Metode *Classification*

Metode *Classification* adalah sebuah metode dari *data mining* yang digunakan untuk memprediksi kategori atau kelas dari suatu *data instance* berdasarkan sekumpulan atribut-atribut dari data tersebut. Atribut yang digunakan mungkin bersifat *categorical* (misalnya golongan darah : "A", "B", "O", dst), *ordinal* (misalnya urutan : *small, medium,* dan *large*), *integer-valued* (misalnya banyaknya suatu kata pada suatu paragraf), atau *real-valued* (misalnya suhu). Kebanyakan algoritma yang menggunakan metode klasifikasi ini hanya menggunakan data yang bersifat diskret dan untuk data yang bersifat kontinu (*real-valued* dan *integer-valued*) maka data tersebut harus dijadikan diskret dengan cara memberikan *threshold* (misal lebih kecil dari 5 atau lebih besar dari 10) supaya data dapat terbagi menjadi grup-grup. Sebagai contoh dari metode klasifikasi adalah menentukan *e-mail* yang masuk termasuk kategori *spam* atau bukan *spam* atau menentukan diagnosis dari pasien berdasarkan umur, jenis kelamin, tekanan darah, dan sebagainya (Tan, 2004).

Algoritma yang mengimplementasikan metode ini disebut dengan *classifier*. Istilah *"classifier"* ini juga terkadang direferensikan sebagai fungsi matematika yang digunakan untuk memetakan input data dengan kategori-kategori tertentu.

Cara kerja dari metode *Classification* adalah sebuah proses 2 langkah. Langkah pertama adalah *Learning*. Pada langkah ini, *classifier* dibangun berdasarkan sekumpulan kelas atau kategori yang sudah ditentukan dari data. Langkah ini disebut *learning step* atau *training step,* dimana sebuah algoritma *classification* membangun *classifier* dengan menganalisis atau "belajar dari" sebuah *training set.* Sebuah *tuple* X, yang direpresentasikan dengan *n*-dimensi *attribute vector*, $X = \{x_1, x_2, \dots, x_n\}$ yang menggambarkan *n* buah pengukuran yang dibuat pada *tuple* pada *n attribute* $A_1, A_2, \dots, A_n$. Setiap *tuple* diasumsikan termasuk dalam kelas atau kategori yang sudah ditentukan oleh *attribute* yang disebut dengan *class label attribute. Class label attribute* mempunyai nilai diskret, tidak berurutan dan tiap nilai berfungsi sebagai kelas atau kategori (Han, 2006).

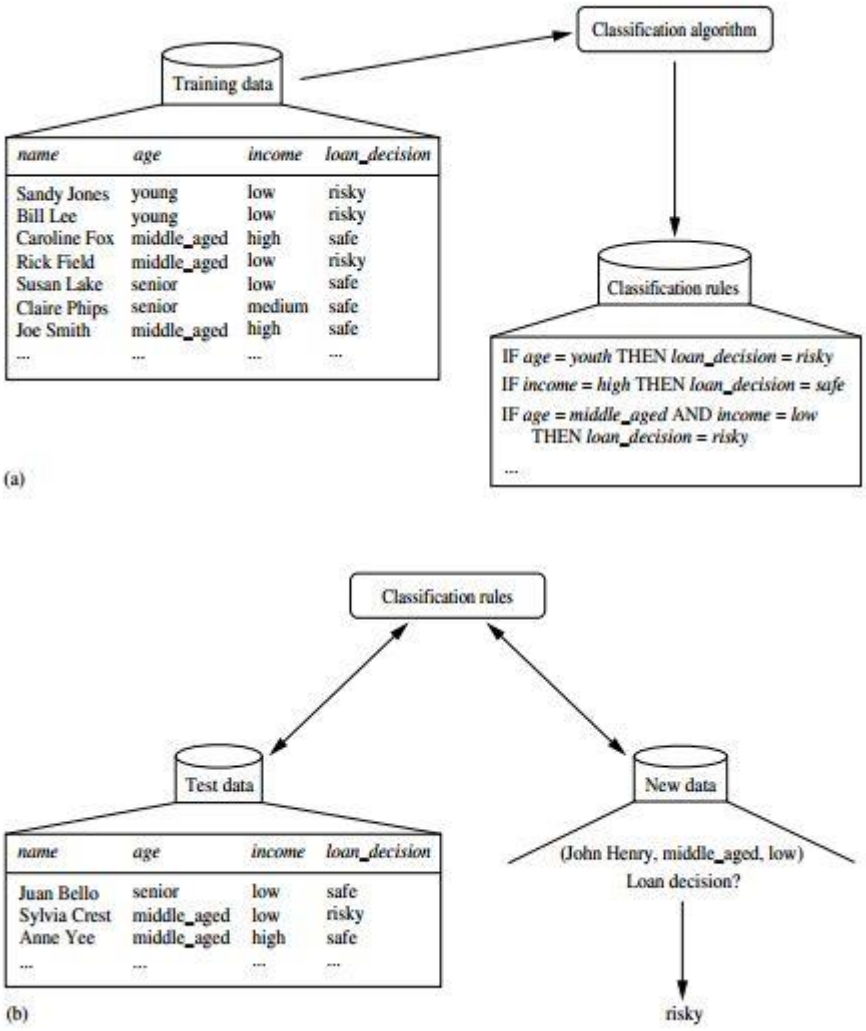
Langkah pertama dari *classification* juga disebut sebagai *learning of mapping* atau *function* $y = f(X)$, suatu fungsi pemetaan yang bisa memprediksi *class label y* pada suatu tuple *X*. Pemetaan ini direpresentasikan dalam bentuk *classification rules, decision tree* atau formula matematika. Dari *rules* atau *tree* tersebut dapat digunakan untuk mengklasifikasi *tuple* baru (Han, 2006).

Langkah kedua adalah *Classification*. Pada langkah ini, *classifier* yang sudah dibangun akan digunakan untuk mengklasifikasi data. Pertama, akurasi dari prediksi *classifier* tersebut diperkirakan. Jika menggunakan *training set* untuk mengukur akurasi dari *classifier*, maka estimasi akan optimis karena data yang digunakan untuk membentuk *classifier* adalah *training set* juga. Oleh karena itu, digunakan *test set*, yaitu sekumpulan *tuple* beserta *class label*-nya yang dipilih secara acak dari dataset. *Test set* bersifat independen dari *training set* dikarenakan *test set* tidak digunakan untuk membangun *classifier* (Han, 2006).

Akurasi dari *classifier* yang diestimasikan dengan *test set* adalah persentase dari *tuple test set* yang diklasifikasi secara benar oleh *classifier*. *Class label* dari setiap *tuple* dari *test set* dibandingkan dengan prediksi *class label* dari *classifier*. Jika akurasi dari *classifier* dapat diterima maka *classifer* dapat digunakan untuk mengklasifikasi data baru. Gambar 2.3 merupakan ilustrasi dari langkah *Learning* dan *Classification* dari metode *Classification* (Han, 2006).

Metode *Classification* termasuk dari "*supervised learning"* karena *class label* dari setiap *tuple* sudah disediakan. Berbeda dengan *"unsupervised learning"* dimana *class label* dari setiap *tuple* tidak diketahui. Metode yang menggunakan *unsupervised learning* adalah metode *Clustering* (Han, 2006).

Terdapat beberapa algoritma data mining yang menggunakan metode *Classification* ini, seperti C4.5, CMAR, Naïve Bayes, K Nearest Neighbours dan algoritma yang penulis implemetasikan, CART.



Gambar 1.1 Ilustrasi langkah dari *Classification Method*. (a) *Learning*. (b) *Classification*. (Han, 2006)

1.1. CART Algorithm

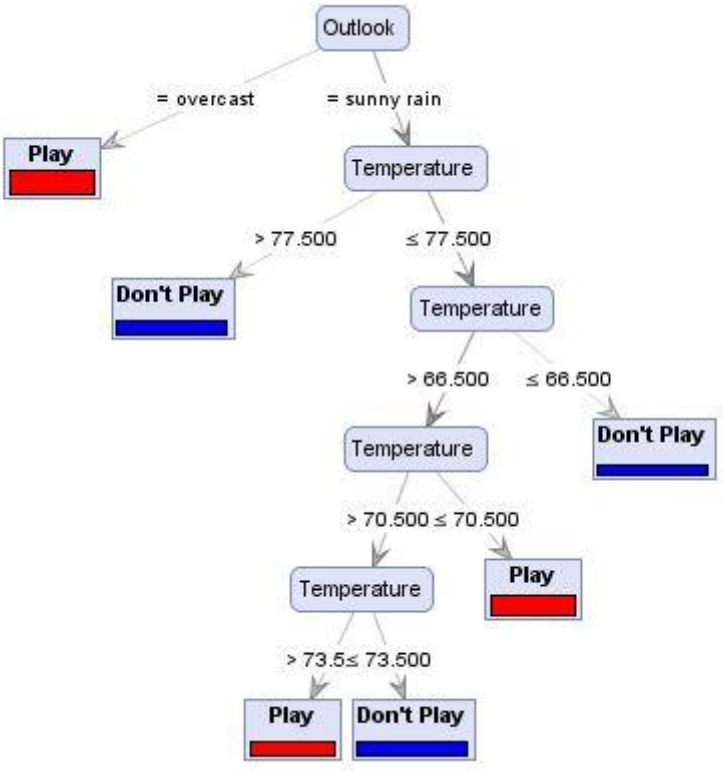
CART merupakan singkatan dari *Classification And Regression Trees*. CART adalah metode klasifikasi yang menggunakan data historis untuk membentuk *decision tree* yang dapat digunakan untuk mengklasifikasi data baru. Metodologi CART dikembangkan oleh Breiman, Freidman, Olshen, Stone pada sekitar tahun 1984 dalam *paper* mereka yang berjudul *"Classification and Regression Trees"*. Untuk membangun *decision tree*, CART menggunakan *learning sample*, sekumpulan data historis yang sudah ditetapkan kelas-kelasnya untuk observasi (Timofeev, 2004).

Decision Tree adalah representasi dari sekumpulan pertanyaan yang akan membelah *learning sample* menjadi bagian yang lebih kecil. Pertanyaan yang diajukan *decision tree* biasanya berupa *yes/no question*, seperti "Is age greater than 50?" atau "Is sex male?". Oleh karena itu, *Decision Tree* yang terbentuk bersifat *binary* atau selalu bercabang dua. Algoritma CART akan mencari semua kemungkinan variabel dan nilai untuk menemukan *split* yang paling baik dari pertanyaan yang yang akan membagi *learning sample* menjadi 2 bagian dengan homogenitas maksimal. Proses akan dilanjutkan sampai *decision tree* menghasilkan *data fragment*, yaitu suatu data yang tidak bisa dibelah lagi (Timofeev, 2004).

Sesuai dengan nama algoritmanya, CART dapat membentuk 2 tipe *decision tree*, yaitu *Classification Tree* dan *Regression Tree*. Kedua *tree* tersebut mempunyai kegunaan yang berbeda.

1.1.1. *Classification Tree*

Classification Tree digunakan untuk mengklasifikasi data historis berdasarkan atribut kelas. Input dari *Classification Tree* adalah sekumpulan *tuple* data yang disebut dengan dataset atau *learning sample* (Tan, 2004). Setiap *tuple* mempunyai sekumpulan atribut $\{x_1, x_2, \dots, x_n, y\}$, dimana $x_1, x_2, \dots, x_n$ atribut-atribut yang akan diklasifikasi dan y adalah atribut kelas dari *tuple*. Sebagai contoh *tree* dan dataset dari *Classification Tree* dapat dilihat pada Gambar 2.4 dan Tabel 2.1



Gambar 1.2 Contoh *Classification Tree* yang dibangun dengan tools RapidMiner

Tabel 1.1 Contoh dataset dari *Classification Tree*

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	FALSE	Don't Play
sunny	80	90	TRUE	Don't Play
overcast	83	78	FALSE	Play
rain	70	96	FALSE	Play
rain	68	80	FALSE	Play
rain	65	70	TRUE	Don't Play
overcast	64	65	TRUE	Play
sunny	72	95	FALSE	Don't Play
sunny	69	70	FALSE	Play
rain	75	80	FALSE	Play
sunny	75	70	TRUE	Play
overcast	72	90	TRUE	Play
overcast	81	75	FALSE	Play
rain	71	80	TRUE	Don't Play

Pada Tabel 2.1 terdapat dataset yang berisi informasi mengenai prediksi keputusan bermain golf yang dipengaruhi oleh kondisi cuaca. Dataset tersebut memiliki 14 *tuple* atau baris data yang terdiri dari 4 atribut, yaitu *Outlook*, *Temperature*, *Humidity*, dan *Windy* serta atribut kelas *Play*. Atribut *Outlook* dan *Windy* bersifat diskret sedangkan atribut *Temperature* dan *Humidity* bersifat kontinu. Untuk atribut kelas harus bersifat diskret karena akan menjadi penentu dari klasifikasi dataset.

Classification Tree dibangun berdasarkan *splitting rule*, yaitu suatu *rule atau* aturan yang menentukan dan melakukan proses pembelahan dari dataset (Tan, 2004). *Classification Tree* mempunyai beberapa *splitting rule* untuk membelah data. Salah satunya adalah *Gini Index*.

Suatu *Classification Tree* bisa jadi memiliki kompleksitas tinggi dan memiliki ratusan tingkat atau level sehingga akan memperburuk performa. Untuk itu *Classification Tree* harus dioptimalisasi dengan cara *pruning* atau memangkas cabang dari *tree* yang tidak diperlukan (Timofeev, 2004). *Classification Tree* memiliki beberapa algoritma *pruning* dan salah satunya adalah *Optimization by minimum number of points*.

1.1.1.1. *Gini Index*

Gini Index mengukur tingkat homogenitas dari data D dengan rumus

$$Gini\ (D) = 1 - \sum_{i=1}^{m} p_i^2,$$

dimana m adalah banyaknya atribut kelas pada D, p_i adalah probabilitas suatu tuple pada D memiliki atribut kelas C_i dan dihitung dengan rumus $\frac{|C_{i,D}|}{|D|}$, yaitu membagi banyaknya atribut kelas C_i pada D dengan banyaknya *tuple* pada D. Tingkat homogenitas dari data tersebut disebut dengan *impurity* (Han, 2006).

Untuk membelah data dengan homogenitas maksimal, *Gini Index* memerlukan *Splitting Attribute* atau atribut pembelah. Atribut pembelah merupakan atribut yang akan membelah data membagi 2 partisi dan memiliki nilai *Gini Index*

terendah. Oleh karena itu, setiap atribut harus dicari nilai *Gini Index*-nya. Perhitungan *Gini Index* dari atribut pembelah A dilakukan dengan perkalian antara *Gini Index* dari tiap hasil partisi dengan bobotnya masing-masing. Hal ini dilakukan karena CART bersifat *binary*. Oleh karena itu, untuk data D yang terpartisi oleh atribut pembelah A akan menjadi D_1 dan D_2 dengan rumus :

$$Gini_A = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

Untuk setiap atribut, setiap kemungkinan partisi akan selalu dihitung. Untuk atribut A yang bersifat diskret yang memiliki v nilai yang berbeda dan $v > 2$, maka semua kemungkinan subset dari A dengan $\frac{2^v - 2}{2}$ banyaknya kemungkinan subset. Setiap subset akan dihitung nilai *Gini Index*-nya dan diambil subset yang memiliki *Gini Index* terendah sebagai kandidat atribut pembelah (Han, 2006).

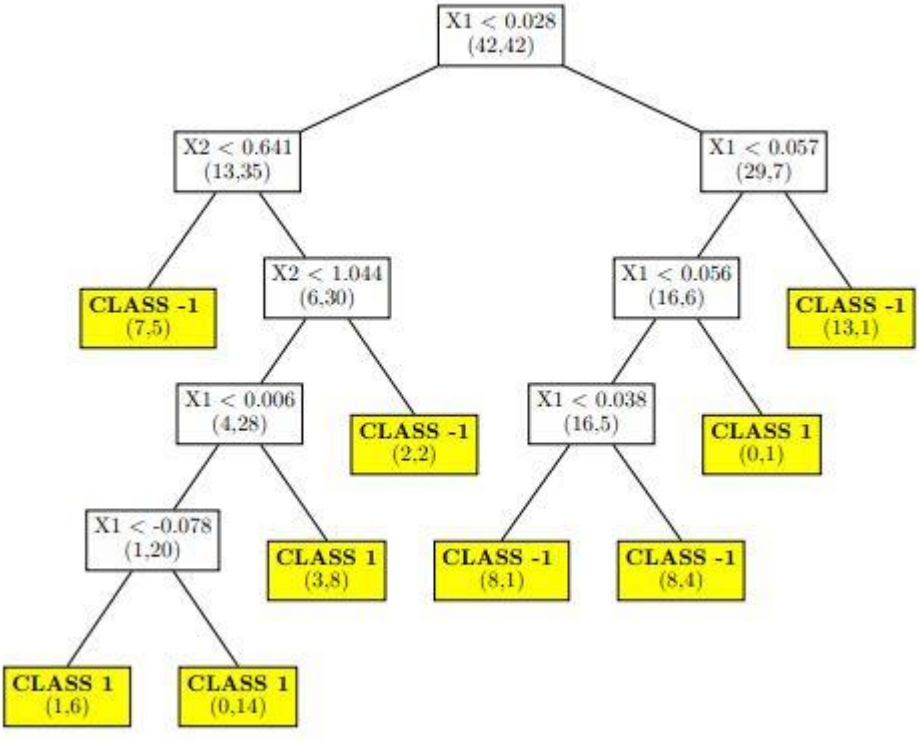
Lalu untuk atribut A yang bersifat kontinu, maka kita harus mencari "*split-point*" terbaik dimana *split-point* ini akan menjadi pembatas nilai-nilai pada atribut A. Untuk mencari *split-point* tersebut, nilai-nilai pada atribut A harus disortir dari yang terkecil sampai yang terbesar, lalu nilai tengah dari pasangan nilai yang berdekatan dianggap sebagai salah satu kemungkinan *split-point*. Oleh karena itu, jika atribut A memiliki v nilai, maka akan ada $v - 1$ banyaknya kemungkinan *split-po*int yang akan dievaluasi. Sebagai contoh nilai tengah dari nilai a_i dan a_{i+1} adalah :

$$\frac{a_i + a_{i+1}}{2}$$

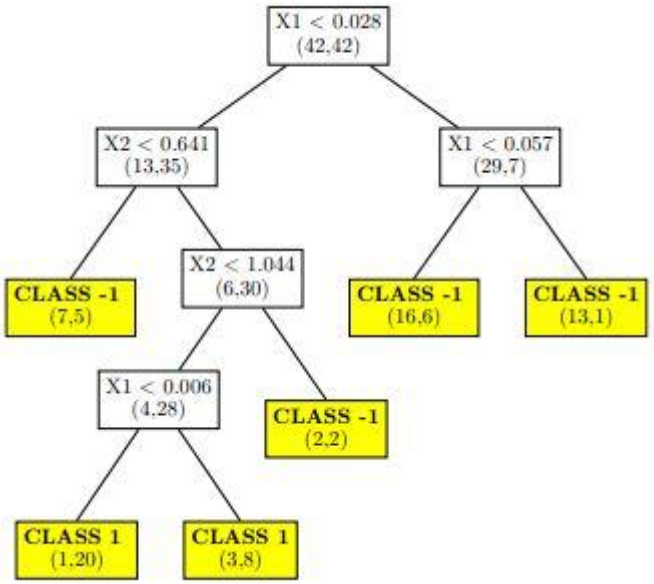
Maka dari rumus tersebut akan didapat sekumpulan kemungkinan *split-point* dari atribut A. Setiap kemungkinan *split-point* tersebut akan dievaluasi dengan cara mencoba mempartisi *learning sample* dengan setiap kemungkinan *split-point* tersebut dengan aturan dimana setiap nilai $A \leq$ *split-point* akan terpartisi ke D_1 dan setiap nilai $A >$ *split-point* akan terpartisi ke D_2 dan *Gini Index*-nya akan dievaluasi dengan cara yang sama seperti mencari *Gini Index* pada atribut diskret (Han, 2006).

1.1.1.2. *Optimization by Minimum Number of Points*

Optimization by Minimum Number of Points adalah salah satu algoritma *pruning* yang digunakan pada CART. Algoritma ini menghentikan proses pembelahan data ketika jumlah data atau *tuple* pada suatu *node* kurang dari N_{min}, jumlah minimal data yang ditentukan. Semakin besar N_{min}, maka semakin kecil *Decision Tree*. Dalam penggunaannya, biasanya ukuran N_{min} adalah 10% dari ukuran *learning sample* (Timofeev, 2004). Contoh dari penggunakan *Optimization by Minimum Number of Points* dapat dilihat pada Gambar 2.5 dan Gambar 2.6. Gambar 2.5 adalah ilustrasi dari suatu *Classification Tree* dengan N_{min} = 15 dan Gambar 2.6 adalah ilustrasi dari suatu *Classification Tree* dengan N_{min} = 30



Gambar 1.3 Contoh *Classification Tree* dengan N_{min} = 15. (Timofeev, 2004)



Gambar 1.4 Contoh *Classification Tree* dengan N_{min} = 30. (Timofeev, 2004)

1.1.2. Menghitung *Gini Index* Dataset

Dataset yang masuk harus dihitung tingkat homogenitas dari kelas atributnya dengan menghitung nilai *Gini Index* dari dataset tersebut. Jika nilai *Gini Index* dari dataset mencapai 0, maka dataset sudah mencapai kondisi *terminal node* sehingga tidak perlu melakukan pembelahan dataset.

Sebagai contoh, pada *learning sample* di Tabel 2.1 dapat dilihat dari 14 *tuple*, terdapat 9 *tuple* yang memiliki kelas atribut *Play* dan 5 tuple yang memiliki kelas atribut *Don't Play*. Dari informasi tersebut, dapat dihitung *Gini Index* dari dataset tersebut.

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0{,}459$$

1.1.3. Pencarian Atribut Pembelah

Untuk mencari atribut pembelah yang membelah dataset dengan tingkat homogenitas maksimum, maka setiap atribut dari dataset harus dicari nilai *Gini Index*-nya. Namun atribut diskret dan atribut kontinu mempunyai aturan yang berbeda.

1.1.3.1. Menghitung *Gini Index* Atribut Diskret

Atribut bertipe diskret mengharuskan mencari *Gini Index* dari tiap subset yang dimiliki atribut. Subset yang memiliki *Gini Index* terkecil adalah subset terbaik dan menjadi kandidat untuk subset atribut pembelah dari atribut tersebut.

Sebagai contoh atribut *Outlook* dari Tabel 2.1 mempunyai 3 nilai yaitu {*Sunny, Overcast, Rainy*}. Maka banyaknya kemungkinan subset dari atribut *Outlook* adalah $\frac{2^3-2}{2} = 3$, dan subset-subset yang mungkin adalah {(*Sunny, Overcast*) , (*Rainy*)}, {(*Sunny*) , (*Overcast, Rainy*)}, dan{(*Sunny, Rainy*) , (*Overcast*)}.)}. Dari informasi tersebut dapat dihitung *Gini Index* dari setiap subset tersebut sebagai contoh pada subset {(*Sunny, Overcast*) , (*Rainy*)}, data terbagi dua menjadi partisi D_1 untuk subset $Outlook \in \{Sunny, Overcast\}$ dan partisi D_2 untuk subset $Outlook \in \{Rainy\}$. Terdapat 9 *tuple* yang memenuhi kondisi D_1 dan 5 *tuple* yang memenuhi kondisi D_2. Maka *Gini Index*-nya adalah

$$Gini_{Outlook \in \{Sunny, Overcast\}}(D)$$

$$= \frac{9}{14} Gini(D_1) + \frac{5}{14} Gini(D_2)$$

$$= \frac{9}{14}\left(1 - \left(\frac{6}{9}\right)^2 - \left(\frac{3}{9}\right)^2\right) + \frac{5}{14}\left(1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2\right)$$

$$= \frac{9}{14}\ (0{,}444) + \frac{5}{14}\ (0{,}480)$$

$$= 0{,}285\ + 0{,}171$$

$$= 0{,}457$$

$$= Gini_{Outlook \in \{Rain\}}(D)$$

Gini Index untuk subset $Outlook \in \{Sunny, Rain\}$ dan $Outlook \in \{Rain, Overcast\}$ masing-masing adalah 0,357 dan 0,393. Maka subset terbaik untuk atribut $Outlook$ adalah $\{Sunny, Rain\}$ karena mempunyai *Gini Index* terkecil.

1.1.3.2. Menghitung *Gini Index* Atribut Kontinu

Untuk atribut bertipe kontinu, sebelum menghitung nilai *Gini Index*-nya, maka harus dicari *split-point* dari atribut tersebut. *split-point* tersebut akan menjadi *threshold* untuk perhitungan nilai *Gini Index* atribut. *split-point* didapat dengan mencari nilai tengah dari 2 nilai atribut yang sudah disortir terlebih dahulu. *Split-point* yang menghasilkan *Gini Index* terkecil diantara *split-point* lainnya adalah *split-point* terbaik dan menjadi kandidat atribut pembelah untuk atribut tersebut.

Sebagai contoh pada Tabel 2.1 nilai-nilai dari atribut *Temperature* adalah {85, 80, 83, 70, 68, 65, 64, 72, 69, 75, 75, 72, 81, 71}, lalu setelah disortir nilai-nilai dari atribut *Temperature* akan berubah menjadi seperti ini {64, 65, 68, 69, 70, 71, 72, 72, 75, 75, 80, 81, 83, 85}, maka kemungkinan-kemungkinan *split-point*-nya adalah {64.5, 66.5, 68.5, 69.5, 70.5, 71.5, 72, 73.5, 75, 77.5, 80.5, 82, 84}. Setiap kemungkinan harus dievaluasi. Sebagai contoh, pada kemungkinan *split-point* 64.5, terdapat 1 *tuple* yang memenuhi kondisi *Temperature* ≤ 64.5 dan 13 *tuple* yang memenuhi kondisi *Temperature* > 64.5 maka *Gini Index*-nya adalah

$$Gini_{Temperature \leq 64.5}(D)$$

$$= \frac{1}{14} Gini(D_1) + \frac{13}{14} Gini(D_2)$$

$$= \frac{1}{14}\left(1 - \left(\frac{1}{1}\right)^2 - \left(\frac{0}{1}\right)^2\right) + \frac{13}{14}\left(1 - \left(\frac{8}{13}\right)^2 - \left(\frac{5}{13}\right)^2\right)$$

$$= \frac{1}{14}\,(0) + \frac{13}{14}\,(0{,}473)$$

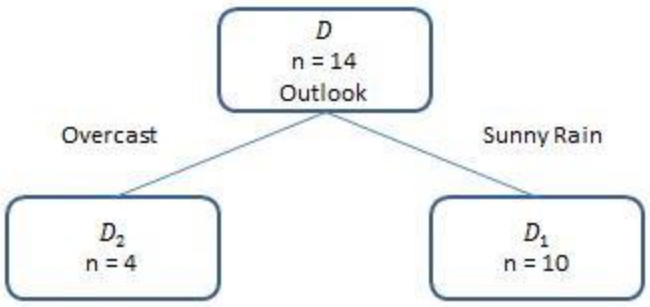
$$= 0{,}439$$

$$= Gini_{Temperature > 64.5}(D)$$

Lalu *Gini Index* dari kemungkinan-kemungkinan *split-point* lainnya adalah {0.452, 0.459, 0.45, 0.432, 0.458, 0.458, 0.458, 0.443, 0.443, 0.459, 0.452, 0.396}. Maka *split-point* dari atribut *Temperature* adalah 84 dengan *Gini Index* 0,396.

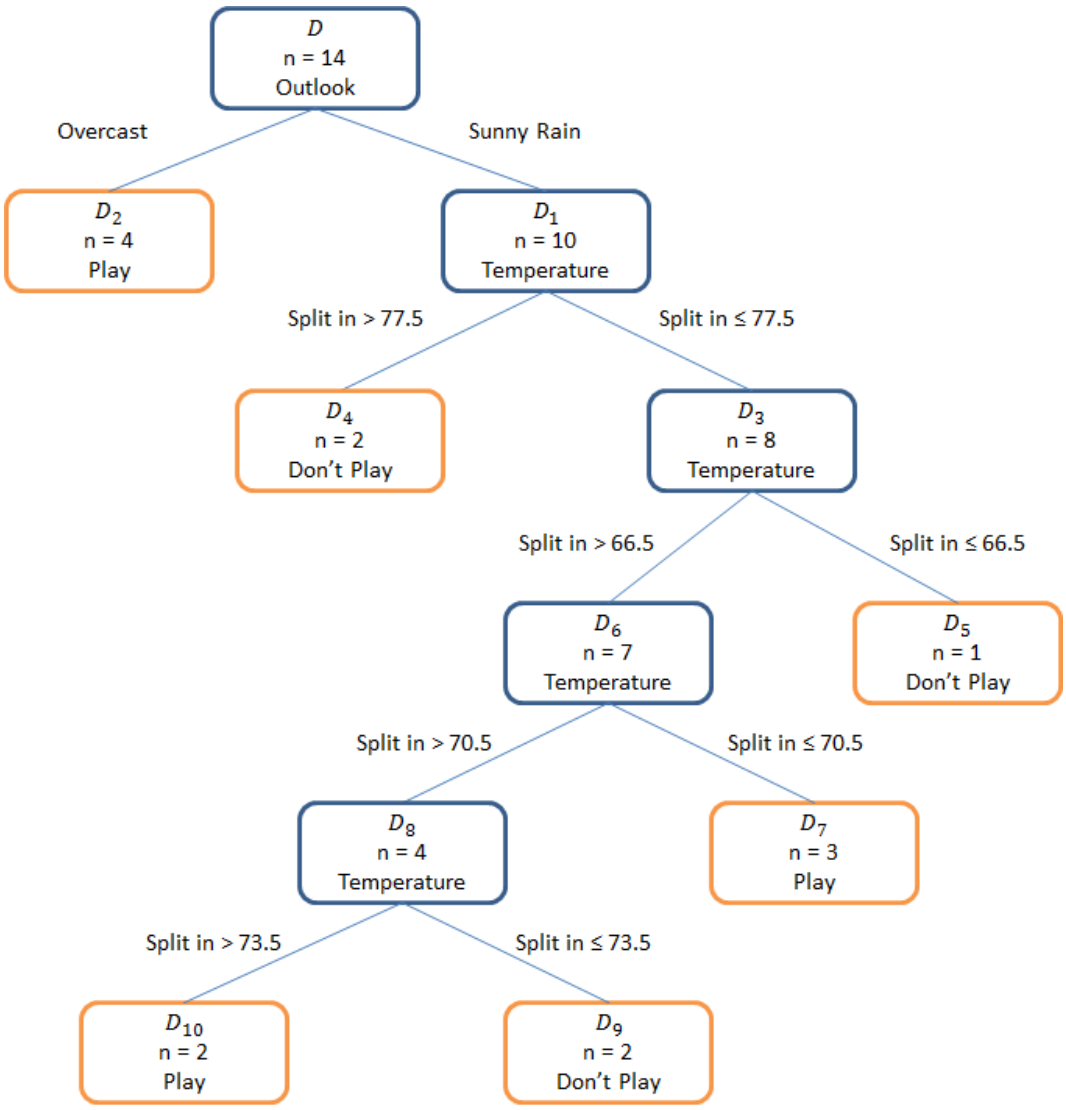
1.1.4. Memilih Atribut Pembelah dan Pembelahan Dataset

Setelah mencari nilai *Gini Index* untuk setiap atribut, maka atribut yang memiliki *Gini Index* terkecil dipilih menjadi atribut pembelah. Sebagai contoh pada Tabel 3.1, untuk atribut *Outlook*, subsetnya adalah $\{Sunny, Rain\}$ dengan *Gini Index* 0,357. Untuk atribut *Temperature*, *split-point*-nya adalah 84 dengan *Gini Index* 0,396. Untuk atribut *Humidity*, *split-point*-nya adalah 82.5 dengan *Gini Index* 0,394. Dan untuk atribut *Windy*, *Gini Index*-nya adalah 0,429. Jadi, atribut pembelah dari dataset adalah *Temperature* dengan subset $\{Sunny, Rain\}$. Maka bentuk *Classification Tree* dari partisi data tersebut dapat dilihat pada Gambar 3.3



Gambar 1.1 Partisi data pertama pada dataset *D*

Setelah dataset dibelah menjadi partisi D_1 dan partisi D_2, maka tiap partisi dihitung nilai *Gini Index*-nya serta dicek jumlah *tuple*-nya. Jika nilai *Gini Index*-nya adalah 0 atau jumlah tuple sudah mencapai N_{min}, maka partisi sudah mencapai kondisi *terminal node*. Jika tidak, maka proses pembelahan akan diulangi sampai mencapai *terminal node*. Gambar 3.4 adalah ilustrasi dari *Classification Tree* yang dibentuk dari dataset pada Tabel 2.1



Gambar 1.2 Contoh *Classification Tree*

1.2. Pengetesan *Classification Tree*

Setelah *Classification Tree* dibentuk, maka hal yang harus dilakukan selanjutnya adalah mengetes akurasi dari *Classification Tree* tersebut. Proses pengetesan dilakukan dengan mengklasifikasi *test set* dengan mengikuti pola dari *Classification Tree*. Akurasi dihitung berdasarkan banyaknya atribut kelas yang tidak sesuai dengan atribut kelas pada *terminal node* dari *Classification*