A Step By Step C4.5 Decision Tree Example

May 13, 2018 / Machine Learning

Decision trees are still hot topics nowadays in data science world. Here, ID3 is the most common conventional decision tree algorithm but it has bottlenecks. Attributes must be nominal values, dataset must not include missing data, and finally the algorithm tend to fall into overfitting. Here, Ross Quinlan, inventor of ID3, made some improvements for these bottlenecks and created a new algorithm named C4.5. Now, the algorithm can create a more generalized models including continuous data and could handle missing data. Additionally, some resources such as Weka named this algorithm as J48. Actually, it refers to re-implementation of C4.5 release 8.

Groot appears in Guardians of Galaxy and Avengers Infinity War

We are going to create a decision table for the following dataset. It informs about decision making factors to play tennis at outside for previous 14 days. The dataset might be familiar from the ID3 post. The difference is that temperature and humidity columns have continuous values instead of nominal ones.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes

13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

We will do what we have done in ID3 example. Firstly, we need to calculate global entropy. There are 14 examples; 9 instances refer to yes decision, and 5 instances refer to no decision.

Entropy(Decision) = $\sum - p(I) . \log_2 p(I) = - p(Yes) . \log_2 p(Yes) - p(No) . \log_2 p(No) = - (9/14) . \log_2(9/14) - (5/14) . \log_2(5/14) = 0.940$

In ID3 algorithm, we've calculated gains for each attribute. Here, we need to calculate gain ratios instead of gains.

GainRatio(A) = Gain(A) / SplitInfo(A)

SplitInfo(A) = $-\sum |Dj|/|D| \times \log_2 |Dj|/|D|$

Wind Attribute

Wind is a nominal attribute. Its possible values are weak and strong.

Gain(Decision, Wind) = Entropy(Decision) – ∑ (p(Decision|Wind) . Entropy(Decision|Wind))

Gain(Decision, Wind) = Entropy(Decision) – [p(Decision|Wind=Weak) . Entropy(Decision|Wind=Weak)] + [p(Decision|Wind=Strong) . Entropy(Decision|Wind=Strong)]

There are 8 weak wind instances. 2 of them are concluded as no, 6 of them are concluded as yes.

Entropy(Decision|Wind=Weak) = $- p(No) . \log_2 p(No) - p(Yes) . \log_2 p(Yes) = - (2/8) . \log_2(2/8) - (6/8) . \log_2(6/8) = 0.811$

Entropy(Decision|Wind=Strong) = $- (3/6) . \log_2(3/6) - (3/6) . \log_2(3/6) = 1$

Gain(Decision, Wind) = 0.940 – (8/14).(0.811) – (6/14).(1) = 0.940 – 0.463 – 0.428 = 0.049

There are 8 decisions for weak wind, and 6 decisions for strong wind.

SplitInfo(Decision, Wind) = $-(8/14).\log_2(8/14) - (6/14).\log_2(6/14) = 0.461 + 0.524 = 0.985$

GainRatio(Decision, Wind) = Gain(Decision, Wind) / SplitInfo(Decision, Wind) = 0.049 / 0.985 = 0.049

Outlook Attribute

Outlook is a nominal attribute, too. Its possible values are sunny, overcast and rain.

Gain(Decision, Outlook) = Entropy(Decision) – ∑ (p(Decision|Outlook) . Entropy(Decision|Outlook)) =

Gain(Decision, Outlook) = Entropy(Decision) – p(Decision|Outlook=Sunny) . Entropy(Decision|Outlook=Sunny) – p(Decision|Outlook=Overcast) . Entropy(Decision|Outlook=Overcast) – p(Decision|Outlook=Rain) . Entropy(Decision|Outlook=Rain)

There are 5 sunny instances. 3 of them are concluded as no, 2 of them are concluded as yes.

Entropy(Decision|Outlook=Sunny) = – p(No) . log$_2$p(No) – p(Yes) . log$_2$p(Yes) = -(3/5).log$_2$(3/5) – (2/5).log$_2$(2/5) = 0.441 + 0.528 = 0.970

Entropy(Decision|Outlook=Overcast) = – p(No) . log$_2$p(No) – p(Yes) . log$_2$p(Yes) = -(0/4).log$_2$(0/4) – (4/4).log$_2$(4/4) = 0

> *Notice that log$_2$(0) is actually equal to -∞ but assume that it is equal to 0. Actually, lim (x->0) x.log$_2$(x) = 0. If you wonder the proof, please look at this post.*

Entropy(Decision|Outlook=Rain) = – p(No) . log$_2$p(No) – p(Yes) . log$_2$p(Yes) = -(2/5).log$_2$(2/5) – (3/5).log$_2$(3/5) = 0.528 + 0.441 = 0.970

Gain(Decision, Outlook) = 0.940 – (5/14).(0.970) – (4/14).(0) – (5/14).(0.970) – (5/14).(0.970) = 0.246

There are 5 instances for sunny, 4 instances for overcast and 5 instances for rain

SplitInfo(Decision, Outlook) = -(5/14).log$_2$(5/14) -(4/14).log$_2$(4/14) -(5/14).log$_2$(5/14) = 1.577

GainRatio(Decision, Outlook) = Gain(Decision, Outlook)/SplitInfo(Decision, Outlook) = 0.246/1.577 = 0.155

Humidity Attribute

As an exception, humidity is a continuous attribute. We need to convert continuous values to nominal ones. C4.5 proposes to perform binary split based on a threshold value. Threshold should be a value which offers maximum gain for that attribute. Let's focus on humidity attribute. Firstly, we need to sort humidity values smallest to largest.

Day	Humidity	Decision
7	65	Yes
6	70	No
9	70	Yes

11	70	Yes
13	75	Yes
3	78	Yes
5	80	Yes
10	80	Yes
14	80	No
1	85	No
2	90	No
12	90	Yes
8	95	No
4	96	Yes

Now, we need to iterate on all humidity values and seperate dataset into two parts as instances less than or equal to current value, and instances greater than the current value. We would calculate the gain or gain ratio for every step. The value which maximizes the gain would be the threshold.

Check 65 as a threshold for humidity

Entropy(Decision|Humidity<=65) = – p(No) . log$_2$p(No) – p(Yes) . log$_2$p(Yes) = -(0/1).log$_2$(0/1) – (1/1).log$_2$(1/1) = 0

Entropy(Decision|Humidity>65) = -(5/13).log$_2$(5/13) – (8/13).log$_2$(8/13) =0.530 + 0.431 = 0.961

Gain(Decision, Humidity<> 65) = 0.940 – (1/14).0 – (13/14).(0.961) = 0.048

** The statement above refers to that what would branch of decision tree be for less than or equal to 65, and greater than 65. It **does not** refer to that humidity is not equal to 65!*

SplitInfo(Decision, Humidity<> 65) = -(1/14).log$_2$(1/14) -(13/14).log$_2$(13/14) = 0.371

GainRatio(Decision, Humidity<> 65) = 0.126

Check 70 as a threshold for humidity

Entropy(Decision|Humidity<=70) = – (1/4).$\log_2$(1/4) – (3/4).$\log_2$(3/4) = 0.811

Entropy(Decision|Humidity>70) = – (4/10).$\log_2$(4/10) – (6/10).$\log_2$(6/10) = 0.970

Gain(Decision, Humidity<> 70) = 0.940 – (4/14).(0.811) – (10/14).(0.970) = 0.940 – 0.231 – 0.692 = 0.014

SplitInfo(Decision, Humidity<> 70) = -(4/14).$\log_2$(4/14) -(10/14).$\log_2$(10/14) = 0.863

GainRatio(Decision, Humidity<> 70) = 0.016

Check 75 as a threshold for humidity

Entropy(Decision|Humidity<=75) = – (1/5).$\log_2$(1/5) – (4/5).$\log_2$(4/5) = 0.721

Entropy(Decision|Humidity>75) = – (4/9).$\log_2$(4/9) – (5/9).$\log_2$(5/9) = 0.991

Gain(Decision, Humidity<> 75) = 0.940 – (5/14).(0.721) – (9/14).(0.991) = 0.940 – 0.2575 – 0.637 = 0.045

SplitInfo(Decision, Humidity<> 75) = -(5/14).$\log_2$(4/14) -(9/14).$\log_2$(10/14) = 0.940

GainRatio(Decision, Humidity<> 75) = 0.047

I think calculation demonstrations are enough. Now, I skip the calculations and write only results.

Gain(Decision, Outlook <> 78) =0.090, GainRatio(Decision, Humidity<> 78) =0.090

Gain(Decision, Outlook <> 80) = 0.101, GainRatio(Decision, Humidity<> 80) = 0.107

Gain(Decision, Outlook <> 85) = 0.024, GainRatio(Decision, Humidity<> 85) = 0.027

Gain(Decision, Outlook <> 90) = 0.010, GainRatio(Decision, Humidity<> 90) = 0.016

As seen, gain maximizes when threshold is equal to 80 for humidity. This means that we need to compare other nominal attributes and comparison of humidity to 80 to create a branch in our tree.

Let's summarize calculated gain and gain ratios. Outlook attribute comes with both maximized gain and gain ratio. This means that we need to put outlook decision in root of decision tree.

Attribute	Gain	GainRatio
Wind	0.049	0.049

Outlook	0.246	0.155
Humidity <> 80	0.101	0.107

After then, we would apply similar steps just like as ID3 and create following decision tree. Outlook is put into root node. Now, we should look decisions for different outlook types.

Outlook = Sunny

We've split humidity for greater than 80, and less than or equal to 80. Surprisingly, decisions would be no if humidity is greater than 80 when outlook is sunny. Similarly, decision would be yes if humidity is less than or equal to 80 for sunny outlook.

Day	Outlook	Temp.	Hum. > 80	Wind	Decision
1	Sunny	85	Yes	Weak	No
2	Sunny	80	Yes	Strong	No
8	Sunny	72	Yes	Weak	No
9	Sunny	69	No	Weak	Yes
11	Sunny	75	No	Strong	Yes

Outlook = Overcast

If outlook is overcast, then no matter temperature, humidity or wind are, decision will always be yes.

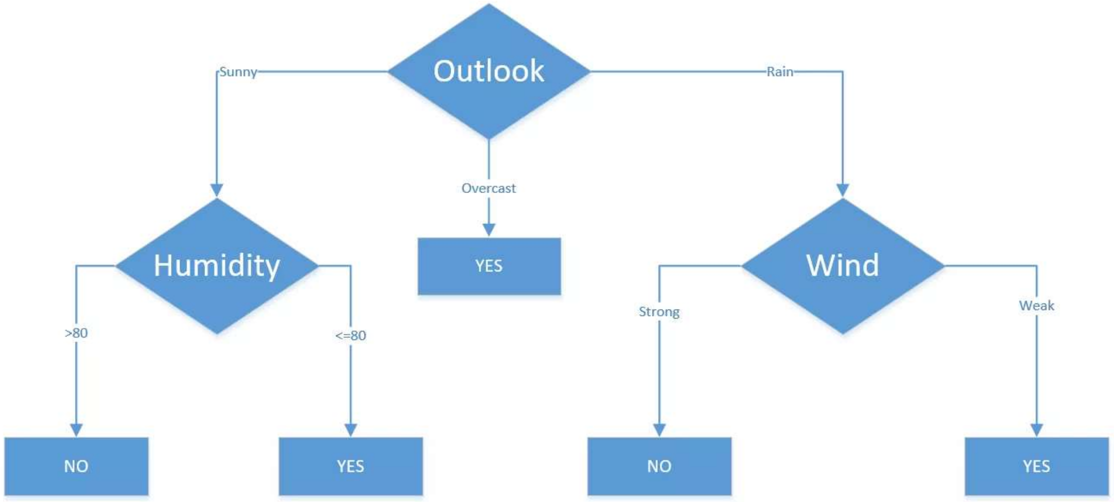
Day	Outlook	Temp.	Hum. > 80	Wind	Decision
3	Overcast	83	No	Weak	Yes
7	Overcast	64	No	Strong	Yes
12	Overcast	72	Yes	Strong	Yes
13	Overcast	81	No	Weak	Yes

Outlook = Rain

We've just filtered rain outlook instances. As seen, decision would be yes when wind is weak, and it would be no if wind is strong.

Day	Outlook	Temp.	Hum. > 80	Wind	Decision
4	Rain	70	Yes	Weak	Yes
5	Rain	68	No	Weak	Yes
6	Rain	65	No	Strong	No
10	Rain	75	No	Weak	Yes
14	Rain	71	No	Strong	No

Final form of decision table is demonstrated below.



Decision tree generated by C4.5

So, C4.5 algorithm solves most of problems in ID3. The algorithm uses gain ratios instead of gains. In this way, it creates more generalized trees and not to fall into overfitting. Moreover, the algorithm transforms continuous attributes to nominal ones based on gain maximization and in this way it can handle continuous data. Additionally, it can ignore instances including missing data and handle missing dataset. On the other hand, both ID3 and C4.5 requires high CPU and memory demand. Besides, most of authorities think decision tree algorithms in data mining field instead of machine learning.

Share this:

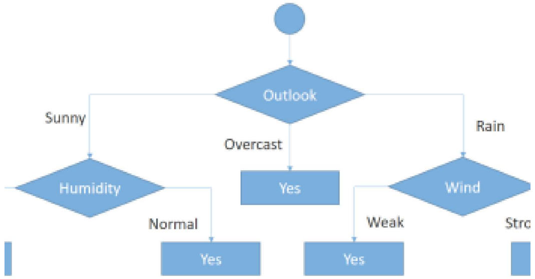
2

Like this:

Like

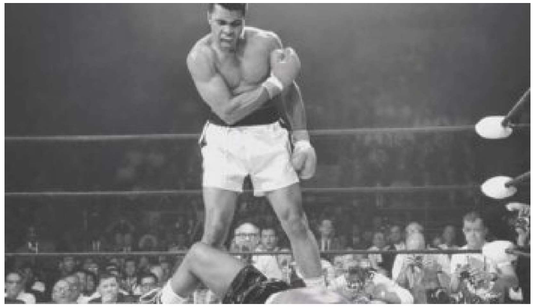
One blogger likes this.

Related



A Step by Step CART Decision Tree Example

A Step by Step ID3 Decision Tree Example



10 Interview Questions Asked in Machine Learning

#c4.5, #decision tree, #gradient boosted, #id3

« Previous / Next »

15 Comments

Dinca Andrei

June 15, 2018 at 8:31 am

GainRatio(Decision, Humidity 65) = 0.126
GainRatio(Decision, Humidity 80) = 0.107

How comes you took into account a threshold of 80 when GainRatio for 65 is higher?

□ Reply

□ **Sefik Serengil**
June 15, 2018 at 9:00 am

In case of Humidity< =80, there are 2 no and 7 yes decisions. Total number of instances is 9
Entropy(Decision|Humidity< =80) = – p(No) . log2p(No) – p(Yes) . log2p(Yes) = - (2/9) * log(2/9) - (7/9)*log(7/9) = 0.764 (BTW, log refers to the base 2)
In case of Humidity>80, there are 3 no and 2 yes decisions. Total number of instances is 5
Entropy(Decision|Humidity>80) = – p(No) . log2p(No) – p(Yes) . log2p(Yes) = -(3/5)*log(3/5) – (2/5)*log(2/5) = 0.971

Global entropy was calculated as 0.940 in previous steps

Now, it is time to calculate Gain.
Gain(Decision, Humidity<> 80) = Entropy(Decision) – p(Humidity< =80) * Entropy(Decision|Humidity<=80) - p(Humidity>80)*Entropy(Decision|Humidity>80)
Gain(Decision, Humidity<> 80) = 0.940 – (9/14)*0.764 – (5/14)*0.971 = 0.101

Now, we can calculate GainRatio but before we need to calculate SplitInfo first.

SplitInfo(Decision, Humidity<> 80) = -p(No)*log(p(No)) – p(Yes)*log(P(Yes)) = -(9/14)*log(9/14) – (5/14)*log(5/14) = 0.940

GainRatio = Gain / SplitInfo = 0.101 / 0.940 = 0.107

I hope this explanation is understandable.

□ Reply

□ **Dinca Andrei**
June 15, 2018 at 9:13 am

Understood, thanks

□ Reply

eddie reader

July 19, 2018 at 10:55 am

You have the following
Gain(Decision, Humidity 70) = 0.940 – (4/14).(0.811) – (10/14).(0.970) = 0.940 – 0.231 – 0.692 = 0.014

there are 3 values of 70 so surely P(Humidity 70) = P(No) = 3/14 not 4/14

Reply

Sefik Serengil
July 19, 2018 at 12:37 pm

Here, 4 is the number of instances which are less than or equal to 70. Humidity of instances for day 6, 7, 9, 11 are less than or equal to 70.

Similarly, 10 is the number of instances which are greater than 70. Number of instances greater than 70 is 10.

Reply

eddie reader
July 19, 2018 at 3:12 pm

I’m sorry I posted the wrong reference. It should have been
Gain(Decision, Humidity 70) = 0.940 – (4/14).(0.811) – (10/14).(0.970) = 0.940 – 0.231 – 0.692 = 0.014

so,
there are 3 values of 70 so surely P(Humidity 70) = P(No) = 3/14 not 4/14
thanks for your attention

Reply

Sefik Serengil
July 19, 2018 at 3:21 pm

You were right if we need Gain(Decision, Humidity = 70) but we need Gain(Decision, Humidity <= 70). Gain(Decision, Humidity ? 70) = 0.940 – (4/14).(0.811) – (10/14).(0.970) = 0.940 – 0.231 – 0.692 = 0.014 In this equation 4/14 is probability of instances less than or equal to 70, and 10/14 is probability of instances greater than 70.

Reply

eddie reader
July 19, 2018 at 3:34 pm

I do understand that but the statement is ‘not equal’ to 70, not less than or equal. If the objective is to have values less than or equal and values greater than then the calculation is that of the global entropy, i.e. all values surely.

Reply

Sefik Serengil
July 19, 2018 at 3:50 pm

Right, cause of poor communication. Actually, I would not intent as your understanding. I should mention that in the post.

The statement Gain(Decision, Humidity ? 70) refers to that what would be if the branch of decision tree is for less than or equal to 70, and greater than 70. All calculations made with this approach.

I hope everything is clear now. Thank you for your attention.

Reply

eddie reader
July 19, 2018 at 5:44 pm

OK, now I get it. Thanks a lot both for your blog, attention and patience. It is much appreciated.

eddie reader
July 19, 2018 at 3:28 pm

Looks as if the editor loses the not-equal sign, hence the poor communication..
The expression in question is
Gain(Decision, Humidity¬= 70) = 0.940 – (4/14).(0.811) – (10/14).(0.970) = 0.940 – 0.231 – 0.692 = 0.014

There being 3 instances of 70 so 3 instances where P(Humidity¬=70) is false, i.e. P(No) = 3/14

Reply

eddie reader
July 22, 2018 at 11:02 am

I’m sorry to have to return to the point made by Dinca Andrei but I think the confusion arises from a statement in your blog
The value which maximizes the gain would be the threshold.

Is it not the case that the threshold is the value that minimises Entropy(Decision|Humidity threshold)
Here are some calculations, which if taken with the ones you perform, does show 80 as the splitting point – Entropy(Decision|Humidity 80) is the least value

le 65 0
> 65 0.961237
65 0.892577
g ratio 0.047423
split 0.371232
g ratio 0.127745

le 78 0.650022
>78 1
78 0.85001
g 0.08999
split 0.985228
g ratio 0.09134

le 80 0.764205
>80 0.970951
80 0.838042
g 0.101958
spli 0.940286
g ratio 0.108433

le 85 0.881291
>85 1
85 0.915208
g 0.024792
split 0.863121
g ratio 0.028724

Thanks for your attention.

□ Reply

Sefik Serengil
July 22, 2018 at 1:36 pm

Yes, you are absolutely right. I summarized gain and gain ratios for every possible threshold point.

branch-> 65 70 75 78 80 85 90
gain 0.048 0.014 0.045 0.09 0.101 0.024 0.01
gain ratio 0.126 0.016 0.047 0.09 0.107 0.027 0.016

I stated that “We would calculate the gain OR gain ratio for every step. The value which maximizes the gain would be the threshold.”. Now, it is all up to you to decide threshold point based on gain or gain ratio. If prefer to use gain and my threshold would be 80. If you prefer to use gain ratio metric, your threshold would be 65. The both approaches are correct.

Reply

---

mzn
August 11, 2018 at 2:45 pm

HI ,, are this algorithm good for a large database , like a dataset for large manufacture
Thank you

Reply

Sefik Serengil
August 11, 2018 at 2:48 pm

Decision tree algorithms require high memory demand. You should look its extended version – random forests, this might be adapted better for your problem

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

ex: jane doe

Email *

ex: janedoe@gmail.com

Website

ex: http://janedoe.wordpress.com

□ Notify me of follow-up comments by email.

□ Notify me of new posts by email.

Submit

This site uses Akismet to reduce spam. Learn how your comment data is processed.

You can subscribe this blog and receive notifications for new posts

Email *

I'm not a robot

reCAPTCHA
Privacy - Terms

bot

reCAPTCHA
Privacy - Terms

Follow Blog