CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 Implementation

This project is implemented with MySql. In this sub-chapter will explain how to use the program and how the program works.

```
1. Select @jumlahdata:=count(*)
2. from tblData;
3.
4. select @ratingLow:=count(*)
5. from tblData
6. where ratingFix LIKE ('%LR%');
7.
8. select @ratingHigh:=count(*)
9. from tblData
10.  where ratingFix LIKE ('%HR%');
11.
12.  Select @nilaiI:=(-
   (@ratingLow/@jumlahdata)*log2(@ratingLow/@jumlahdata))
13.  +
14.  (-(@ratingHigh/@jumlahdata)*log2(@ratingHigh/@jumlahdata));
```

Codes above is the first step to calculate iterations after data is entered, Line 1 and 2 are used to calculate the amount of incoming data, line 4-5 are used to calculate the amount of data that is Low Rating or abbreviated as LR, line 8-10 are used to calculate the amount of data that is High Rating or abbreviated as HR. Then line 12-14 to calculate the gain value from the amount of data.

```
15.  Insert into tblHitung
16.    (informasi, jumlahdata, ratingLow, ratingHigh)
17.    select distinct(A.branch) as BRANCH, count(A.branch) as JUMLAHDATA,
18.    (
19.      select COUNT(*)
20.      from tblData as B
21.      where B.ratingFix LIKE ('%LR%') and
22.      B.branch = A.branch
23.    )AS RATINGLOW,
24.    (
25.      select COUNT(*)
26.      from tblData as C
27.      where C.ratingFix LIKE ('%HR%') and
28.      C.branch = A.branch
29.    )as RATINGHIGH
30.  from tblData as A
31.  group by A.branch;
```

At line 15 to 31 are used to calculate the number of LR and HR for each existing attribute, then entered into the table. This process is repeated for each attribute.

```
32.  Update tblHitung set nilaiI =
33.   (-(ratingLow/jumlahdata) *log2(ratingLow/jumlahdata))
34.   +(-(ratingHigh/jumlahdata) *log2(ratingHigh/jumlahdata));
35.
36.   update tblHitung set nilaiI = 0
37.   where nilaiI is NULL;
38.
39.   insert into tblTampung(atribut, gain)
40.   select atribut, @nilaiI - SUM((jumlahdata/@jumlahdata)*nilaiI) as HITUNGGAIN
41.     from tblHitung
42.     group by atribut;
43.
44.     update tblHitung set gain =
45.     ROUND(
46.     (
47.       select tblTampung.gain
48.       from tblTampung
49.       where tblTampung.atribut = tblHitung.atribut
50.     ),4);
```

At line 32 to 35 are used to calculate Entropy, line 37 and 38 used to set Entropy 0 if the value is null. Line 40 to 51 used to calculate gain for each atribute and then insert to table tblHitung.

```
51.  Create table tblData2 as
52.  select id from tblData where branch = @tampungInformasi;
53.
54.  delete from tblData
55.  where id not in (select id from tblData2);
```

After getting the gain value for each attribute and selecting the attribute with the highest gain value, line 52 to 56 the next step is to delete data that does not have the attribute with the highest gain value to proceed to the next iteration. The steps above are repeated until the last iteration.

```
56.  If (@tampungTotalData = 2) then
57.  LEAVE looping;
58.  end if;
59.  end while looping;
```

At line 57 to 60 if there are 2 remaining data then the looping process will stop.

5.2 Testing

In the C-45 algorithm, after calculating all the data and getting the results from each iteration, the next step is to make a decision tree to get the final conclusion.

Table 1.10 : Total Iteration 700 data

Iteration	Information
1	Cogs
2	Product Line
3	qty

The table above is the number of iterations obtained from a total of 700 data, and in the information column is the highest gain in each iteration. Based on results of application above, it will produce a decision tree as below.

To calculate the accuracy of the original data with predictive data using the confusion matrix.

Table 1.11 : Confusion Matrix First Data

	Predicted : High Rating	**Predicted : Low Rating**
Actual : High Rating	TP=96	FN=20
Actual : Low Rating	FP=129	TN=25

Information :

TP (True Positive) : When the product is predicted High Rating, It is High Rating.

TN (True Negative) : When the product is predicted Low Rating, It is Low Rating.

FP (False Positive) : When the product predicted High Rating, but it is Low Rating.

FN (False Negative) : When the product predicted Low Rating, but it is High Rating.

Formula :

$$Precision = \frac{TP}{TP+FP} = \frac{96}{96+12} = 0{,}889 = 88{,}9\%$$

$$Recall = \frac{TP}{TP+FN} = \frac{96}{96+20} = 0{,}827 = 82{,}7\%$$

$$Accuracy = \frac{TP+TN}{Total} = \frac{96+25}{153} = 0{,}791 = 79{,}1\%$$

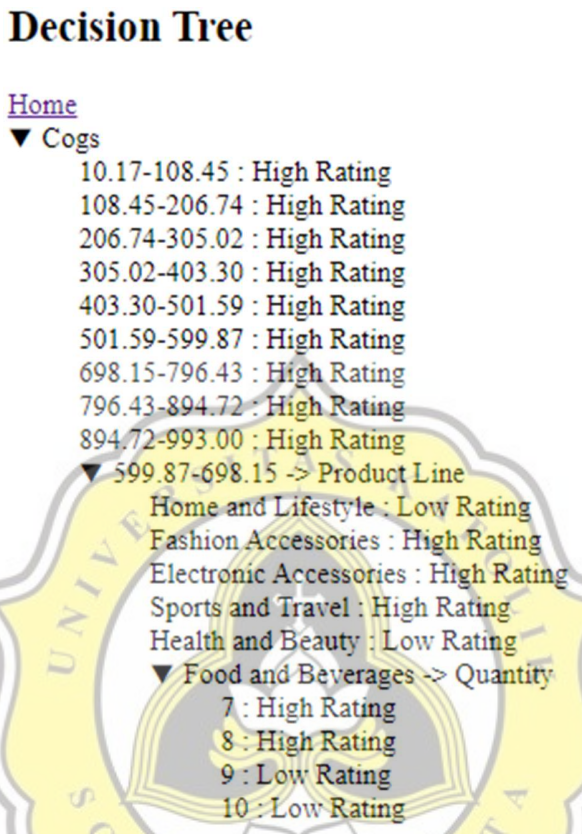

Figure 1.3. Decision Tree using Tree View
First Data

After calculating the training data using 700 data, then compare the training data using 300 new data to show whether the results of the previous data calculations are correct.

Table 1.12 : Data Testing 300 data

Atribut	Informasi	Jumlah	Rating Low	Rating High
Cogs	10.17-108.45	84	19	65
	108.45-206.74	50	6	44

	206.74-305.02	46	4	42
	305.02-403.30	30	8	22
	403.30-501.59	26	7	19
	501.59-599.87	18	2	16
	698.15-796.43	21	5	16
	796.43-894.72	8	1	7
	894.72-993.00	3	2	1
Product Line	Home and Lifestyle	51	10	41
	Fashion Accessories	52	12	40
	Electronic Accessories	58	8	50
	Sports and Travel	39	6	33
	Health and Beauty	51	10	41
Qty	7	37	7	30
	8	34	7	27
	9	25	5	20
	10	32	8	24

after calculating the testing data, from 300 data only information Cogs 894.72-993.00, Product Line Home and Lifestyle, Health and Beauty and Qty 9, 10.