pemrosesan awal data terdiri dari pembersihan data, normalisasi data, transformasi data, penanganan data yang salah, reduksi dimensi, pemilihan *subset fitur*, dan sebagainya.

1. Membangun model dan melakukan validasi terhadapnya

Membangun model dan melakukan validasi terhadapnya berarti melakukan analisis sebagai model dan memilih model dengan kinerja prediksi yang terbaik. Dalam langkah ini digunakan metode-metode seperti estimasi, regresi, analisis cluster, deteksi anomali, analisis asosiasi, analisis pola sekuensial, dan sebagainya.

1. Penerapan

Penerapan berarti menerapkan model pada data yang baru untuk menghasilkan perkiraan/prediksi masalah yang akan diteliti.

**2.1.3 Metode *Data Mining***

Pada proses pemecahan masalah dan pencarian pengetahuan baru terdapat beberapa klasifikasi secara umum yaitu:

1. Asosiasi
2. *Clustering*
3. *Classification*
4. Estimasi
5. Prediksi
   1. **Pengertian Algoritma C4.5**

Algoritma C4.5 adalah algoritma klasifikasi dengan teknik pohon keputusan yang terkenal dan disukai karena memiliki kelebihan-kelebihan. Kelebihan ini dapat mengolah data numerik (*kontinyu*) dan *diskret*, dapat menangani nilai atribut yang hilang, menghasilkan aturan-aturan yang mudah diinterpretasikan dan tercepat di antara algoritma-algoritma yang menggunakan memori utama komputer.

Algoritma C4.5 mengkonstruksi pohon keputusan dari data pelatihan, yang berupa kasus-kasus atau record dalam basis data. Setiap kasus berisikan nilai dari atribut-atribut untuk sebuah kelas. Setiap atribut dapat berisi data *diskret* atau *kontinyu* (numerik). C4.5 juga menangani kasus yang tidak memiliki nilai untuk sebuah atau lebih atribut. Akan tetapi, atribut kelas hanya bertipe *diskret* dan tidak boleh kosong.

**2.2.1 Prinsip Kerja Algoritma C4.5**

Prinsip kerja algoritma C4.5 adalah sebagai berikut:

1. Pembuatan Pohon Keputusan

Obyektif dari algoritma pohon keputusan adalah mengkonstruksi struktur data pohon keputusan yang dapat digunakan untuk memprediksi kelas dari sebuah kasus atau *record* baru yang belum memiliki kelas. Algoritma ini memilih pemecahan kasus-kasus yang terbaik dengan menghitung dan membanding *gain ratio,* kemudian pada node-node yang terbentuk di level berikutnya. Demikian seterusnya sampai terbentuk daun-daun.

1. Pemangkasan Pohon dan Keputusan dan Evaluasi (Opsional)

C4.5 dapat menyederhanakan pohon dengan melakukan pemangkasan berdasarkan nilai tingkat kepercayaan (confidence level) bertujuan untuk mengurangi tingkat kesalahan prediksi pada kasus *(record)* baru.

1. Pembuatan Aturan-Aturan dari Pohon Keputusan (Opsional)

Aturan-aturan dalam bentuk *If-then* diturunkan dari pohon keputusan dengan melakukan penelusuran dari akar sampai daun. Setiap node dan syarat percabangannya akan diberikan *If* , sedangkan nilai pada daun akan menjadi ditulis di *then*. Setelah semua aturan dibuat, maka aturan akan disederhanakan.

**2.2.2 Langkah-Langkah Pohon Keputusan Algoritma C4.5**

Langkah-langkah konstruksi pohon keputusan dengan algoritma C4.5 adalah sebagai berikut:

1. Pohon dimulai dengan sebuah simpul pada sampel data pelatihan yaitu dengan membuat simpul akar.
2. Jika semua sampel berada dalam kelas yang sama, maka simpul ini menjadi daun dan dilabeli menjadi kelas. Jika tidak, *gain ratio* akan digunakan untuk memilih atribut *split*, yaitu atribut yang terbaik dalam memisahkan data sampel menjadi kelas-kelas individu.
3. Cabang akan dibuat untuk setiap nilai pada atribut dan data sampel akan dipartisi lagi.
4. Algoritma ini menggunakan proses rekursif untuk membentuk pohon keputusan pada setiap data partisi. Jika sebuah atribut sudah digunakan disebuah simpul, maka atribut tidak digunakan lagi disimpul anak-anaknya.
5. Proses ini berhenti jika dicapai kondisi sebagai berikut:
6. Semua sampel pada simpul berada di dalam satu kelas
7. Tidak ada atribut lainnya yang dapat digunakan untuk mempatisi sampel lebih lanjut.

Algoritma C4.5 diperkenalkan oleh Quinlan (1996) sebagai versi ID3. Dalam ID3, induksi *decision tree* hanya bisa dilakukan pada fitur bertipe kategorikal (nominal atau ordinal), sedangkan tipe numerik (interval atau rasio) tidak dapat digunakan. Perbaikan yang membedakan algoritma C4.5 dari ID3 adalah dapat menangani fitur dengan tipe numerik, melakukan pemotongan (*prunning*) *decision tree*, dan penurunan (*deriving*) *rule set*. Algoritma C4.5 juga menggunakan kriteria *gain* dalam menentukan fitur yang menjadi pemecah *node* pada pohon yang diinduksi.

pria wanita

Gambar 2.3 Syarat Pengujian Fitur Biner (*Prasetyo, 2014)*

Yang menjadi hal penting dalam induksi *decision tree* adalah bagaimana menyatakan syarat pengujian pada *node*. Ada 3 kelompok penting dalam syarat pengujian *node* :

1. Fitur biner

Fitur yang hanya mempunyai dua nilai berbeda disebut dengan fitur biner. Syarat pengujian ketika fitur ini menjadi *node* (akar maupun internal) hanya punya dua pilihan cabang. Contoh pemecahannya pada Gambar 2.3.

1. Fitur bertipe kategorikal

Untuk fitur yang nilainya bertipe kategorikal (nominal atau ordinal) bisa mempunyai beberapa nilai berbeda. Contohnya adalah fitur ‘cuaca’ mempunyai 3 nilai berbeda, dan ini bisa mempunyai banyak kombinasi syarat pengujian pemecahan. Secara umum ada 2, yaitu pemecahan biner (*binary splitting*) dan (*multi splitting*). Kombinasinya seperti Gambar 2.4. Untuk pemecahan yang hanya membolehkan pemecahan biner, seperti algoritma CART, maka akan memberikan kemungkinan jumlah kombinasi pemecahan sebanyak , di mana *k* adalah jumlah nilai berbeda dalam fitur tersebut. Contoh pemecahannya pada Gambar 2.4 dan 2.5.

1. Fitur bertipe numerik

Untuk fitur bertipe numerik, syarat pengujian dalam *node* (akar maupun internal) dinyatakan dengan pengujian perbandingan (A < *v*) atau (A ≥ *v*) dengan hasil biner, atau untuk multi dengan hasil berupa jangkauan nilai dalam bentuk , untuk *i* = 1, 2, ..., *k*. Untuk kasus pemecahan biner, maka algoritma akan memeriksa semua kemungkinan posisi pemecahan *v* dan memilih posisi *v* yang terbaik. Untuk cara multi, maka algoritma harus memeriksa semua kemungkinan jangkauan nilai kontinyu. Contoh pemecahan pada fitur numerik pada Gambar 2.6.

cerah mendung hujan cerah {mendung, hujan}

1. Multi spliting b. Binary splitting

Mendung {cerah, hujan} {cerah, mendung} hujan

1. Binary splitting d. Binary splitting

Gambar 2.4 Syarat Pengujian Fitur Bertipe Nominal (*Prasetyo, 2014)*

S M L XL {S, M} {L, XL}

1. Multi spliting b. Binary splitting

{S} {M, L, XL} {S, L} {M, XL}

1. Binary splitting d. Binary splitting

Gambar 2.5 Syarat Pengujian Fitur Bertipe Ordinal (*Prasetyo, 2014)*

<25 25-34 35-45 >45 ya tidak

1. Multi spliting b. Binary splitting

Gambar 2.6 Syarat Pengujian Fitur Bertipe Numerik (*Prasetyo, 2014)*

Kriteria yang paling banyak digunakan untuk memilih fitur sebagai pemecah dalam algoritma C4.5 adalah *rasio gain*, yang diformulasikan oleh persamaan berikut :

*RasioGain*(s,j) = ............................................... (2.4)

Persamaan (2.4) menyatakan nilai *rasio gain* pada fitur ke-*j.* *SplitInfo(s,j)* didapat dari rumus sebagai berikut :

*SplitInfo(s,j)* =

di mana nilai *Gain(s,j)*  didapatkan dengan rumus :

*Gain (S,J)* *E (S)*

Dimana :

*S*  : Himpunan kasus

*J*  : Atribut

*k* : Jumlah partisi S

*Gain*(S,J) : *Information gain* pada atribut J

*SplitInfo*(S,J) : *Split information* pada atribut J

: proporsi kelas ke-*i*

**2.2.3 Penyelesaian Kasus Prediksi dalam Algoritma C4.5**

Berikut ini adalah uraian langkah-langkah dalam algoritma C4.5 untuk menyelesaikan kasus prediksi “Apakah harus bermain baseball?” dengan menjawab ya atau tidak. Berdasarkan cuaca, suhu, kelembaban, dan angin. Data yang telah ada pada Tabel 2.1 yang akan digunakan untuk membentuk pohon keputusan. Nilai fitur menggunakan tipe kategorikal dan numerik adalah sebagai berikut:

Cuaca = {cerah, mendung, hujan}

Suhu = kontinu

Kelembaban = kontinu

Angin = {pelan, kencang}

Tabel 2.1 Data klasifikasi bermain baseball dengan tipe fitur campuran

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cuaca** | **Suhu** | **Kelembaban** | **Angin** | **Bermain** |
| Cerah | 85 | 85 | Biasa | Tidak |
| Cerah | 80 | 90 | Kencang | Tidak |
| Mendung | 83 | 78 | Biasa | Ya |
| Hujan | 70 | 96 | Biasa | Ya |
| Hujan | 68 | 80 | Biasa | Ya |
| Hujan | 65 | 70 | Kencang | Tidak |
| Mendung | 64 | 65 | Kencang | Ya |
| Cerah | 72 | 95 | Biasa | Tidak |
| Cerah | 69 | 70 | Biasa | Ya |
| Hujan | 75 | 80 | Biasa | Ya |
| Cerah | 75 | 70 | Kencang | Ya |
| Mendung | 72 | 80 | Kencang | Ya |
| Mendung | 81 | 90 | Biasa | Ya |
| Hujan | 71 | 80 | Kencang | Tidak |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:68)*

1. Proses untuk *node* akar

Dimulai dari *node* akar, harus dihitung dulu *entropy* untuk *node* akar (semua data) terhadap komposisi kelas.

*E* =

*E* (s) = ­

=

= 0.9403

Selanjutnya untuk fitur yang bertipe numerik harus ditentukan posisi *v* yang terbaik untuk pemecahan. Dalam contoh ini digunakan pemecahan biner. Hasil uji coba pada fitur ‘suhu’ dengan menghitung nilai *gain* pada tabel 2.2 Nilai *gain* tertinggi didapatkan pada posisi *v* = 70. Oleh karena itu, untuk fitur ‘suhu’ dilakukan diskretisasi pada *v* = 70 ketika menghitung *entropy* dan *gain* pada semua fitur.

Tabel 2.2 Posisi *v* untuk pemecah fitur ‘suhu’ di nodeakar

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Suhu** | **70** | | **75** | | **80** | |
|  | <= | > | <= | > | <= | > |
| Ya | 4 | 5 | 7 | 2 | 7 | 2 |
| Tidak | 1 | 4 | 3 | 2 | 4 | 1 |
| Gain | 0.0453 | | 0.0251 | | 0.0005 | |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:68)*

Hasil uji coba pada fitur ‘kelembaban’ dengan menghitung nilai *gain* disajikan pada tabel 2.3. Nilai *gain* tertinggi didapatkan pada posisi *v* = 80. Oleh karena itu, untuk fitur ‘kelembaban’ dilakukan *diskretisasi* pada *v* = 80 ketika menghitung *entropy* dan *gain* semua fitur.

Tabel 2.3 Posisi *v* untuk pemecahan fitur ‘kelembaban’ di *node* akar

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Kelembaban** | **70** | | **75** | | **80** | | **85** | |
|  | <= | > | <= | > | <= | > | <= | > |
| Ya | 2 | 7 | 3 | 6 | 7 | 2 | 7 | 2 |
| Tidak | 1 | 4 | 1 | 4 | 2 | 3 | 3 | 2 |
| Gain | 0.0005 | | 0.015 | | 0.1022 | | 0.0251 | |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:69)*

Selanjutnya dihitung *entropy* untuk setiap nilai fitur terhadap kelas, kemudian dihitung *gain* untuk setiap fitur. Hasilnya pada Tabel 2.4

Tabel 2.4 Hasil perhitungan entropy dan gain untuk node akar

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Node** | | | **Jml** | **Ya** | **Tidak** | **Entropy** | **Gain** |
| 1 | Total |  | 14 | 9 | 5 | 0.9403 |  |
|  | Cuaca |  |  |  |  |  | 0.2467 |
|  | Cerah | 5 | 2 | 3 | 0.9710 |  |
| Mendung | 4 | 4 | 0 | 0 |  |
| Hujan | 5 | 3 | 2 | 0.9710 |  |
| Suhu |  |  |  |  |  | 0.0453 |
|  | <= 70 | 5 | 4 | 1 | 0.7219 |  |
| > 70 | 9 | 5 | 4 | 0.9911 |  |
| Kelembaban |  |  |  |  |  | 0.1022 |
|  | < = 80 | 9 | 7 | 2 | 0.7642 |  |
| > 80 | 5 | 3 | 2 | 0.971 |  |
| Angin |  |  |  |  |  | 0.0481 |
|  | Pelan | 8 | 6 | 2 | 0.8113 |  |
| Kencang | 6 | 3 | 3 | 1.0000 |  |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:69)*

Hasil yang didapat di Tabel 2.4 menunjukkan bahwa *gain* tertinggi ada fitur ‘cuaca’ sehingga ‘cuaca’ dijadikan sebagai *node* akar. Selanjutnya dihitung posisi *split* untuk fitur ‘cuaca’ dengan menghitung *rasio gain.* Selengkapnya disajikan pada Tabel 2.5.

Hasil perhitungan *rasio gain* posisi *split* untuk opsi satu sebagai berikut:

*SplitInfo (semua,cuaca*) =

=

= )

= 1.5774

*RasioGain(semua, cuaca)* =

=

= 0.16

Dengan cara yang sama, akan didapatkan nilai *rasio gain* untuk opsi yang lain. Hasil di Tabel 2.5 menunjukkan bahwa *rasio gain* tertinggi ada di opsi 4, yaitu *split* {cerah, hujan} dengan {mendung}. Itu artinya cabang untuk akar ada 2, yaitu {cerah, hujan} dan {mendung}.

Tabel 2.5 Perhitungan rasio gain untuk fitur ‘cuaca’

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Node** |  |  | **jml** | **Entropy** | **Gain** | **Rasio Gain** |
| 1 | Total |  | 14 |  | 0.2467 |  |
| **Opsi 1** | Cuaca |  |  | 1.5774 |  | 0.16 |
|  |  | Cerah | 5 |  |  |  |
|  |  | Mendung | 4 |  |  |  |
|  |  | Hujan | 5 |  |  |  |
| **Opsi 2** | Cuaca |  |  | 0.9403 |  | 0.26 |
|  |  | Cerah | 5 |  |  |  |
|  |  | Mendung dan Hujan | 9 |  |  |  |
| **Opsi 3** | Cuaca |  |  | 0.9403 |  | 0.26 |
|  |  | Cerah, Mendung | 9 |  |  |  |
|  |  | Hujan | 5 |  |  |  |
| **Opsi 4** | Cuaca |  |  | 0.8631 |  | 0.29 |
|  |  | Cerah, Hujan | 10 |  |  |  |
|  |  | Mendung | 4 |  |  |  |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:70)*

Hasil pemisahan data menurut *node* akar sebagai berikut :

Tabel 2.6 Pemisahan data menurut node akar

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cuaca** | **Suhu** | **Kelembaban** | **Angin** | **Bermain** |
| Cerah | 85 | 85 | Biasa | Tidak |
| Cerah | 80 | 90 | Kencang | Tidak |
| Hujan | 70 | 96 | Biasa | Ya |
| Hujan | 68 | 80 | Biasa | Ya |
| Hujan | 65 | 70 | Kencang | Tidak |
| Cerah | 72 | 95 | Biasa | Tidak |
| Cerah | 69 | 70 | Biasa | Ya |
| Hujan | 75 | 80 | Biasa | Ya |
| Cerah | 75 | 70 | Kencang | Ya |
| Hujan | 71 | 80 | Kencang | Tidak |
| Mendung | 83 | 78 | Biasa | Ya |
| Mendung | 64 | 65 | Kencang | Ya |
| Mendung | 72 | 90 | Kencang | Ya |
| Mendung | 81 | 75 | Biasa | Ya |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:69-71)*

1. Proses untuk *node* 2

Untuk *node* 2, nilai *entropy* yang didapat adalah 0 (karena semua baris memiliki kelas yang sama). Oleh karena itu dipastikan *node* 2 menjadi daun seperti gambar berikut :

ya

mendung {cerah, hujan}

Gambar 2.7 Hasil Pembentukan Cabang di Akar untuk Kasus “Apakah harus Bermain Baseball?” (*Prasetyo, 2014)*

1. Proses untuk *node* 3

Selanjutnya, di *node* 3 harus dihitung dulu *entropy* untuk sisa data terhadap komposisi kelas dalam *node* 2. Untuk fitur yang bertipe numerik harus ditentukan lagi posis *v* yang terbaik untuk pemecahan. Dalam contoh ini digunakan pemecahan biner. Hasil uji coba pada fitur ‘suhu’ dengan menghitung nilai *gain* pada Tabel 2.7. Nilai *gain* tertinggi didapatkan pada posisi *v* =75 ketika menghitung *entropy* dan *gain* pada semua fitur.

Tabel 2.7 Posisi v untuk pemecahan fitur ‘suhu’ di node 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Suhu** | **70** | | **75** | | **80** | |
|  | <= | > | <= | > | <= | > |
| Ya | 3 | 2 | 5 | 0 | 5 | 0 |
| Tidak | 1 | 4 | 3 | 2 | 4 | 1 |
| Gain | 0.1245 | | 0.2365 | | 0.1080 | |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:72)*

Hasil uji coba fitur ‘kelembaban’ dengan menghitung nilai *gain* pada Tabel 2.8. Nilai *gain* tertinggi didapatkan pada posisi *v* = 80. Oleh karena itu, untuk fitur ‘kelembaban’ dilakukan *diskretasi* pada *v* = 80 ketika menghitung *entropy* dan *gain* pada semua fitur.

Tabel 2.8 posisi v untuk pemecahan fitur ‘kelembaban’ di node 3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Kelembaban** | **70** | | **75** | | **80** | | **85** | |
|  | <= | > | <= | > | <= | > | <= | > |
| Ya | 2 | 3 | 2 | 3 | 4 | 3 | 4 | 1 |
| Tidak | 1 | 4 | 1 | 4 | 2 | 1 | 3 | 2 |
| Gain | 0.0349 | | 0.0349 | | 0.1245 | | 0.0349 | |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:72)*

Selanjutnya dihitung *entropy* untuk setiap nilai fitur terhadap kelas, kemudian dihitung *gain* untuk setiao fitur. Hasilnya pada Tabel 2.9

Tabel 2.9 Hasil perhitungan entropy dan gain untuk node 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Node** |  |  | **jml** | **Ya** | **Tidak** | **Entropy** | **Gain** |
| 3 | Total |  | 10 | 5 | 5 | 1.0000 |  |
|  | Cuaca |  |  |  |  |  | 0.0290 |
|  | Cerah | 5 | 2 | 3 | 0.9710 |  |
|  | Hujan | 5 | 3 | 2 | 0.9710 |  |
|  |  |  |  |  |  |  |
| Suhu |  |  |  |  |  | 0.2365 |
|  | <= 75 | 8 | 5 | 3 | 0.9544 |  |
|  | > 75 | 2 | 0 | 2 | 0 |  |
| Kelembaban |  |  |  |  |  | 0.1245 |
|  | <= 80 | 6 | 4 | 2 | 0.9183 |  |
|  | >80 | 4 | 1 | 3 | 0.8113 |  |
| Angin |  |  |  |  |  | 0.1245 |
|  | Pelan | 6 | 4 | 2 | 0.9183 |  |
|  | Kencang | 4 | 1 | 3 | 0.8113 |  |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:72)*

Hasil yang ditunjukkan pada Tabel 2.9 menunjukkan bahwa gain tertinggi ada di fitur ‘suhu’ sehingga fitur ‘suhu’ syarat kondisi di *node* 3 seperti pada Gambar 2.9. Pemisahan datanya pada Tabel 2.10.

ya

tidak

mendung {cerah, hujan}

ya tidak

Gambar 2.8 Hasil Pembentukan Cabang di *Node* 3 untuk Kasus “Apakah harus Bermain Baseball?” (*Prasetyo, 2014)*

Tabel 2.10 Pemisahan data menurut node 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Cuaca** | **Suhu** | **Kelembaban** | **Angin** | **Bermain** | **Keterangan** |
| Hujan | 70 | 96 | Pelan | Ya | Hasil proses node 3 |
| Hujan | 68 | 80 | Pelan | Ya | Hasil proses node 3 |
| Hujan | 65 | 70 | Kencang | Tidak | Hasil proses node 3 |
| Cerah | 72 | 95 | Pelan | Tidak | Hasil proses node 3 |
| Cerah | 69 | 70 | Pelan | Ya | Hasil proses node 3 |
| Hujan | 75 | 80 | Pelan | Ya | Hasil proses node 3 |
| Cerah | 75 | 70 | Kencang | Ya | Hasil proses node 3 |
| Hujan | 71 | 80 | Kencang | Tidak | Hasil proses node 3 |
| Cerah | 85 | 85 | Pelan | Tidak | Hasil proses node 3 |
| Cerah | 80 | 90 | Kencang | Tidak | Hasil proses node 3 |
| Mendung | 83 | 78 | Biasa | Ya |  |
| Mendung | 64 | 65 | Kencang | Ya |
| Mendung | 72 | 90 | Kencang | Ya |
| Mendung | 81 | 75 | Biasa | Ya |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:73)*

1. Proses untuk *node* 4

*Node* 4 untuk cabang suhu > 75 dengan label kelas bernilai tidak, dipastikan mempunyai *entropy* 0. Oleh karena itu, *node* 4 (yang dituju) dijadikan daun seperti pada Gambar 2.9

1. Proses untuk *node* 5

Pada *node* 5, fitur numerik kembali dilakukan perhitungan posisi *v* yang terbaik untuk pemecahan. Dalam contoh ini digunakan pemecahan biner. Hasil uji coba pada fitur ‘suhu’ dengan menghitung nilai *gain* pada Tabel 2.11. Nilai *gain* tertinggi didapatkan posisi *v* = 70 ketika menghitung *entropy* dan *gain* pada semua fitur.

Tabel 2.11 posisi v untuk pemecahan fitur ‘suhu’ di node 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Suhu** | **70** | | **75** | |
|  | **<=** | **>** | **<=** | **>** |
| Ya | 3 | 2 | 5 | 0 |
| Tidak | 1 | 2 | 3 | 0 |
| Gain | 0.0488 | | 0 | |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:73)*

Hasil uji coba pada fitur ‘kelembaban’ dengan menghitung nilai gain pada Tabel 2.12. Nilai *gain* tertinggi didapatkan pada posisi *v* = 80. Oleh karena itu, untuk fitur ‘kelembaban’ dilakukan *diskretisasi* pada *v* = 80 ketika menghitung *entropy* dan *gain* pada semua fitur.

Tabel 2.12 Posisi v untuk pemecahan fitur ‘kelembaban’dinode 5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Kelembaban** | **70** | | **75** | | **80** | |
|  | **<=** | **>** | **<=** | **>** | **<=** | **>** |
| Ya | 2 | 3 | 2 | 3 | 4 | 1 |
| Tidak | 1 | 2 | 1 | 2 | 2 | 1 |
| Gain | 0.0488 | | 0.0032 | | 0.0157 | |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:74)*

Selanjutnya dihitung *entropy* untuk setiap nilai fitur terhadap kelas, kemudian dihitung *gain* untuk setiap fitur. Hasilnya sebagai berikut:

Tabel 2.13 Hasil perhitunganentropy dan gain untuk node 5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Node** |  |  | **jml** | **Ya** | **Tidak** | **Entropy** | **Gain** |
| 5 | Total |  | 8 | 5 | 3 | 0.9544 |  |
|  | Cuaca |  |  |  |  |  | 0.0032 |
|  | Cerah | 3 | 2 | 1 | 0.9183 |  |
|  | Hujan | 5 | 3 | 2 | 0.9710 |  |
|  |  |  |  |  |  |  |
| Suhu |  |  |  |  |  | 0.0488 |
|  | <= 70 | 4 | 3 | 1 | 0.8113 |  |
|  | > 70 | 4 | 2 | 2 | 1.0000 |  |
| Kelembaban |  |  |  |  |  | 0.0157 |
|  | <= 80 | 6 | 4 | 2 | 0.9183 |  |
|  | >80 | 2 | 1 | 1 | 1.0000 |  |
| Angin |  |  |  |  |  | 0.1589 |
|  | Pelan | 5 | 4 | 1 | 0.7219 |  |
|  | Kencang | 3 | 1 | 2 | 0.9183 |  |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:74)*

Hasil yang ditunjukkan pada Tabel 2.13 menunjukkan bahwa *gain* tertinggi ada di fitur ‘angin’ sehingga fitur ‘angin’ dijadikan syarat kondisi di *node* 5 seperti Gambar 2.10. Pemisahan datanya ditunjukkan pada Tabel 2.14.

ya

tidak

mendung {cerah, hujan}

ya tidak

pelan kencang

Gambar 2.9 Hasil pembentukan cabang di node 5 untuk kasus “Apakah harus bermain baseball?” (*Prasetyo, 2014)*

Tabel 2.14 Pemisahan data menurut node 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Cuaca** | **Suhu** | **Kelembaban** | **Angin** | **Bermain** | **Keterangan** |
| Hujan | 70 | 96 | Pelan | Ya | Hasil proses node 5 |
| Hujan | 68 | 80 | Pelan | Ya | Hasil proses node 5 |
| Cerah | 65 | 95 | Pelan | Tidak | Hasil proses node 5 |
| Cerah | 72 | 70 | Pelan | Ya | Hasil proses node 5 |
| Hujan | 69 | 80 | Pelan | Ya | Hasil proses node 5 |
| Hujan | 75 | 70 | Kencang | Tidak | Hasil proses node 5 |
| Cerah | 75 | 70 | Kencang | Ya | Hasil proses node 5 |
| Hujan | 71 | 80 | Kencang | Tidak | Hasil proses node 5 |
| Cerah | 85 | 85 | Pelan | Tidak |  |
| Cerah | 80 | 90 | Kencang | Tidak |
| Mendung | 83 | 78 | Biasa | Ya |
| Mendung | 64 | 65 | Kencang | Ya |
| Mendung | 72 | 90 | Kencang | Ya |
| Mendung | 81 | 75 | Biasa | Ya |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:75)*

Pada perhitungan cabang berikutnya dari *node* 5, fitur ‘angin’ tidak lagi dilibatkan dalam perhitungan.

1. Proses untuk node 6

Pada *node* 6, fitur numerik kembali dilakukan perhitungan posisi *v* yang terbaik untuk pemecahan. Dalam contoh ini digunakan pada Tabel 2.15. Nilai *gain* tertinggi didapatkan pada posisi *v* = 70. Oleh karena itu, untuk fitur ‘suhu’ dilakukan *diskretisasi* pada *v* = 70 ketika menghitung *entropy* dan *gain* pada semua fitur.

Tabel 2.15 Posisi v untuk pemecahan fitur ‘suhu’ di node 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Suhu** | **70** | | **75** | |
|  | **<=** | **>** | **<=** | **>** |
| Ya | 3 | 1 | 3 | 1 |
| Tidak | 1 | 1 | 1 | 0 |
| Gain | 0.3219 | | 0.0729 | |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:75)*

Hasil uji coba pada fitur ‘kelembaban’ dengan menghitung nilai *gain* pada Tabel 2.16. Nilai *gain* didapatkan pada posisi *v* = 80 dan *v* = 85. Karena ada 2 yang tertinggi, maka bisa dipilih salah satu, misalnya 80. Oleh karena itu, ‘kelembaban’ dilakukan *diskretasi* pada *v* = 80 ketika menghitung *entropy* dan *gain* pada semua fitur.

Tabel 2.16 Posisi v untuk pemecahan fitur ‘kelembaban’ dinode 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Kelembaban** | **75** | | **80** | | **85** | |
|  | **<=** | **>** | **<=** | **>** | **<=** | **>** |
| Ya | 1 | 3 | 3 | 1 | 3 | 1 |
| Tidak | 0 | 1 | 0 | 1 | 0 | 1 |
| Gain | 0.0729 | | 0.3219 | | 0.3219 | |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:76)*

Selanjutnya dihitung *entropy* untuk setiap nilai fitur terhadap kelas, kemudian dihitung *gain* untuk setiap fitur. Hasilnya sebagai berikut:

Tabel 2.17 Hasil perhitungan entropy dan gainuntuk node 6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Node** |  |  | **jml** | **Ya** | **Tidak** | **Entropy** | **Gain** |
| 6 | Total |  | 5 | 4 | 1 | 0.7219 |  |
|  | Cuaca |  |  |  |  |  | 0.3219 |
|  | Cerah | 2 | 1 | 1 | 1.0000 |  |
|  | Hujan | 3 | 3 | 0 | 0 |  |
|  |  |  |  |  |  |  |
| Suhu |  |  |  |  |  | 0.3219 |
|  | <= 70 | 3 | 3 | 0 | 0 |  |
|  | > 70 | 2 | 1 | 1 | 1.0000 |  |
| Kelembaban |  |  |  |  |  | 0.3219 |
|  | <= 80 | 3 | 3 | 0 | 0 |  |
|  | >80 | 2 | 1 | 1 | 1.0000 |  |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:76)*

Hasil yang ditunjukkan pada Tabel 2.17 menunjukkan bahwa *gain* tertinggi ada pada ketiga fitur, maka bisa dipilih salah satu, misalnya ‘kelembaban’. Oleh karena itu, fitur ‘kelembaban’ dijadikan syarat kondisi di *node* 6 seperti Gambar 2.10, pemisahan datanya ditunjukkan pada Tabel 2.18.

ya

tidak

ya

mendung {cerah, hujan}

ya tidak

pelan

kencang

ya tidak

Gambar 2.10 Hasil Pembentukan Cabang di *Node* 6 untuk Kasus “Apakah harus Bermain Baseball?” (*Prasetyo, 2014)*

Tabel 2.18 Pemisahan data menurut node 6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Cuaca** | **Suhu** | **Kelembaban** | **Angin** | **Bermain** | **Keterangan** |
| Hujan | 68 | 80 | Pelan | Ya | Hasil proses node 6 |
| Cerah | 69 | 70 | Pelan | Ya | Hasil proses node 6 |
| Hujan | 75 | 80 | Pelan | Ya | Hasil proses node 6 |
| Hujan | 70 | 96 | Pelan | Ya | Hasil proses node 6 |
| Cerah | 72 | 95 | Pelan | Tidak | Hasil proses node 6 |
| Hujan | 65 | 70 | Kencang | Tidak |  |
| Cerah | 75 | 70 | Kencang | Ya |
| Hujan | 71 | 80 | Kencang | Tidak |
| Cerah | 85 | 85 | Pelan | Tidak |
| Cerah | 80 | 90 | Kencang | Tidak |
| Mendung | 83 | 78 | Biasa | Ya |
| Mendung | 64 | 65 | Kencang | Ya |
| Mendung | 72 | 90 | Kencang | Ya |
| Mendung | 81 | 75 | Biasa | Ya |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:77)*

Jika Tabel 2.18 diamati, untuk *node* 8 (cabang dari *node* 6) dipastikan menjadi daun karena nilai *entropy* 0 dengan cabang kiri (kelembaban <= 80) jatuh pada label kelas yang sama. Proses berikutnya dilanjutkan untuk *node* 7.

1. Proses untuk *node* 7

Pada *node* 7, fitur numerik kembali dilakukan perhitungan posisi *v* yang terbaik untuk pemecahan. Dalam contoh ini digunakan pemecahan biner. Hasil uji coba pada fitur ‘suhu’ dengan menghitung nilai *gain* pada Tabel 2.19. Nilai *gain* tertinggi didapatkan hanya pada posisi *v* = 70. Oleh karena itu, untuk fitur ‘suhu’ dilakukan *diskretasi* pada *v* = 70 ketika menghitung *entropy* dan *gain* pada semua fitur.

Tabel 2.19 Posisi v untuk pemecahan fitur ‘suhu’ di node7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Suhu** | **70** | | **75** | |
|  | <= | > | <= | > |
| Ya | 0 | 1 | 1 | 0 |
| Tidak | 1 | 1 | 2 | 0 |
| Gain | 0.2516 | | 0.0000 | |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:77)*

Hasil uji coba fitur ‘kelembaban’ dengan menghitung nilai *gain* pada Tabel 2.20. Nilai *gain* didapatkan hanya pada posisi *v* = 75. Oleh karena itu, untuk fitur ‘kelembaban’ dilakukan diskretasi pada *v* = 75 ketika menghitung *entropy* dan *gain* pada semua fitur.

Tabel 2.20 Posisi *v* untuk pemecahan fitur ‘kelembaban’ di node 7

|  |  |  |
| --- | --- | --- |
| **Kelembaban** | **75** | |
|  | <= | > |
| Ya | 1 | 0 |
| Tidak | 1 | 1 |
| Gain | 0.2516 | |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:78)*

Selanjutnya dihitung *entropy* untuk setiap nilai fitur terhadap kelas, dan *gain* untuk setiap fitur. Hasilnya sebagai berikut:

Tabel 2.21 Hasil perhitungan entropy dan gain untuk node 7

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Node** |  |  | **Jml** | **Ya** | **Tidak** | **Entropy** | **Gain** |
| 6 | Total |  | 3 | 1 | 2 | 0.9183 |  |
|  | Cuaca |  |  |  |  |  | 0.9183 |
|  | Cerah | 1 | 1 | 0 | 0 |  |
|  | Hujan | 2 | 0 | 2 | 0 |  |
| Suhu |  |  |  |  |  | 0.2516 |
|  | <= 70 | 1 | 0 | 1 | 0 |  |
|  | > 70 | 2 | 1 | 1 | 1.0000 |  |
| Kelembaban |  |  |  |  |  | 0.2516 |
|  | <= 75 | 2 | 1 | 1 | 1.0000 |  |
|  | > 75 | 1 | 0 | 1 | 0 |  |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:78)*

Hasil yang ditunjukkan pada Tabel 2.21 menunjukkan bahwa *gain* tertinggi ada di fitur ‘cuaca’, berarti fitur ‘cuaca’ dijadikan syarat kondisi di *node* 7, seperti ditunjukkan pada Gambar 2.12. Pemisahan datanya pada Tabel 2.22.

tidak

ya

ya

ya

tidak

mendung {cerah, hujan}

ya tidak

pelan kencang

ya tidak cerah hujan

Gambar 2.11 Hasil pembentukan cabang di node 7 untuk kasus “Apakah harus bermain baseball?” (*Prasetyo, 2014)*

Tabel 2.22 Pemisahan data menurut node 7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Cuaca** | **Suhu** | **Kelembaban** | **Angin** | **Bermain** | **Keterangan** |
| Hujan | 68 | 80 | Pelan | Ya |  |
| Cerah | 69 | 70 | Pelan | Ya |
| Hujan | 75 | 80 | Pelan | Ya |
| Hujan | 70 | 96 | Pelan | Ya |
| Cerah | 72 | 95 | Pelan | Tidak |
| Hujan | 65 | 70 | Kencang | Tidak | Hasil proses node 7 |
| Hujan | 71 | 80 | Kencang | Tidak | Hasil proses node 7 |
| Cerah | 75 | 70 | Kencang | Ya | Hasil proses node 7 |
| Cerah | 85 | 85 | Pelan | Tidak |  |
| Cerah | 80 | 90 | Kencang | Tidak |
| Mendung | 83 | 78 | Biasa | Ya |
| Mendung | 64 | 65 | Kencang | Ya |
| Mendung | 72 | 90 | Kencang | Ya |
| Mendung | 81 | 75 | Biasa | Ya |

*Sumber : Eko Prasetyo, Data Mining- Mengolah data & Informasi (2014:79)*

Jika Tabel 2.21 diamati, untuk *node* 10 dan 11 (cabang dari *node* 7) dipastikan menjadi daun karena nilai *entropy* 0, dengan masing-masing cabang jatuh pada label kelas yang sama.

1. Proses untuk *node* 8

*Node* 8 untuk cabang ‘kelembaban <= 80’ dengan label kelas bernilai tidak, dipastikan mempunyai *entropy* 0. Oleh karena itu, *node* 8 (yang dituju) dijadikan daun. Hal ini seperti Gambar 2.12.

1. Proses pada *node* 9

*Node* 9 hanya tersisa 2 data dengan komposisi label 1 kelas ya dan 1 kelas tidak. Nilai *gain* yang didapat dari fitur ‘cuaca’, ‘suhu’, dan ‘kelembaban’ dipastikan juga sama sehingga bisa langsung dipilih salah satu misalnya ‘cuaca’. Hasil pohon yang didapat seperti Gambar 2.13. *node* 9 ini menjadi *node* cabang yang seharusnya tidak perlu dibentuk (sebaiknya cabang dipotong) karena menyebabkan pohon menjadi *overfitting,* dapat memengaruhi *trade-off* antara akurasi prediksi terhadap pelatihan dilakukan, Karena tidak ada lagi *node* yang diproses, maka induksi *decision tree* dinyatakan selesai. Hasil akhir *decision tree* seperti Gambar 2.13.

Bentuk aturan *if then* untuk *decision tree* sebagai berikut:

*If*  cuaca = mendung *then* playball = ya

*If* cuaca = {cerah, hujan} *and* suhu > 75 *then* playball = tidak

*If* cuaca = {cerah, hujan} *and* suhu <= 75 *and* angin = pelan *and* kelembaban <= 80 *then* playball = ya

*If* cuaca = cerah *and* suhu <= 75 *and* angin = pelan *and* kelembaban >80 *then* playball = ya

*If* cuaca = hujan *and* suhu <= 75 *and* angin = pelan *and* kelembaban >80 *then* playball = tidak

*If* cuaca = cerah *and* suhu <= 75 *and* angin = kencang *then* playball = ya

*If* cuaca = hujan *and* suhu <= 75 *and* angin = kencang *then* playball = tidak

ya

tidak

ya

ya

ya

tidak

tidak

mendung {cerah, hujan}

ya tidak

pelan kencang

ya tidak

hujan cerah hujan cerah

Gambar 2.12 Hasil Pembentukan cabang di node 9 untuk kasus “Apakah harus bermain baseball?” (*Prasetyo, 2014)*

* 1. ***Unified Modelling Language* (UML)dan *Use Case Diagram* (UCD)**

Menurut Nugroho (2010:6) *UML* *(Unified Modeling Language)* adalah bahasa pemodelan untuk sistem atau perangkat lunak yang paradigma berorientasi objek. Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami.

Sedangkan *use case* Menurut Nugroho (2010:34) pada dasarnya merupakan unit fungsionalitas koheren yang diekspresikan sebagai transaksi-transaksi yang terjadi antara *actor* dan sistem.

**2.3.1 Sejarah Singkat *UML***

Pendekatan analisa & rancangan dengan menggunakan model OO mulai diperkenalkan sekitar pertengahan 1970 hingga akhir 1980 dikarenakan pada saat itu aplikasi *software* sudah meningkat dan mulai kompleks. Jumlah yang menggunakan metode OO mulai diuji coba dan diaplikasikan antara 1989 hingga 1994, seperti halnya oleh *Grady Booch* dari *Rational Software Co.* Metode inidikenal dengan *OOSE* (*Object-Oriented Software Engineering*), serta *James Rumbaugh* dari *General Electric*, dikenal dengan *OMT* (*Object Modelling Technique*).

Kelemahan saat itu disadari oleh *Booch* maupun *Rumbaugh* adalah tidak adanya standar penggunaan model yang berbasis OO, ketika mereka bertemu ditemani rekan lainnya *Ivar Jacobson* dari *Objectory* mulai mendiskusikan untuk mengadopsi masing-masing pendekatan metode OO untuk membuat suatu model bahasa yang *uniform*/seragam yang disebut *UML* (*Unified Modeling Language*) dan dapat digunakan oleh seluruh dunia.

Secara resmi bahasa *UML* dimulai pada bulan oktober 1994, ketika *Rumbaugh* bergabung *Booch* untuk membuat sebuah project pendekatan metode yang *uniform* atau seragam dari masing-masing metode mereka. Saat itu baru dikembangkan *draft* metode *UML* *version* 0.8 dan diselesaikan serta di *release* pada bulan Oktober 1995. Bersamaan dengan saat itu, *Jacobson* bergabung dan *UML* tersebut diperkaya ruang lingkupnya dengan metode *OOSE* sehingga muncul *release* version 0.9 pada bulan Juni 1996. Hingga saat ini sejak Juni 1998 *UML* *version* 1.3 telah diperkaya dan direspon oleh *OMG* (*Object Management Group*), *Anderson Consulting*, *Ericsson*, *Platinum Technology*, *Object Time Limited*, dll serta di pelihara oleh *OMG* yang dipimpin oleh *Cris Kobryn*.

**2.3.2 Pengenalan** ***Unified Modelling Language (UML)***

*UML* adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan dan membangun sistem *software*. *UML* adalah hasil pengembangan dari bahasa pemodelan berorientasi objek (*object oriented modeling language).* Saat ini, versi *UML* yang digunakan adalah versi 2.0 yang merupakan hasil dari pengembangan metode yang dikreasikan oleh *Grady Booch*, *Jim Rambaugh* dan *Ivar Jacobson*. Sebenarnya terdapat 9 jenis diagram pada *UML*. Namun, hanya dua jenis diagram dalam *UML* yang umum diketahui yakni :

1. Diagram *use case,* yang menunjukkan sekelompok *use case* dan bagaimana pelaku (*actor*) dapat menggunakannya.
2. Diagram *class*, menggambarkan struktur sistem, membaginya dalam kelas-kelas dengan koneksi dan relasi yang berbeda-beda.

*UML* adalah salah satu *tool* / model untuk merancang pengembangan *software* yang berbasis *object oriented. UML* sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalambahasa program yang spesifik, skema *database*, dan komponen-komponenyang diperlukan dalam sistem *software*.

*UML* adalah sebuah bahasa standar untuk pengembangan sebuah *software* yang dapat menyampaikan bagaimana membuat dan membentuk model-model, tetapi tidak menyampaikan apa dan kapan model yang seharusnya dibuat yang merupakan salah satu proses implementasi pengembangan *software*. *UML* tidak hanya merupakan sebuah bahasa pemograman visual saja, namun juga dapat secara langsung dihubungkan ke berbagai bahasa pemograman, seperti *JAVA*, *C++, Visual Basic*, atau bahkan dihubungkan secara langsung ke dalam sebuah *object-oriented database*.

Begitu juga mengenai pendokumentasian dapat dilakukan seperti *requirements*, arsitektur, *design*, *source code*, *project plan*, *tests*, dan *prototypes*. Untuk dapat memahami *UML* membutuhkan bentuk konsep dari sebuah bahasa model dan mempelajari 3 (tiga) elemen utama *UML*. Ketiga elemen itu seperti *building block*, aturan-aturan yang menyatakan bagaimana *building block* diletakkan secara bersamaan, dan beberapa mekanisme umum (*common*).

**2.3.2.1 *Use Case Diagram***

Berikut ini akan diuraikan mengenai pengertian *Use Case Diagram* dan *Activity Diagram*.

**2.3.2.2 Pengertian *Use Case Diagram***

*Use case diagram* menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. Menggambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem.

*Use case* adalah deskripsi fungsi yang disediakan oleh sistem dalam bentuk teks sebagai dokumentasi dari *use case symbol* namun dapat juga dilakukan dalam *activity diagram*. *Use case* digambarkan hanya yang dilihat dari luar oleh *actor* (keadaan lingkungan sistem yang dilihat *user*) dan bukan bagaimana fungsi yang ada di dalam sistem.

Beberapa komponen *use case diagram* dapat dilihat pada tabel dibawah ini:

Tabel 2.23 Simbol-Simbol *Use Case* *Diagram*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Gambar** | **Nama Komponen** | **Deskripsi** |
| 1 |  | *Use Case* | Menerangkan “apa” yang dikerjakan sistem, bukan “bagaimana” sistem mengerjakannya. |
| 2 |  | *Actor* | Menggambarkan orang, sistem atau *external* entitas / *stakeholder*  yang menyediakan atau menerima informasi dari sistem. |
| 3 |  | *System Boundary* | Menggambarkan jangkauan sistem. |
| 4 |  | *Association* | Menggambarkan bagaimana *actor* terlibat dalam *use case*. |
| 5 |  | *Generalization* | Dibuat ketika ada sebuah keadaan yang lain/perlakuan khusus. |

Tabel 2.23 Simbol-Simbol *Use Case Diagram* (lanjutan)

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Gambar** | **Nama Komponen** | **Deskripsi** |
| 6 | << *Extend*>> | *Extend* | Perluasan dari *use case* lain jika kondisi atau syarat terpenuhi. |
| 7 | << *include*>> | *Include* | Menjelaskan bahwa *use case* termasuk didalam *use case* lain. |

*(Sumber: Nugroho, 2010)*

**2.3.2.3 *Activity Diagram***

Menggambarkan proses bisnis dan urutan aktifitas dalam sebuah proses. Dipakai pada *business modelling* untuk memperlihatkan urutan aktifitas proses bisnis. Struktur diagram ini mirip *flowchart* atau *Data Flow Diagram* pada perancangan terstruktur. Sangat bermanfaat apabila kita membuat diagram ini terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan. *Activity diagram* dibuat berdasarkan sebuah atau beberapa *use case* pada *use case diagram*. Berikut simbol-simbol *activity diagram* dapat dilihat pada tabel dibawah ini :

Tabel 2.24 Simbol-Simbol *Activity* *Diagram*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Simbol** | **Nama** | **Keterangan** |
| 1 |  | *Start Point* | Titik Awal yang menunjukan mulainya suatu aliran aktivitas dalam suatu sistem pada sebuah *activity diagram*. Hanya ada satu *start point* dalam sebuah aliran aktivitas. |

Tabel 2.24 Simbol-Simbol *Activity* *Diagram* (lanjutan)

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Simbol** | **Nama** | **Keterangan** |

|  |  |  |  |
| --- | --- | --- | --- |
| 2 |  | *End Point* | Titik Akhir yang menunjukan akhir suatu aliran aktivitas dalam suatu sistem pada sebuah *activity diagram*. Bisa terdapat lebih dari satu *end point* dalam sebuah aliran aktivitas |
| 3 |  | *Activity* | Simbol yang mengambarkan sebuah pekerjaan atau tugas dalam aliran aktivitas dalam suatu sistem. |

|  |  |  |  |
| --- | --- | --- | --- |
| 4. |  | *Fork* (percabangan) | Simbol yang digunakan untuk menggambarkan proses percabangan dalam aliran aktivitas sistem. |
| 5. |  | *Join* (penggabungan) | Simbol yang digunakan untuk menggambarkan proses penggabungan dalam aliran aktivitas sistem. |
| 6. |  | *Decision* | Suatu titik atau *point* pada *activity* diagram yang mengindikasikan suatu kondisi dimana ada kemungkinan perbedaan keputusan antara ya dan tidak. |
| 7. | **Nama Swimlane** | *Swimlane* | Sebuah cara untuk mengelompokkan *activity* berdasarkan *actor* (mengelompokkan *activity* dalam sebuah urutan yang sama) |

*(Sumber: Nugroho, 2010)*

**2.4Flowchart**

*Flowchart* merupakan gambar atau bagan yang memperlihatkan urutan dan hubungan antar proses beserta instruksinya. Gambaran ini dinyatakan dengan simbol. Dengan demikian setiap simbol menggambarkan proses tertentu. Sedangkan hubungan antar proses digambarkan dengan garis penghubung. *Flowchart* ini merupakan langkah awal pembuatan program. Dengan adanya *flowchart* urutan poses kegiatan menjadi lebih jelas. Jika ada penambahan proses maka dapat dilakukan lebih mudah. Setelah *flowchart* selesai disusun, selanjutnya menerjemahkannya ke bentuk program dengan bahasa pemrograman.

**2.4.1 Simbol - simbol flowchart**

*Flowchart* disusun dengan simbol-simbol. Simbol ini dipakai sebagai alat bantu menggambarkan proses di dalam program. Seperti pada tabel dibawah ini:

Tabel 2.25 Simbol *Flowchart*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Simbol** | **Nama** | **Keterangan** |
| 1 |  | *Document* | Menggambarkan semua jenis dokumen yang merupakan formulir yang digunakan untuk data terjadinya suatu transaksi. |
| 2 |  | *Manual Operation* | Digunakan untuk menggambarkan kegiatan manual. |
| 3 |  | *Punched card* | Menunjukan *input* atau *output* yang menggunakan kartu plong (*punched card*). |

Tabel 2.25 Simbol *Flowchart* (lanjutan)

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Simbol** | **Nama** | **Keterangan** |
| 4 |  | *Process* | Menunjukan kegiatan proses dari operasi program komputer. |
| 5 |  | *Input / Output* | Merepresentasikan *input* atau *output* data. |
| 6 |  | *Decision* | Menggambarkan keputusan yang harus dibuat dalam proses pengolahan data. |
| 7 |  | *Flow line* | Menggambarkan arah dari proses. |
| 8 |  | *Preparation* | Pemberian harga awal. |

*(Sumber: Nugroho, 2010)*

* 1. **Pengertian PHP (Hypertext Preprocessor)**

PHP singkatan dari Hypertext Preprocessor yang digunakan sebagai bahasa *script server-side* dalam pengembangan web yang disisipkan pada dokumen HTML. PHP merupakan software *open-source* yang disebarkan dan dilisensikan secara gratis.

**2.5.1 Kelebihan Pemrograman PHP**

PHP memiliki banyak kelebihan yang tidak dimiliki oleh bahasa *script* sejenis. PHP berbasis *server side* *scripting*, PHP sendiri dapat melakukan tugas-tugas yang dilakukan dengan mekanisme CGI seperti mengambil, mengumpulkan data dari database, menghasilkan isi halaman web dinamis dan kemampuan mengirim serta menerima *cookies*.

PHP dapat digunakan pada semua sistem operasi, antara lain Linux, Unix, Windows, Mac OSX, RISC OS, dan sistem operasi lainnya.

**2.5.2 Sejarah PHP**

Pada awalnya PHP merupakan kependekan dari *Personal Home Page* (situs personal). PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1994. Awalnya PHP digunakan untuk mencatat jumlah serta untuk mengetahui siapa saja pengunjung pada homepagenya. Rasmus Lerdof mengeluarkan *Personal Home Page Tools* versi 1.0 dan meluncurkan PHP 2.0 secara gratis.

Pada tahun 1996, PHP telah banyak digunakan dalam *website* di dunia. Pada tahun 1998, PHP 3.0 diluncurkan, penyempurnaan terus dilakukan sehingga pada tahun 2000 dikeluarkan PHP 4.0. Kemampuan PHP terus ditambah dan versi terbaru dikeluarkan PHP 5.0.

* 1. **Pengertian MySQL**

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*database management system*) atau DBMS yang *multithread, multi-user,* dengan sekitar 6 juta instalasi di seluruh dunia. MySQL merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian database, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomastis.

DBMS memiliki kemampuan yang terintegrasi seperti :

1. Membuat, menghapus, menambah, dan memodifikasi basis data
2. DBMS pengelolanya berbasis windows sehingga lebih mudah digunakan
3. Kemampuan berkomunikasi dengan program aplikasi lain. Misalnya untuk mengakses basis data MySQL menggunakan aplikasi yang menggunakan PHP
4. Kemampuan pengaksesan melalui komunikasi antar komputer (*client server*)

MySQL adalah salah satu aplikasi DBMS yang sudah sangat banyak digunakan oleh pemrogram aplikasi *web.* Contoh DBMS lainnya adalah : *PostgreSQL* (*freeware*), *SQLserver*, *MS access* dari *Microsoft*, *DB2* dari *IBM, Oracle dan Oracle Corp, Dbase, FoxPro*, dan sebagainya.