

Machine Learning

Video 76:

Precision, Recall and F1 Score | Classification Metrics Part 2

We know,

model \rightarrow No one is terrorist

		Predicted	
		1	0
Actual	1	0	1
	0	0	999

Accuracy = $\frac{999}{999+1} = 99.9\%$

But, what if accuracy of two models be same as well? Then we will look for the Precision:

Precision in the context of classification metrics refers to the ratio of correctly predicted positive instances to the total predicted positives. It measures the accuracy of positive predictions, highlighting how many of the predicted positive cases are actually true positives. Higher precision indicates fewer false positives. Thus, what proportion of predicted positives is truly positive?

Precision

Wednesday, June 23, 2021 8:46 AM

{ spam: 1, not spam: 0 }

	Predicted (A)			Predicted (B)		
	Sent to Spam	Not sent to spam		Sent to Spam	Not sent to spam	
Actual	Spam	100	170 FN	Spam	100	190
	Not Spam	30 FP	700	Not Spam	10	700

80% $FP_A > FP_B$ | $FN_A < FN_B$

Precision (B) deploy

80% FP Not spam \downarrow spam \downarrow FN \rightarrow spam \downarrow Not spam

$$P_A = \frac{100}{100 + 30} \quad P_B = \frac{100}{100 + 10}$$

	Sent to Spam	Not sent to spam
Spam	True Positive	False Negative
Not Spam	False Positive	True Negative

Precision = $\frac{TP}{TP+FP}$

Now, what if Accuracy is same for two models?

Recall, also known as sensitivity or true positive rate, is a classification metric that measures the proportion of actual positive instances correctly identified by the model. It indicates how well the model detects positive cases, highlighting how many true positives were captured out of all actual positives. Thus, what proportion of actual positives is correctly classified?

What proportion of actual Positives is correctly classified?

	Detected Cancer	Not detected Cancer
Has Cancer	True Positive	False Negative
No Cancer	False Positive	True Negative

Recall = $\frac{TP}{TP+FN}$

Recall

Wednesday, June 23, 2021 8:46 AM

has cancer: 1, no cancer: 0

Predicted

	Detected Cancer	Not Detected
Has Cancer	1000	200 (FN)
No Cancer	800 FP	8000

Actual

	Detected Cancer	Not Detected
Has Cancer	1000	500 + B
No Cancer	500	8000

A 90% B 90%

Recall = $\frac{1000}{1200}$

$R_A > R_B$

$R_B = \frac{1000}{1500}$ (FP)

If type 1 error is more dangerous then select the high precision model, if type 2 is more dangerous then select the high recall model.

The F1 score is the harmonic mean of precision and recall, providing a single metric that balances both. It is calculated as:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Use the F1 score when you need a balance between precision and recall, especially when dealing with imbalanced datasets where one class is significantly underrepresented. It's helpful when both false positives and false negatives are equally important.

Example:

Code link:

<https://github.com/campusx-official/100-days-of-machine-learning/blob/main/day59-classification-metrics/classification-metrics-binary.ipynb>

Multi-class precision and **multi-class recall** are extensions of precision and recall for classification tasks with more than two classes.

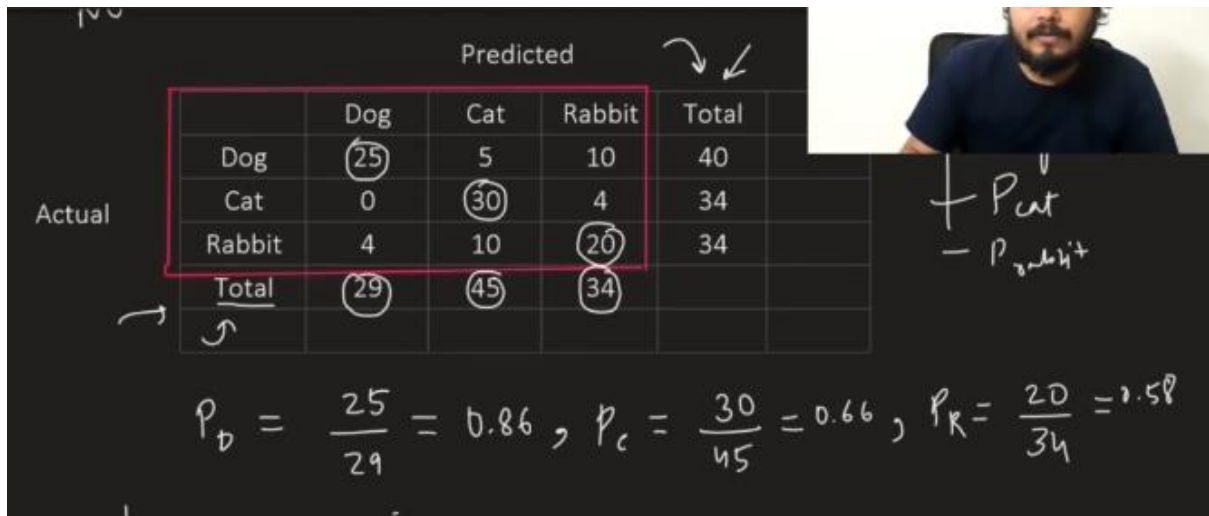
- **Multi-class Precision:** This is the precision calculated for each class, considering it as the positive class and all others as negative. Then, an average (typically macro or weighted) is taken across all classes.

- **Multi-class Recall:** Similarly, recall is calculated for each class, considering it as the positive class. The average recall across all classes is computed, often using macro or weighted averages.

Types of Averages:

1. **Macro-average:** Computes precision/recall for each class independently and then averages them. This treats all classes equally.
2. **Weighted-average:** Computes precision/recall for each class independently but weighs them by the number of true instances for each class, giving more importance to the classes with more instances.

These metrics are useful for evaluating models where there are more than two classes and the class distribution may be imbalanced.



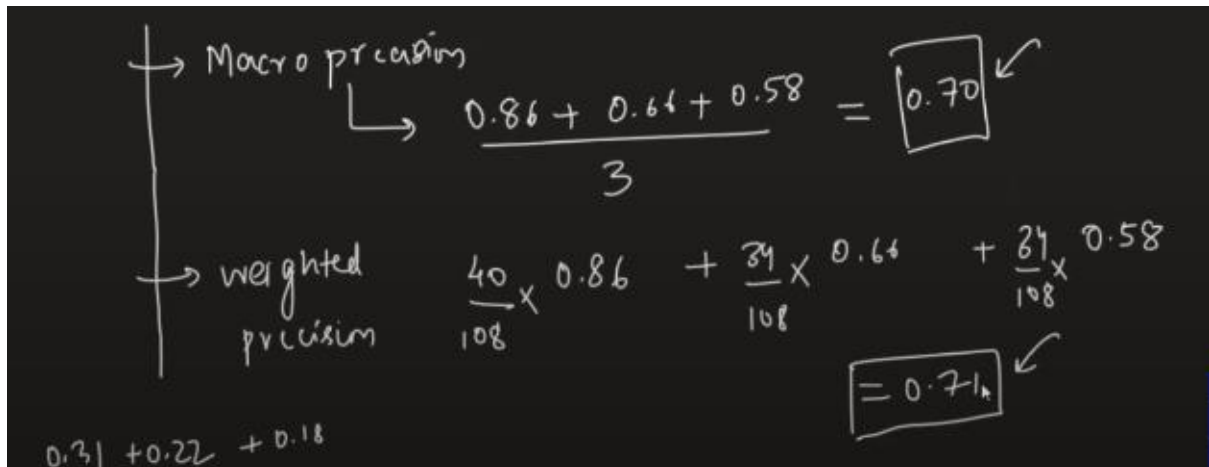
Confusion Matrix (Actual vs Predicted):

	Predicted			
	Dog	Cat	Rabbit	Total
Dog	25	5	10	40
Cat	0	30	4	34
Rabbit	4	10	20	34
Total	29	45	34	

Handwritten calculations for precision:

$$P_D = \frac{25}{29} = 0.86, \quad P_C = \frac{30}{45} = 0.66, \quad P_R = \frac{20}{34} = 0.58$$

Handwritten notes: $+ P_{cat}$, $- P_{rabbit}$



Macro precision:

$$\frac{0.86 + 0.66 + 0.58}{3} = 0.70$$

Weighted precision:

$$\frac{40}{108} \times 0.86 + \frac{34}{108} \times 0.66 + \frac{34}{108} \times 0.58 = 0.71$$

Handwritten notes: $0.31 + 0.22 + 0.18$

Now, calculating recall:

① Yes
No

② class

③

$$\frac{2 \times 0.86 + 0.64}{0.86 + 0.64}$$

R_D, R_C, R_R

$$F1_D = \frac{2 P_D R_D}{P_D + R_D}$$

$$F1_C = \frac{2 P_C R_C}{P_C + R_C}$$

$$F1_R = \frac{2 P_R R_R}{P_R + R_R}$$

	Predicted					
	Dog	Cat	Rabbit	Total	Recall	
Actual	Dog	25	5	10	40	0.62
	Cat	0	30	4	34	0.88
	Rabbit	4	10	20	34	0.58
	Total	29	45	34		
	Precision	0.86	0.66	0.58		

$$R_D = \frac{25}{40} = 0.62, R_C = \frac{30}{34} = 0.88, R_R = \frac{20}{34} = 0.58$$

macro recall
weighted recall

$F1_D, F1_C, F1_R$
macro f1
weighted f1

Example:

Code link:

<https://github.com/campusx-official/100-days-of-machine-learning/blob/main/day59-classification-metrics/classification-metrics-multi-mnist1.ipynb>

Video 77:

Softmax Regression || Multinomial Logistic Regression || Logistic Regression Part 6

Logistic regression fails when data is highly non-linear, has multicollinearity, or contains outliers. It also struggles with complex relationships between features or when classes are imbalanced. Additionally, if features are not scaled or are poorly chosen, logistic regression may not perform well in capturing intricate patterns.

Softmax regression (also known as **multinomial logistic regression**) can be used instead of logistic regression when dealing with **multi-class classification problems**. While logistic regression is typically used for binary classification, softmax regression extends it to handle multiple classes by applying the **softmax function** to model the probability distribution across multiple classes.

Softmax function

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

$K = \text{no. of class } \{3\}$

How training is done in softmax?

In **softmax regression**, training is done using a similar approach to logistic regression, but with modifications to handle multiple classes. Here's a step-by-step breakdown of how data is trained:

1. **Initialization:**

- The model starts with random weights for each class. For K classes, there will be K sets of weights, one for each class.

2. **Linear Transformation:**

- For each input \mathbf{x} , the model computes a linear combination of the features for each class: $z_k = \mathbf{w}_k^T \mathbf{x} + b_k$ for each class k where \mathbf{w}_k is the weight vector for class k , and b_k is the bias term.

3. **Softmax Function:**

- The linear outputs are passed through the **softmax function** to convert them into probabilities. The softmax function for class k is:

$$P(y = k | \mathbf{x}) = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$

- This ensures that the predicted outputs sum to 1 and are in the range $[0, 1]$.

4. **Loss Function (Cross-Entropy Loss):**

- The model is trained using the **cross-entropy loss function**. For a single data point \mathbf{x} with true label y , the loss is calculated as:

$$L(\mathbf{x}, y) = -\log P(y | \mathbf{x}) = -\log \left(\frac{e^{z_y}}{\sum_{j=1}^K e^{z_j}} \right)$$

- This penalizes the model more when the predicted probability for the correct class is low.

5. Gradient Descent:

- The weights w_k are updated using gradient descent to minimize the loss function. The gradient of the loss with respect to the weights is computed and used to adjust the weights in the direction that reduces the loss.

6. Iteration:

- This process is repeated for multiple epochs (iterations over the entire dataset) until the model converges to a set of weights that minimize the loss.

Key Points:

- **Softmax regression** generalizes logistic regression to multi-class classification by modeling the probabilities for each class.
- It uses the **softmax function** to output probabilities and **cross-entropy loss** to measure the difference between the predicted and true labels.
- The model is trained using **gradient descent** or other optimization techniques.

Example:

Code link:

<https://colab.research.google.com/drive/1sRHRp-RC8HjAT-Dg6v7t7wlpnne-d0o6?usp=sharing>

Video 78:

Polynomial Features in Logistic Regression | Non Linear Logistic Regression | Logistic Regression 7

Example:

Code link:

<https://colab.research.google.com/drive/1sRHRp-RC8HjAT-Dg6v7t7wlpnne-d0o6?usp=sharing>

Data link:

<https://www.kaggle.com/datasets/hafeezabro/ushape>

Video 79:

Logistic Regression Hyperparameters | | Logistic Regression Part 8

Document link:

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Video 80:

Decision Trees Geometric Intuition | Entropy | Gini impurity | Information Gain

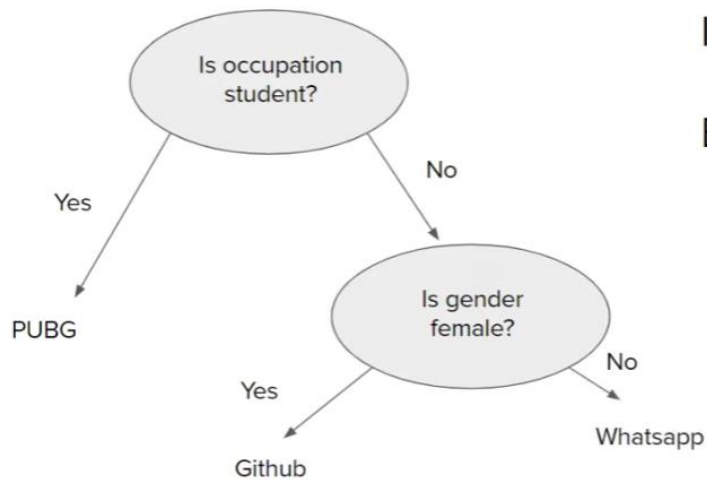
A decision tree in machine learning is a model used for classification or regression. It splits data into subsets based on feature values, forming a tree-like structure with nodes representing features and branches representing outcomes. It predicts outcomes by following the path from root to leaf nodes.

Example 1

Gender	Occupation	Suggestion
F	Student	PUBG
F	Programmer	Github
M	Programmer	Whatsapp
F	Programmer	Github
M	Student	PUBG
M	Student	PUBG

```
If occupation==student
    print(PUBG)
Else
    If gender==female
        print(Github)
    Else
        print(Whatsapp)
```


Where is the Tree?

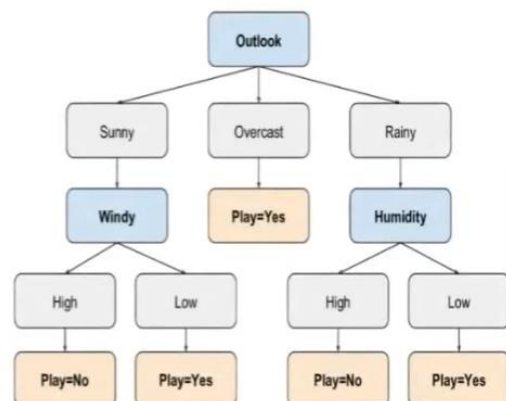


```

If occupation==student
    print(PUBG)
Else
    If gender==female
        print(Github)
    Else
        print(Whatsapp)
  
```

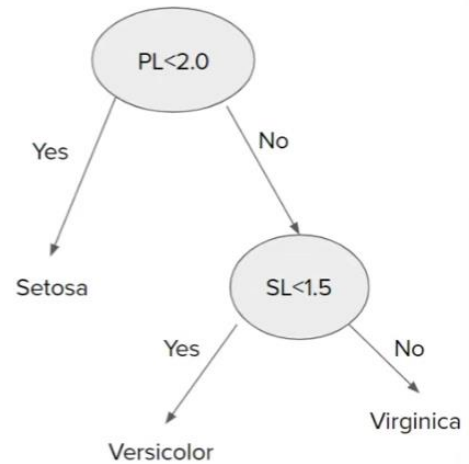
Example 2

Day	Outlook	Temp	Humid	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

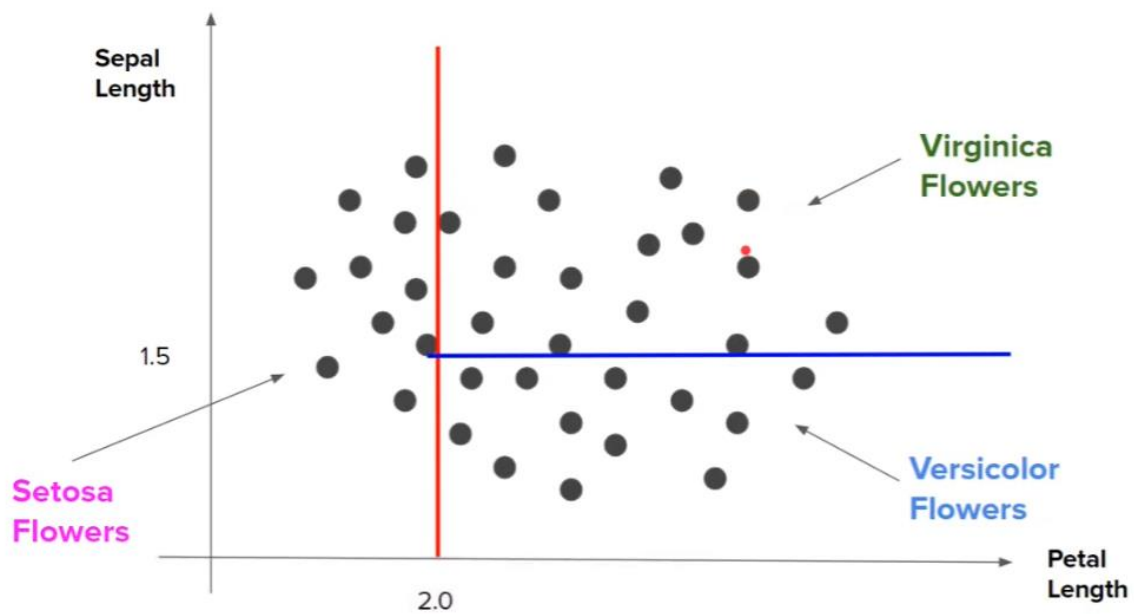


What if we have numerical data?

Petal Length	Sepal Length	Type
1.34	0.34	Setosa
3.45	1.45	Versicolor
1.69	0.98	Setosa
2.56	1.79	Virginica
3.00	1.13	Versicolor
1.3	0.88	Setosa



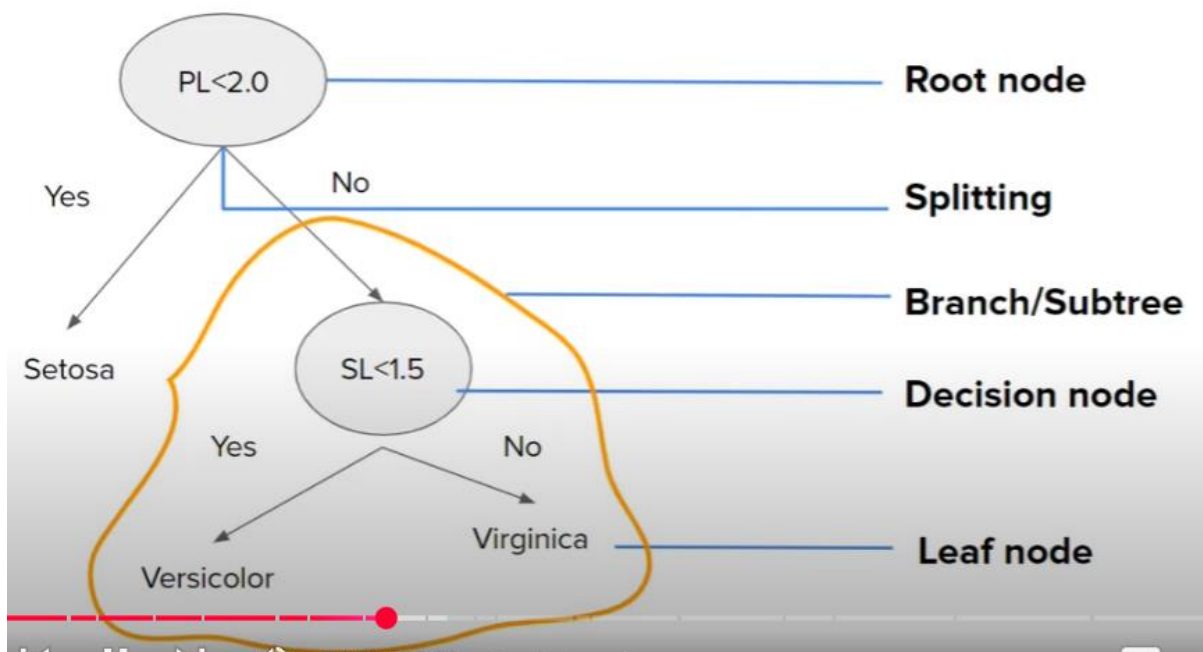
Geometric Intuition



Pseudo code

- Begin with your training dataset, which should have some feature variables and classification or regression output.
- Determine the “best feature” in the dataset to split the data on; more on how we define “best feature” later
- Split the data into subsets that contain the correct values for this best feature. This splitting basically defines a node on the tree i.e each node is a splitting point based on a certain feature from our data.
- Recursively generate new tree nodes by using the subset of data created from step 3.

Terminology



Advantages

Intuitive and easy to understand

Minimal data preparation is required

The cost of using the tree for inference is **logarithmic** in the number of data points used to train the tree

Disadvantages

Overfitting

Prone to errors for imbalanced datasets

What is Entropy?

In the most layman terms, Entropy is nothing but the measure of disorder. Or you can also call it the measure of purity/impurity. Let's see an example...

How to calculate Entropy?

The mathematical formula for entropy is:

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

For e.g if our data has only 2 class labels **Yes** and **No**.

Where 'Pi' is simply the frequentist probability of an element/class 'i' in our data.

$$E(D) = -p_{\text{yes}} \log_2(p_{\text{yes}}) - p_{\text{no}} \log_2(p_{\text{no}})$$

Example 3 - Dataset

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
12000	18	No

$$H(d) = -P_y \log_2(P_y) - P_n \log_2(P_n)$$

$$H(d) = -2/5 \log_2(2/5) - 3/5 \log_2(3/5)$$

$$H(d) = 0.97$$

Salary	Age	Purchase
34000	31	No
15000	25	No
69000	57	Yes
25000	21	No
32000	28	No

$$H(d) = -P_y \log_2(P_y) - P_n \log_2(P_n)$$

$$H(d) = -1/5 \log_2(1/5) - 4/5 \log_2(4/5)$$

$$H(d) = 0.72$$

Example 3 - Dataset

Salary	Age	Purchase
20000	21	No
10000	45	No
60000	27	No
15000	31	No
12000	18	No

$$H(d) = -P_y \log_2(P_y) - P_n \log_2(P_n)$$

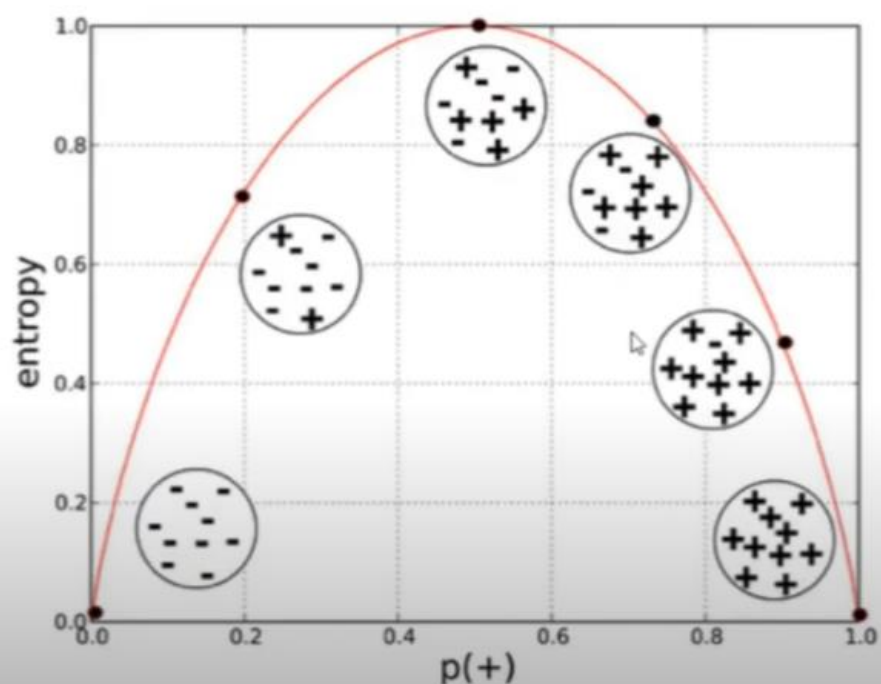
$$H(d) = -0/5 \log_2(0/5) - 5/5 \log_2(5/5)$$

$$H(d) = 0$$

Observation

- More the uncertainty more is entropy
- For a 2 class problem the min entropy is 0 and the max is 1
- For more than 2 classes the min entropy is 0 but the max can be greater than 1
- Both \log_2 or \log_e can be used to calculate entropy

Entropy Vs Probability



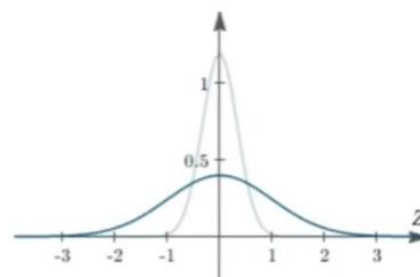
Entropy for continuous variables

Area	Built in	Price
1200	1999	3.5
1800	2011	5.6
1400	2000	7.3
...

Dataset 1

Area	Built in	Price
2200	1989	4.6
800	2018	6.5
1100	2005	12.8
...

Dataset 2



Quiz: Which of the above datasets have higher entropy?

Ans: Whichever is less peaked

Information Gain

Information Gain, is a metric used to train Decision Trees. Specifically, this metric measures the quality of a split.

The information gain is based on the decrease in entropy after a data-set is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain

$$\text{Information Gain} = E(\text{Parent}) - \{\text{Weighted Average}\} * E(\text{Children})$$

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Step 1:

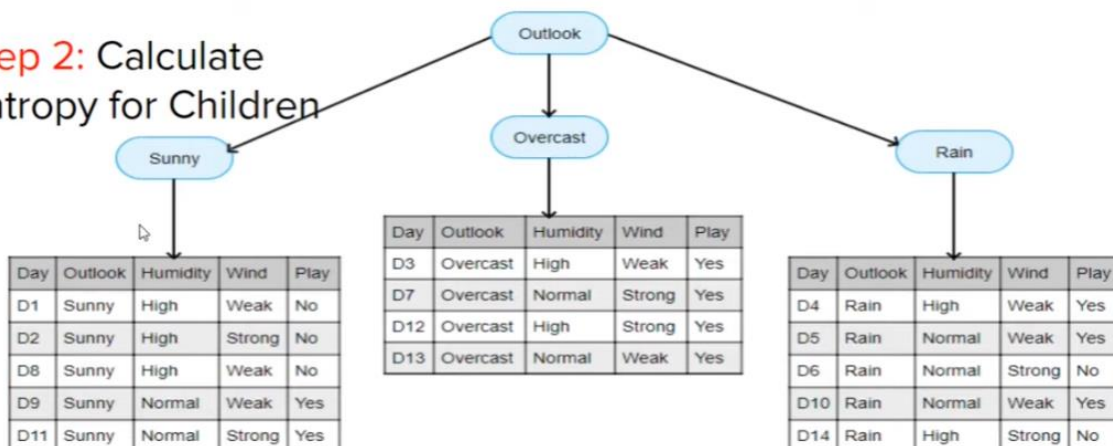
Entropy of Parent

$$E(P) = -p_y \log_2(p_y) - p_n \log_2(p_n)$$

$$= 9/14 \log_2(9/14) - 5/14 \log_2(5/14)$$

$$E(P) = \mathbf{0.94}$$

Step 2: Calculate Entropy for Children



$E(S) = -2/5 \log(2/5) - 3/5 \log(3/5)$ $E(S) = 0.97$	$E(O) = -5/5 \log(5/5) - 0/5 \log(0/5)$ $E(O) = 0$	$E(R) = -3/5 \log(3/5) - 2/5 \log(2/5)$ $E(S) = 0.97$
--	---	--

Step 3 : Calculate weighted Entropy of Children

$$\text{Weighted Entropy} = 5/14 * 0.97 + 4/14 * 0 + 5/14 * 0.97$$

$$W.E(\text{Children}) = \mathbf{0.69}$$

P(Overcast) is a leaf node as it's entropy is 0

Step 4 : Calculate Information Gain

$$\text{Information Gain} = E(\text{Parent}) - \{\text{Weighted Average}\} * E(\text{Children})$$

$$IG = \mathbf{0.97 - 0.69 = 0.28}$$

So the information gain(or the decrease in entropy/impurity) when you split this data on the basis of **Outlook** condition/column is **0.28**

Step 5 : Calculate Information Gain for all the columns

Whichever column has the highest Information Gain(maximum decrease in entropy) the algorithm will select that column to split the data.

Step 6 : Find Information Gain recursively

Decision tree then applies a recursive greedy search algorithm in top bottom fashion to find Information Gain at every level of the tree.

Once a leaf node is reached (Entropy = 0), no more splitting is done.

Gini impurity:

$$E = -P_Y \log_2(P_Y) - P_N \log_2(P_N)$$
$$G_i = 1 - (P_Y^2 + P_N^2)$$

Example 3 - Dataset

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
12000	18	No

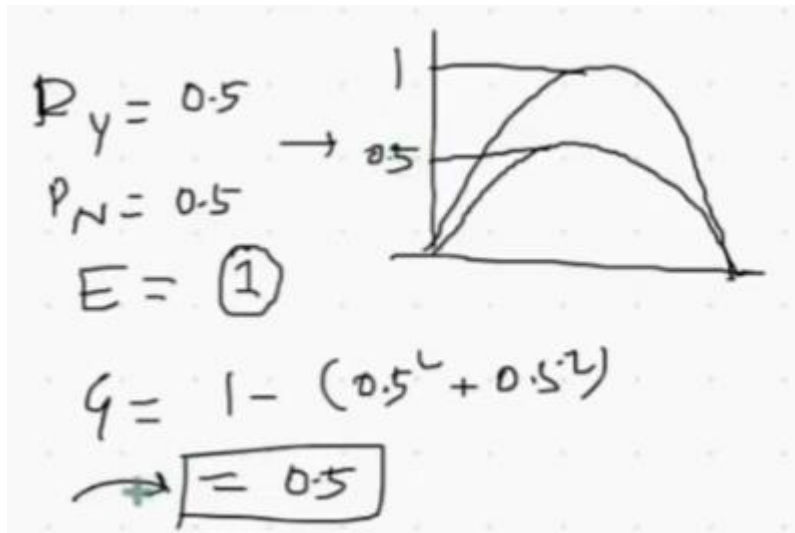
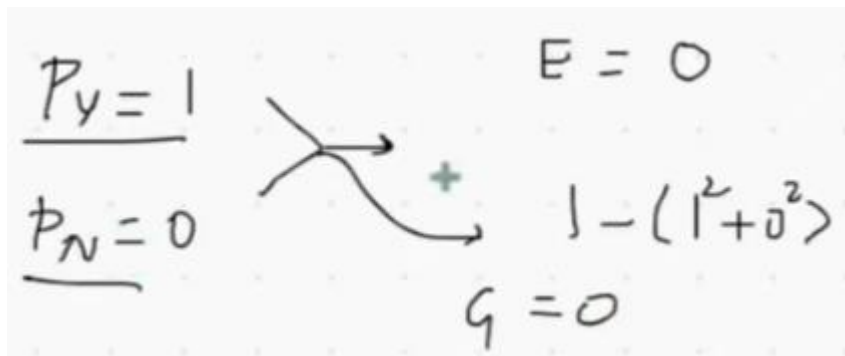
$$G = 1 - (P_Y^2 + P_N^2)$$
$$G = 1 - (4/25 + 9/25)$$

$$G = 0.48$$

Salary	Age	Purchase
34000	31	No
15000	25	No
69000	57	Yes
25000	21	No
32000	28	No

$$G = 1 - (P_Y^2 + P_N^2)$$
$$G = 1 - (1/25 + 16/25)$$

$$G = 0.32$$



Handling numerical data:

S No	User Rating	Downloaded
1	3.5	Yes
2	4.6	Yes
3	2.2	No
4	1.6	Yes
5	4.1	No
6	3.9	No
7	3.2	No
9	2.9	Yes
10	4.8	Yes
11	3.3	No
12	2.5	Yes

`data['user_rating'].nunique()=n`

We will then have n subtrees

51:59 / 58:28 • Handling Numerical Data >

Step 1:

Sort the data on the basis of numerical column

S No	User Rating	Downloaded
1	1.6	Yes
2	1.9	Yes
3	2.2	No
4	2.5	Yes
5	2.9	Yes
6	3.2	No
7	3.3	No
9	3.5	Yes
10	3.9	No
11	4.1	No
12	4.6	Yes
13	4.8	Yes

Step 2:

Split the entire data on the basis of every value of user_rating

Diagram illustrating a split operation based on the condition $\text{rating} > 1.6$.

The condition $\text{rating} > 1.6$ is shown in a circle, with arrows pointing to the resulting data partitions.

S No	User Rating	Downloaded
1	1.6	Yes

S No	User Rating	Downloaded
2	1.9	Yes
3	2.2	No
4	2.5	Yes
5	2.9	Yes
6	3.2	No
7	3.3	No
9	3.5	Yes
10	3.9	No
11	4.1	No
12	4.6	Yes
13	4.8	Yes

Step 2:

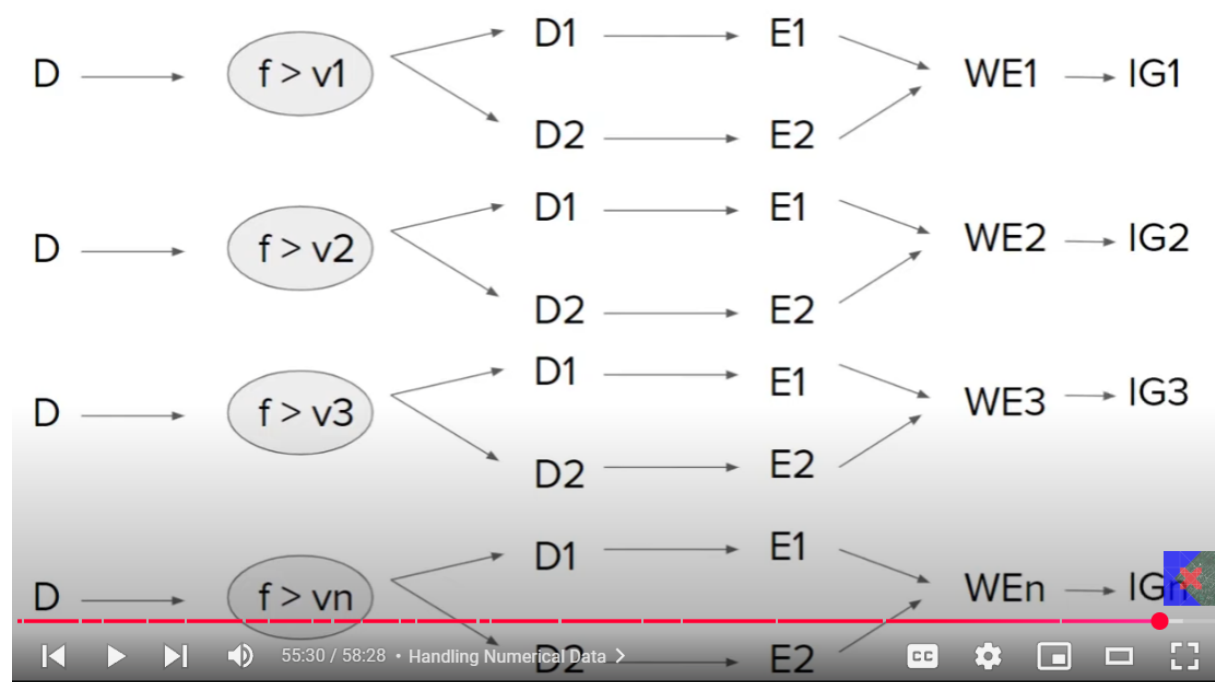
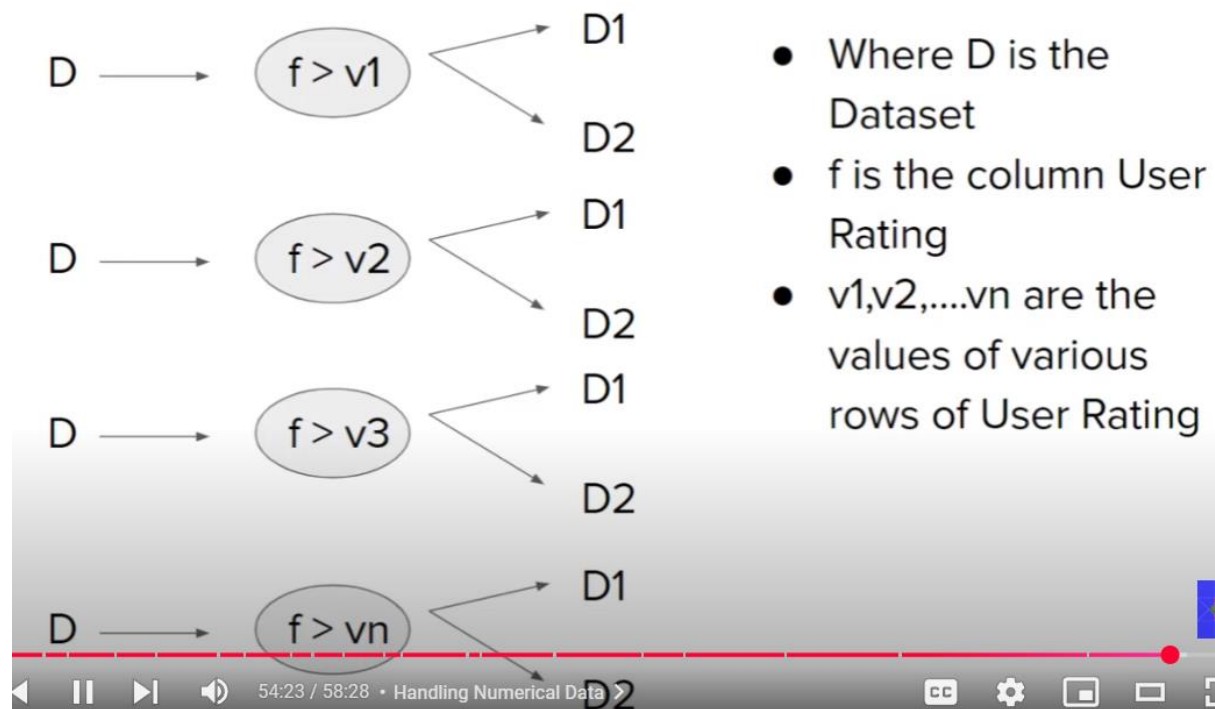
Split the entire data on the basis of every value of user_rating

rating > 1.9

S No	User Rating	Downloaded
1	1.6	Yes
2	1.9	Yes
3	2.2	No
4	2.5	Yes
5	2.9	Yes
6	3.2	No
7	3.3	No
9	3.5	Yes
10	3.9	No
11	4.1	No
12	4.6	Yes
13	4.8	Yes

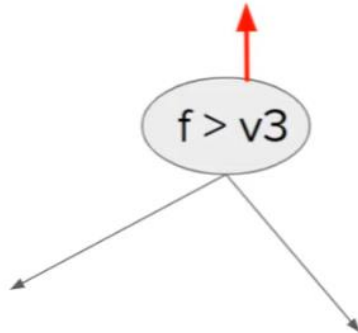
rating > 2.9

S No	User Rating	Downloaded
1	1.6	Yes
2	1.9	Yes
3	2.2	No
4	2.5	Yes
5	2.9	Yes
6	3.2	No
7	3.3	No
9	3.5	Yes
10	3.9	No
11	4.1	No
12	4.6	Yes
13	4.8	Yes



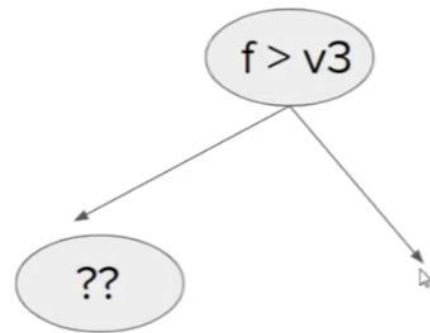
Step 5:

$\text{Max}\{ IG1, IG2, IG3\ldots, IGn \}$



Step 6:

Do this recursively until you get all the leaf nodes



Question

This entire thing looks computationally too expensive. Is is the right way of finding the splitting criteria?

Yes