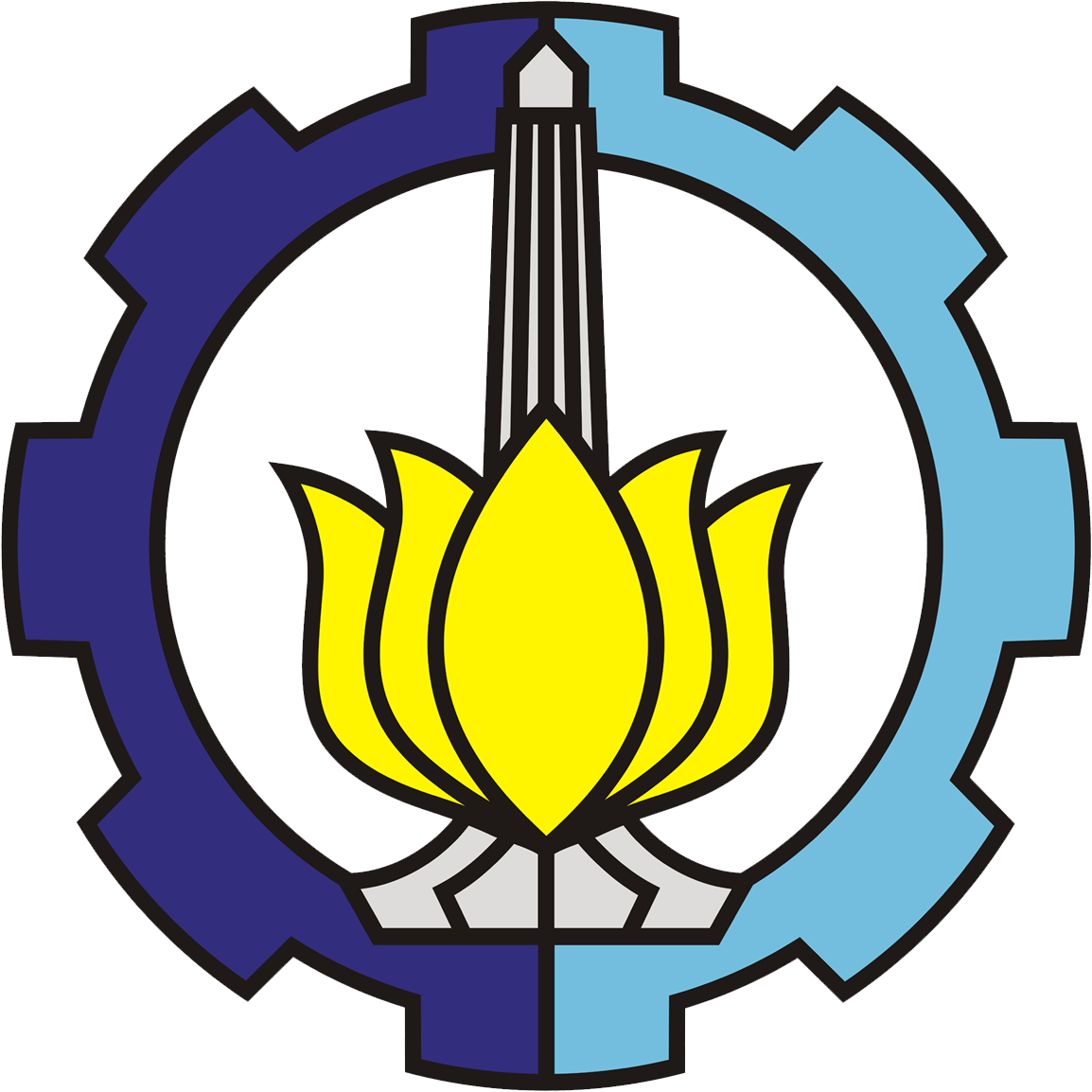
**Laporan Tugas Kecerdasan Komputasional**



**Disusun Oleh :**

**Rahadian K Putra *05111640000006***

**Alcredo Simanjuntak *05111640000045***

**Alfian *05111640000073***

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**SURABAYA**

**2018**

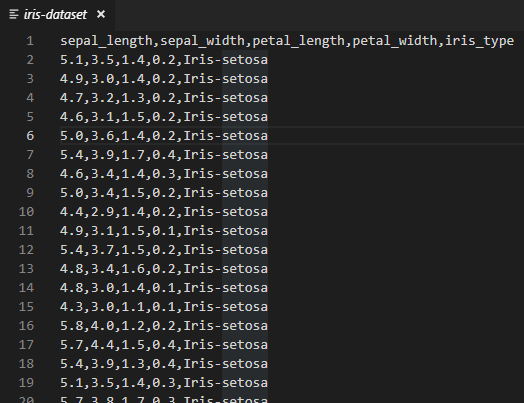
**Dataset**

Ada dua jenis dataset yang digunakan dalam codingan tersebut, yaitu iris-dataset(numerical) dan survey-dataset(categorical). Tujuan penggunaan kedua dataset tersebut agar codingan lebih flexibel terhadap data uji.

1. Iris-Dataset

Data yang digunakan untuk mengklasifikasikan iris kedalam tiga kelas yaitu Iris-setosa, Iris-versicolor, Iris-virginica. Dataset ini berjumlah 150 data dengan properties sebagai berikut :

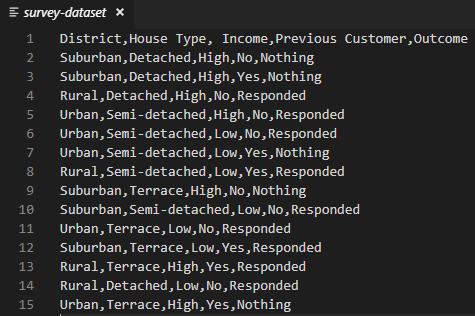
* Sepal Length
* Sepal Width
* Petal Lenght
* Petal Width



1. Survey-Dataset

Data yang digunakan untuk menentukan keputusan(outcome) yaitu Responded atau Nothing. Dataset ini berjumlah 14 data dengan properties sebagai berikut :

* District
* House Type
* Income
* Previous Customer



**Metode**

Dalam hal ini kami menggunakan Decision Tree untuk melakukan klasfikasi data uji. Decision Tree terdiri atas 3 bagian yaitu :

* Simpul root yaitu titik awal dari sebuah tree
* Simpul perantara yang berhubungan dengan sebuah pertanyaan atau pengujian
* Simput leaf yaitu titik akhir/sebuah keputusan akhir setelah beberapa kali pengujian

Split

Di dalam decision tree terdapat 2 jenis split yaitu :

* Binary Split

Jumlah child pada suatu node lebih kecil atau sama dengan 2

* Multi-Way Split

Jumlah child pada suatu node lebih besar dari 2

Untuk mengimplementasikan codingan ini, digunakan Binary Split.

Node Impurity

Untuk menentukan split terbaik untuk setiap node ada 3 metode yaitu :

* Gini Index
* Entropy
* Misclassification Error

Untuk mengimplementasikan codingan ini, digunakan dengan perhitungan Gini Index. Dengan rumus sebagai berikut :

Gini Index



Gini Split



**Optimasi**

Optimasi Algoritma Gini Split untuk data numeric dengan prefix-sum

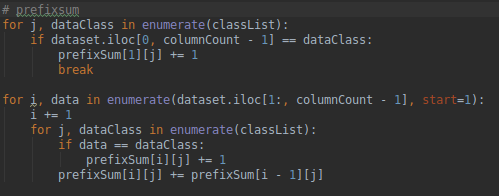
Untuk menemunkan index dimana hasil splitnya menghasilkan Gini Index terendah, maka dilakukan pengecekan berulang-ulang untuk menemukan jumlah data yang lebih kecil atau sama dengan x, dimana x adalah rata-rata dari (data[i] + data[i+1])/2, dengan i ≤ n-2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sorted Sepal Width** | | | | | | | | | | | | |
| **Index ⟶** | | **0** | **1** | **2** | **15** | **16** | **17** | **30** | **31** | **32** | **...** | **n** |
| **Number of Iris** | **Sepal Width ⟶** | Sw ≤ 2.0 | Sw ≤ 2.2 | Sw ≤ 2.3 | Sw ≤ 2.5 | Sw ≤ 2.6 | Sw ≤ 2.7 | Sw ≤ 3.0 | Sw ≤ 3.1 | Sw ≤ 3.2 |  | Sw ≤ 4.4 |
| **Iris-sentosa** | 0 | 0 | 1 | 1 | 1 | 1 | 7 | 12 | 17 |  | 50 |
| **Iris-virginica** | 0 | 2 | 5 | 13 | 16 | 20 | 42 | 45 | 48 |  | 50 |
| **Iris-versicolor** | 1 | 1 | 1 | 4 | 6 | 11 | 33 | 37 | 42 |  | 50 |

*NB : Sw adalah Sepal Width*

Agar komputasi lebih efisien, maka digunakan metode prefix-sum dengan cara mensort data terlebih dahulu terhadap column yang akan dihitung gini splitnya, Lalu dilakukan prefixsum terhadap class hasil (iris-sentosa, iris-virginica, iris versi-color).

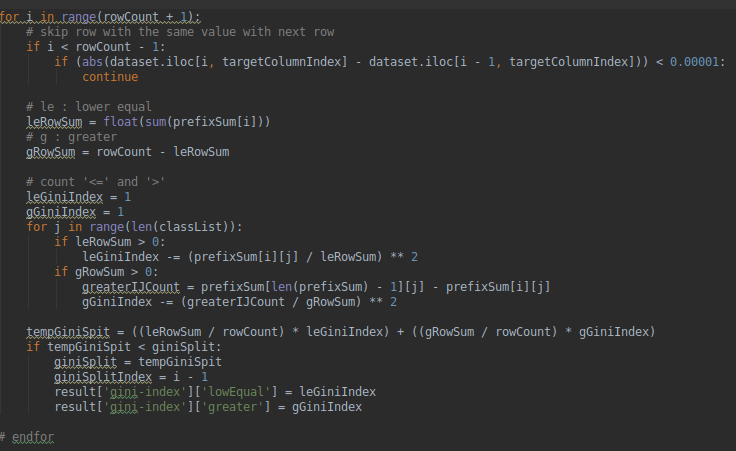
*Gambar : Prefixsum*



Data dengan Prefixsum

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sorted data by Sepal Width** | | | | | | | | | | | | | | |
| **Prefix Sum of each Iris type** | **Data Index →** | **0** | **1** | **2** | **...** | **15** | **16** | **17** | **...** | **30** | **31** | **32** | **...** | **150** |
| Iris-sentosa | 0 | 0 | 0 |  | 1 | 1 | 1 |  | 1 | 1 | 1 |  | 50 |
| Iris-versicolor | 0 | 1 | 2 |  | 11 | 11 | 12 |  | 20 | 20 | 20 |  | 50 |
| Iris-virginica | 0 | 0 | 0 |  | 2 | 4 | 4 |  | 9 | 10 | 11 |  | 50 |

*Gambar : Penerapan prefix-sum untuk menghitung gini index*



**Uji Coba**

|  |  |  |  |
| --- | --- | --- | --- |
| **Train Percentage (%)** | **Iteration** | **Error (%)** | **Running Time(seconds)** |
| **70** | **1** | 4,44 | 0,109840393 |
| **2** | 2,22 | 0,136533737 |
| **3** | 2,22 | 0,111982107 |
| **4** | 2,22 | 0,112685442 |
| **5** | 2,22 | 0,115119219 |
| **Average** | | **2,664** | **0,11723218** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Train Percentage (%)** | **Iteration** | **Error (%)** | **Running Time(seconds)** |
| **80** | **1** | 3,33 | 0,127348423 |
| **2** | 13,33 | 0,154061317 |
| **3** | 6,66 | 0,103788853 |
| **4** | 6,66 | 0,11050272 |
| **5** | 0 | 0,153044462 |
| **Average** | | **5,996** | **0,129749155** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Train Percentage (%)** | **Iteration** | **Error (%)** | **Running Time(seconds)** |
| **90** | **1** | 0.0 | 0,139310122 |
| **2** | 6,66 | 0,109426498 |
| **3** | 13,33 | 0,114414692 |
| **4** | 6,66 | 0,115596533 |
| **5** | 0.0 | 0,115361452 |
| **Average** | | **5,33** | **0,118821859** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Train Percentage (%)** | **Iteration** | **Error (%)** | **Running Time(seconds)** |
| **95** | **1** | 0.0 | 0,100138903 |
| **2** | 12,5 | 0,08749485 |
| **3** | 12,5 | 0,089473724 |
| **4** | 0.0 | 0,112467766 |
| **5** | 12,5 | 0,087864876 |
| **Average** | | **7,5** | **0,095488024** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Train Percentage (%)** | **Iteration** | **Error (%)** | **Running Time(seconds)** |
| **100** | **1** | 0.0 | 0,161643505 |
| **2** | 0.0 | 0,15798831 |
| **3** | 0.0 | 0,169991016 |
| **4** | 0.0 | 0,155905962 |
| **5** | 0.0 | 0,158593178 |
| **Average** | | **0** | **0,160824394** |