**UNIT – 2 CLASSIFICATION**

Classification in data mining is a common technique that separates data points into different classes. It allows you to organize data sets of all sorts, including complex and large datasets as well as small and simple ones.

**Classification Techniques in Data Mining**

* [Regression](https://hevodata.com/learn/classification-techniques-in-data-mining/#1.2)
* [Naive Bayes Classification](https://hevodata.com/learn/classification-techniques-in-data-mining/#1.3)
* [K-Nearest Neighbour](https://hevodata.com/learn/classification-techniques-in-data-mining/#1.4)(KNN)
* Decision Trees

1. **Bayesian Classification** – It is a supervised learning algorithm based on the Bayes theorem. Bayesian classifiers view high efficiency and speed when used to high databases.

**P(Y/X)= (P(X/Y) \* P(Y) ) / P(X)**

**P(Y/X 1, X2 ,…..Xn)=P(X1/Y)\*P(X2/Y)……P(Xn/Y) \* P(Y) / P(X1) \*P(X2)…..P(Xn) for yes**

**P (N/X 1, X2 ,…..Xn)=P(X1/N)\*P(X2/N)……P(Xn/N) \* P(N) / P(X1) \*P(X2)…..P(Xn) for no**

* In Bayes classification ,we get the output from pre based knowledge.
* Bayes classification can predict class membership probability ,such as,the probability that a given tuple belongs to a particular class or not.
* Bayes classifiers are statistical classifiers. Ie. Here we use numerical or mathematical formulas to calculate bayes classification.
* It predicts probalility that a given record belongs to a particular class or not.

Problem 1: given in the below table,find whether person(Flu,Covid)belongs to which class ie. Fever(yes/no).

|  |  |  |  |
| --- | --- | --- | --- |
| Person | Covid(yes/no) | Flu(yes/no) | Fever(yes/no) |
| 1 | Yes | No | Yes |
| 2 | No | Yes | Yes |
| 3 | Yes | Yes | Yes |
| 4 | No | No | No |
| 5 | Yes | No | Yes |
| 6 | No | No | Yes |
| 7 | Yes | No | Yes |
| 8 | Yes | No | No |
| 9 | No | Yes | Yes |
| 10 | No | Yes | No |
|  |  |  |  |

Step 1: Prior probability

P(fever = yes) = 7 / 10

P(fever = no) = 3 /10

Step 2: Conditional probability

|  |  |  |
| --- | --- | --- |
|  | Yes | NO |
| COVID | 4/7 | 2/3 |
| FLU | 3/7 | 2/3 |

Note: 4/7 : if covid is yes and fever is yes and No with No is the condition

P(Yes / Flu , Covid) = P(Flu/yes)\*P(covid/yes)\*P(yes)

3/7 \* 4/7 \* 7/10 = **0.17**

P(No/Flu, Covid) = P(Flu/No)\*P(covid/yes)\*P(No)

= 2/3\* 2/3\*3/10 = **0.13**

Therefore, given probablity (flu,covid) belongs to yes class because

P(Yes/Flu,Covid) > P(No/Flu,Covid).

**PROBLEM 2:** Given the table below

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CAR NO. | COLOUR | TYPE | ORIGIN | STOLEN(CLASS) |
| 1 | RED | SPORTS | DOMESTIC | YES |
| 2 | RED | SPORTS | DOMESTIC | NO |
| 3 | RED | SPORTS | DOMESTIC | YES |
| 4 | YELLOW | SPORTS | DOMESTIC | NO |
| 5 | YELLOW | SPORTS | IMPORTED | YES |
| 6 | YELLOW | SUV | IMPORTED | NO |
| 7 | YELLOW | SUV | IMPORTED | YES |
| 8 | YELLOW | SUV | DOMESTIC | NO |
| 9 | RED | SUV | IMPORTED | NO |
| 10 | RED | SPORTS | IMPORTED | YES |

Given instance : Red,Suv,Domestic belongs to which class?

Step 1: Prior probability : P(yes)=5/10

P(no)=5/10

Step 2: Conditional Probability:

|  |  |  |
| --- | --- | --- |
| Color | Yes | No |
| Red | 3/5 | 2/5 |
| Yellow | 2/5 | 3/5 |

|  |  |  |
| --- | --- | --- |
| Type | Yes | No |
| Sports | 4/5 | 2/5 |
| Suv | 1/5 | 3/5 |

|  |  |  |
| --- | --- | --- |
| Origin | Yes | No |
| Domestic | 2/5 | 3/5 |
| Imported | 3/5 | 2/5 |

P(Yes/Red,Suv,Domestic)=P(Red/Yes)\*P(Suv/Yes)\*P(Domestic/yes)\*P(Yes)

3/5 \* 1/5 \* 2/5 \* 5/10= 0.024

P(No/Red,Suv,Domestic)=P(Red/No)\*P(Suv/No)\*P(Domestic/No)\*P(No)

2/5 \* 3/5 \*3/5 \* 5/10=0.072

Therefore Red,Suv,Domestic belongs to “ No” class because 0.072>0.024

1. **K-Nearest Neighbors algorithm:**

**Step #1 -**Assign a value to **K**.

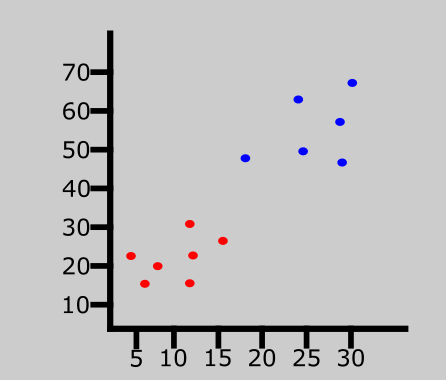
**Step #2 -**Calculate the distance between the new data entry and all other existing data entries (you'll learn how to do this shortly). Arrange them in ascending order.

**Step #3 -**Find the **K** nearest neighbors to the new entry based on the calculated distances.

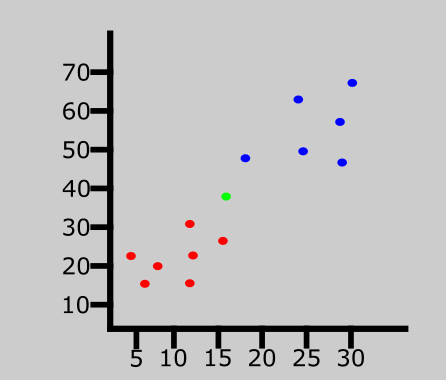
**Step #4 -**Assign the new data entry to the majority class in the nearest neighbors.

**K-Nearest Neighbors Classifiers and Model Example With Diagrams**

Consider the diagram below:

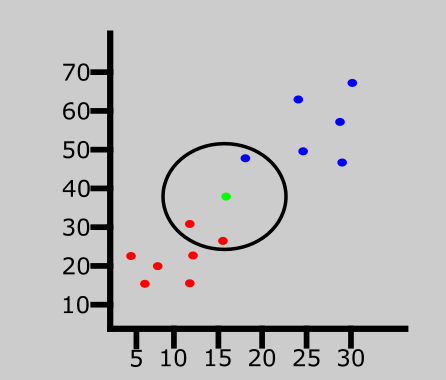


The graph above represents a data set consisting of two classes — red and blue.



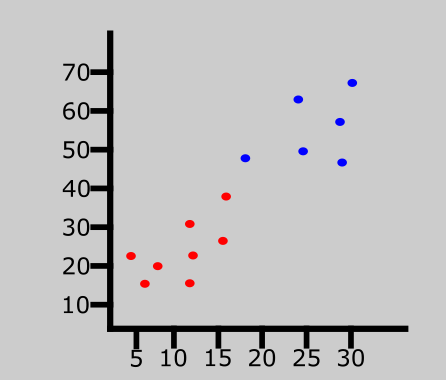
A new data entry has been introduced to the data set. This is represented by the green point in the graph above.

We'll then assign a value to **K**which denotes the number of neighbors to consider before classifying the new data entry. Let's assume the value of **K**is 3.



Since the value of **K**is 3, the algorithm will only consider the 3 nearest neighbors to the green point (new entry). This is represented in the graph above.

Out of the 3 nearest neighbors in the diagram above, the majority class is red so the new entry will be assigned to that class.



The last data entry has been classified as red.

**K-Nearest Neighbors Classifiers and Model Example With Data Set**

calculate the distance between a new entry and other existing values using the Euclidean distance formula.

Note: you can also calculate the distance using the Manhattan and Minkowski distance formulas.

| **BRIGHTNESS** | **SATURATION** | **CLASS** |
| --- | --- | --- |
| 40 | 20 | Red |
| 50 | 50 | Blue |
| 60 | 90 | Blue |
| 10 | 25 | Red |
| 70 | 70 | Blue |
| 60 | 10 | Red |
| 25 | 80 | Blue |

The table above represents our data set. We have two columns — **Brightness**and **Saturation**. Each row in the table has a class of either **Red**or **Blue**.

Before we introduce a new data entry, let's assume the value of **K**is 5.

**How to Calculate Euclidean Distance in the K-Nearest Neighbors Algorithm**

Here's the new data entry:

| **BRIGHTNESS** | **SATURATION** | **CLASS** |
| --- | --- | --- |
| 20 | 35 | ? |

We have a new entry but it doesn't have a class yet. To know its class, we have to calculate the distance from the new entry to other entries in the data set using the Euclidean distance formula.

Here's the formula: √(X₂-X₁)²+(Y₂-Y₁)²

Where:

* X₂ = New entry's brightness (20).
* X₁= Existing entry's brightness.
* Y₂ = New entry's saturation (35).
* Y₁ = Existing entry's saturation.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

d1 = √(20 - 40)² + (35 - 20)²  
= √400 + 225  
= √625  
= 25

d2 = √(20 - 50)² + (35 - 50)²  
= √900 + 225  
= √1125  
= 33.54

d3 = √(20 - 60)² + (35 - 90)²  
= √1600 + 3025  
= √4625  
= 68.01

Table after all the distances have been calculated:

| **BRIGHTNESS** | **SATURATION** | **CLASS** | **DISTANCE** |
| --- | --- | --- | --- |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 60 | 90 | Blue | 68.01 |
| 10 | 25 | Red | 10 |
| 70 | 70 | Blue | 61.03 |
| 60 | 10 | Red | 47.17 |
| 25 | 80 | Blue | 45 |

Let's rearrange the distances in ascending order:

| **BRIGHTNESS** | **SATURATION** | **CLASS** | **DISTANCE** |
| --- | --- | --- | --- |
| 10 | 25 | Red | 10 |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 25 | 80 | Blue | 45 |
| 60 | 10 | Red | 47.17 |
| 70 | 70 | Blue | 61.03 |
| 60 | 90 | Blue | 68.01 |

Since we chose 5 as the value of **K**, we'll only consider the first five rows. That is:

| **BRIGHTNESS** | **SATURATION** | **CLASS** | **DISTANCE** |
| --- | --- | --- | --- |
| 10 | 25 | Red | 10 |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 25 | 80 | Blue | 45 |
| 60 | 10 | Red | 47.17 |

As you can see above, the majority class within the 5 nearest neighbors to the new entry is **Red**. Therefore, we'll classify the new entry as **Red**.

Here's the updated table:

| **BRIGHTNESS** | **SATURATION** | **CLASS** |
| --- | --- | --- |
| 40 | 20 | Red |
| 50 | 50 | Blue |
| 60 | 90 | Blue |
| 10 | 25 | Red |
| 70 | 70 | Blue |
| 60 | 10 | Red |
| 25 | 80 | Blue |
| 20 | 35 | Red |

**How to Choose the Value of K in the K-NN Algorithm**

There is no particular way of choosing the value **K**, but here are some common conventions to keep in mind:

* Choosing a very low value will most likely lead to inaccurate predictions.
* The commonly used value of **K**is 5.
* Always use an odd number as the value of **K**.

**Advantages of K-NN Algorithm**

* It is simple to implement.
* No training is required before classification.

**Disadvantages of K-NN Algorithm**

* Can be cost-intensive when working with a large data set.
* A lot of memory is required for processing large data sets.
* Choosing the right value of **K** can be tricky.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. **Decision Tree**   A Decision Tree is a popular machine learning algorithm used for both classification and regression tasks. It is a tree-like structure that represents a series of decisions and their possible outcomes. Each internal node of the tree corresponds to a feature or attribute, each branch represents a decision based on that attribute, and each leaf node represents the final outcome or class label. Decision Trees are interpretable and easy to understand, making them useful for both analysis and prediction.  **Decision Tree Terminologies**  **Root Node**- It is the topmost node in the tree, which represent the complete dataset. Also we can say it is the starting point of the decision-making process.  **Decision/Internal Node-**Decision nodes are nothing but the result in the splitting of data into multiple data segments and main goal is to have the children nodes with maximum homogeneity or purity( means all of the same kind).  **Leaf/Terminal Node**- This node represent the data section having highest homogeneity (means all of the same kind).  **Entropy**- Entropy is the measurement of impurities or randomness in the data points.   If all elements belong to a single class, then it is termed as “Pure”, and if not then the distribution is named as “Impurity”.  It is used for checking the impurity or uncertainty present in the data. Entropy is used to evaluate the quality of a split. When entropy is zero the sample is completely homogeneous, meaning that each instance belongs to the same class and entropy is one when the sample is equally divided between different classes.  Decision tree algorithms:   1. **ID3 Algorithm:** 2. **C4.5 algorithm** 3. **CART**   **ID3 Algorithm:**  The ID3 (Iterative Dichotomiser 3) algorithm is one of the earliest and most widely used algorithms to create Decision Trees from a given dataset. It uses the concept of entropy and information gain to select the best attribute for splitting the data at each node. Entropy measures the uncertainty or randomness in the data, and information gain quantifies the reduction in uncertainty achieved by splitting the data on a particular attribute. The ID3 algorithm recursively splits the dataset based on the attributes with the highest information gain until a stopping criterion is met, resulting in a Decision Tree that can be used for classification tasks.  **Steps to Create a Decision Tree using the ID3 Algorithm:**  **Step 1: Data Preprocessing:** Clean and preprocess the data. Handle missing values and convert categorical variables into numerical representations if needed.  **Step 2: Selecting the Root Node:** Calculate the entropy of the target variable (class labels) based on the dataset. The formula for entropy is:  Entropy(S) = -Σ(Pi \* log2(P\_i)) where Pi is the probability of instances belonging to class i.  **Step 3: Calculating Information Gain:** For each attribute in the dataset, calculate the information gain when the dataset is split on that attribute. The formula for information gain is: Information Gain(S, A) = Entropy(S) - Σ ((|S v| / |S|) \* Entropy(S\_v)) where S\_v is the subset of instances for each possible value of attribute A, and |S\_v| is the number of instances in that subset.  **Step 4: Selecting the Best Attribute:** Choose the attribute with the highest information gain as the decision node for the tree.  **Step 5: Splitting the Dataset:** Split the dataset based on the values of the selected attribute.  **Step 6: Repeat the Process:** Recursively repeat steps 2 to 5 for each subset until a stopping criterion is met (e.g., the tree depth reaches a maximum limit or all instances in a subset belong to the same class).  Example:  Let’s illustrate the ID3 algorithm with a simple example of classifying whether to play tennis based on weather conditions. Consider the following dataset:  Ezoic   | **Weather** | **Temperature** | **Humidity** | **Windy** | **Play Tennis?** | | --- | --- | --- | --- | --- | | 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: Data Preprocessing:** The dataset does not require any preprocessing, as it is already in a suitable format.  **Step 2: Calculating Entropy:** To calculate entropy, we first determine the proportion of positive and negative instances in the dataset:   * Positive instances (Play Tennis = Yes): 9 * Negative instances (Play Tennis = No): 5   Entropy(S) = -(9/14) \* log2(9/14) – (5/14) \* log2(5/14) ≈ 0.940  **Step 3: Calculating Information Gain:** We calculate the information gain for each attribute (Weather, Temperature, Humidity, Windy) and choose the attribute with the highest information gain as the root node.  Information Gain(S, Weather) = Entropy(S) – [(5/14) \* Entropy(Sunny) + (4/14) \* Entropy(Overcast) + (5/14) \* Entropy(Rainy)] ≈ 0.246  Information Gain(S, Temperature) = Entropy(S) – [(4/14) \* Entropy(Hot) + (4/14) \* Entropy(Mild) + (6/14) \* Entropy(Cool)] ≈ 0.029  Information Gain(S, Humidity) = Entropy(S) – [(7/14) \* Entropy(High) + (7/14) \* Entropy(Normal)] ≈ 0.152  Information Gain(S, Windy) = Entropy(S) – [(8/14) \* Entropy(False) + (6/14) \* Entropy(True)] ≈ 0.048  **Step 4: Selecting the Best Attribute:** The “Weather” attribute has the highest information gain, so we select it as the root node for our decision tree.  **Step 5: Splitting the Dataset:** We split the dataset based on the values of the “Weather” attribute into three subsets (Sunny, Overcast, Rainy).  **Step 6: Repeat the Process:** Since the “Weather” attribute has n0o repeating values in any subset, we stop splitting and label each leaf node with the majority class in that subset. The decision tree will look like below:    **Advantages**   * Inexpensive to construct * Extremely fast at classifying unknown records Easy to interpret for small-sized trees. * Robust to noise (especially when methods to avoid over-fitting are employed). * Can easily handle redundant or irrelevant attributes (unless the attributes are interacting).   **Disadvantages**   * The space of possible decision trees is exponentially large. Greedy approaches are often unable to find the best tree. * Does not take into account interactions between attributes. * Each decision boundary involves only a single attribute.   **C4.5 algorithm**  C 4.5 is the successor of ID3.It is the improved version of ID3. It makes use of Gain ratio.  **Calculating Gain & Gain Ratios:**   * 1. GainRatio(A) = Gain(A) / SplitInfo(A)   2. Information Gain(S, A) = Entropy(S) - Σ ((|S v| / |S|) \* Entropy(S\_v)) where S\_v is the subset of instances for each possible value of attribute A, and |S\_v| is the number of instances in that subset.   3. Entropy(S) = -Σ(Pi \* log2(P\_i)) where Pi is the probability of instances belonging to class i.   4. SplitInfo(A) = -∑ |Dj|/|D| \* log2|Dj|/|D|   Where Dj is the number of cases of a particular value of an attribute. D here is the total number of cases of that attribute.   * 1. Select the highest value of gain ratio and proceed .   **Dataset:**  The data contains information on weather – related to temperature, humidity, wind, etc. This is a small dataset of 14 rows only.  The column description is as follows:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **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 |   **Calculating Global Entropy**  There are 14 rows in our data. 9 of them lead to “Yes” decision and 5 lead to “No” decision.  Entropy = – ∑ p(i) \* log2p(i)  = – [**p(Yes) \* log2p(Yes)**] – [**p(No) \* log2p(No)**]  = – (9/14) \* log2(9/14) – (5/14) \* log2(5/14)  = 0.940  **Calculating Gain & Gain Ratios:**  GainRatio(A) = Gain(A) / SplitInfo(A)  SplitInfo(A) = -∑ |Dj|/|D| \* log2|Dj|/|D|  Dj is number of cases of a particular value of an attribute. D here is the total number of cases of that attribute.  **I. Gain & Gain Ratio for Outlook Variable:**  Outlook variable is nominal. It has 3 values: Sunny, Overcast, Rain.  **Gain (Decision, Outlook) = Entropy(Decision) – ∑ [ p(Decision|Outlook) \* Entropy(Decision|Outlook) ]**  The above big formula is nothing but the formula for calculating gain. Let’s call this **Equation 1.**  The first part, i.e, **Entropy(Decision)** has already been calculated by us as **0.940**  The second part is the negative summation of the products of (i) Probability of that Outlook value to result in a “Yes” or “No” case, and (ii) Entropy of that Outlook value.  Let’s calculate this 2nd part, i.e, Entropy  **1. entropy for Outlook = Sunny**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Day** | **Outlook** | **Temp.** | **Humidity** | **Wind** | **Decision** | | 1 | Sunny | 85 | 85 | Weak | No | | 2 | Sunny | 80 | 90 | Strong | No | | 8 | Sunny | 72 | 95 | Weak | No | | 9 | Sunny | 69 | 70 | Weak | Yes | | 11 | Sunny | 75 | 70 | Strong | Yes |   We have 3 No decisions and 2 Yes decisions.  **Entropy(Decision|Outlook=Sunny)**  = – p(No) \* log2p(No) – p(Yes) \* log2p(Yes)  = -(3/5).log2(3/5) – (2/5).log2(2/5)  = 0.441 + 0.528  = 0.970  **2. Entropy for Outlook = Overcast**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Day** | **Outlook** | **Temp.** | **Humidity** | **Wind** | **Decision** | | 3 | Overcast | 83 | 78 | Weak | Yes | | 7 | Overcast | 64 | 65 | Strong | Yes | | 12 | Overcast | 72 | 90 | Strong | Yes | | 13 | Overcast | 81 | 75 | Weak | Yes |   All decisions are Yes here.  **Entropy(Decision|Outlook=Overcast)**  = – p(No) \* log2p(No) – p(Yes) \* log2p(Yes)  = -(0/4)\*log2(0/4) – (4/4)\*log2(4/4)  [Here log20 should be undefined. But we took it as 0. Because if we consider x\*log2x, then if x tends to 0, x\*log2x also tends to 0]  = 0  **3. Entropy for Outlook = Rain**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Day** | **Outlook** | **Temp.** | **Humidity** | **Wind** | **Decision** | | 4 | Rain | 70 | 96 | Weak | Yes | | 5 | Rain | 68 | 80 | Weak | Yes | | 6 | Rain | 65 | 70 | Strong | No | | 10 | Rain | 75 | 80 | Weak | Yes | | 14 | Rain | 71 | 80 | Strong | No |   We have 3 Yes and 2 No decisions.  **Entropy(Decision|Outlook=Rain)**  = – p(No) \* log2p(No) – p(Yes) \* log2p(Yes)  = -(2/5)\*log2(2/5) – (3/5)\*log2(3/5)  = 0.528 + 0.441  = 0.970  **4. Gain for Outlook variable:**  We are done with calculating Entropies for Outlook variable.  Putting these in the Equation 1 above:  **Gain(Decision, Outlook)**  = 0.940 – (5/14)\*(0.970) – (4/14)\*(0) – (5/14)\*(0.970)  = 0.247  **5. SplitInfo for Outlook variable:**  Sunny: 5 cases  Overcast: 4 cases  Rain: 5 cases  **SplitInfo(Decision, Outlook)**  = -(5/14)\*log2(5/14) -(4/14)\*log2(4/14) -(5/14)\*log2(5/14)  = 1.577  **6. Finally, Gain Ratio for Outlook variable:**  **GainRatio(Decision, Outlook)**  = Gain(Decision, Outlook)/SplitInfo(Decision, Outlook)  = 0.247/1.577  = 0.156  More work needs to be done. This is Gain Ratio for just 1 of the attributes. we have to calculate this Gain Ratio for the other variables too so that we can compare them at the end.  **II. Gain & Gain Ratio for Wind Variable:**  This is also a nominal variable. It has 2 values: Weak & Strong.  **Gain (Decision, Wind) = Entropy(Decision) – ∑ [ p(Decision|Wind) \* Entropy(Decision|Wind) ]**  Let’s call this **Equation 2.**  **1. Entropy for Wind = Weak**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Day** | **Outlook** | **Temp.** | **Humidity** | **Wind** | **Decision** | | 1 | Sunny | 85 | 85 | Weak | No | | 3 | Overcast | 83 | 78 | Weak | Yes | | 4 | Rain | 70 | 96 | Weak | Yes | | 5 | Rain | 68 | 80 | Weak | Yes | | 8 | Sunny | 72 | 95 | Weak | No | | 9 | Sunny | 69 | 70 | Weak | Yes | | 10 | Rain | 75 | 80 | Weak | Yes | | 13 | Overcast | 81 | 75 | Weak | Yes |   We have 6 Yes and 2 No decisions.  **Entropy(Decision|Wind=Weak)**  = – p(No) \* log2p(No) – p(Yes) \* log2p(Yes)  = – (2/8) \* log2(2/8) – (6/8) \* log2(6/8)  = 0.811  **2. Entropy for Wind = Strong**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Day** | **Outlook** | **Temp.** | **Humidity** | **Wind** | **Decision** | | 2 | Sunny | 80 | 90 | Strong | No | | 6 | Rain | 65 | 70 | Strong | No | | 7 | Overcast | 64 | 65 | Strong | Yes | | 11 | Sunny | 75 | 70 | Strong | Yes | | 12 | Overcast | 72 | 90 | Strong | Yes | | 14 | Rain | 71 | 80 | Strong | No |   We have 3 Yes and 3 No decisions.  **Entropy(Decision|Wind=Strong)**  = – (3/6) \* log2(3/6) – (3/6) \* log2(3/6)  = 1  **3. Gain for Wind variable:**  **Gain(Decision, Wind)**  = 0.940 – (8/14)\*(0.811) – (6/14)\*(1)  = 0.940 – 0.463 – 0.428  = 0.049  **4. SplitInfo for Wind variable:**  Weak: 8 cases  Strong: 6 cases  **SplitInfo(Decision, Wind)**  = -(6/14)\*log2(6/14) -(8/14)\*log2(8/14)  = 0.524 + 0.461  = 0.985  **5. Finally, Gain Ratio for Wind variable:**  **GainRatio(Decision, Wind)**  = Gain(Decision, Wind)/SplitInfo(Decision, Wind)  = 0.049 / 0.985  = 0.049  **III. Gain & Gain Ratio for Humidity Variable:**  This is where things get interesting because Humidity is a **continuous variable.**How do we deal with them?  **Step 1.** Arrange the values in ascending order.  **Step 2.** Convert them to nominal values by performing a binary split on a threshold value.  [Gain for this variable must be maximum at the threshold value.]  **Step 3.** The gain at this threshold value will be used for comparison of gains and gain ratios of all the variables.  **1. Let’s arrange it in ascending order of values of Humidity:**   |  |  |  | | --- | --- | --- | | **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 calculate the gains and gain ratios for every value of Humidity. The value which maximizes the gain would be the threshold. Here, we will separate our dataset in 2 parts: (i) values less than or equal to the current value, and (ii) values greater than the current value.  **2. Calculating Gains and Gain Ratios for all values:**  **2.a. For Humidity = 65**  We have 1 Yes & 0 No decisions at <= 65 and 8 Yes & 5 No decisions at > 65  **Entropy(Decision|Humidity<=65)**  = – p(No) . log2p(No) – p(Yes) . log2p(Yes)  = -(0/1).log2(0/1) – (1/1).log2(1/1)  = 0  **Entropy(Decision|Humidity>65)**  = -(5/13).log2(5/13) – (8/13).log2(8/13)  =0.530 + 0.431  = 0.961  **Gain(Decision, Humidity<> 65)**  = 0.940 – (1/14).0 – (13/14).(0.961)  = **0.048**  **SplitInfo(Decision, Humidity<> 65)** =  -(1/14).log2(1/14) -(13/14).log2(13/14)  = 0.371  **GainRatio(Decision, Humidity<> 65)**  = 0.048/0.371  = **0.129**  **2.b. For Humidity = 70**  We have 3 Yes & 1 No decisions at <= 70 and 6 Yes & 4 No decisions at > 70  **Entropy(Decision|Humidity<=70)**  = – p(No) . log2p(No) – p(Yes) . log2p(Yes)  = -(1/4).log2(1/4) – (3/4).log2(3/4)  = 0.811  **Entropy(Decision|Humidity>70)**  = -(4/10).log2(4/10) – (6/10).log2(6/10)  = 0.971  **Gain(Decision, Humidity<> 70)**  = 0.940 – (4/14).(0.811) – (10/14).(0.971)  = **0.014**  **SplitInfo(Decision, Humidity<> 70)**  = -(4/14).log2(4/14) -(10/14).log2(10/14)  = 0.863  **GainRatio(Decision, Humidity<> 70)**  = 0.014/0.863  = **0.016**  Similarly, calculate the Gains and Gain Ratios for all other values of Humidity.  We found out that the Gain was **maximum for Humidity = 80**  [Note: Here is something interesting. You can take either Gain or Gain Ratio as the threshold value. Both of them will result in different Decision Trees. We are taking Gain.]  **Gain(Decision, Humidity <> 80) = 0.101**  **GainRatio(Decision, Humidity <> 80) = 0.107**  **IV. Gain & Gain Ratio for Temp. Variable:**  This is also a continuous variable. We will repeat the steps we did for Humidity variable.  **1. Let’s arrange it in ascending order of values of Temp:**   |  |  |  | | --- | --- | --- | | **Day** | **Temp.** | **Decision** | | 7 | 64 | Yes | | 6 | 65 | No | | 5 | 68 | Yes | | 9 | 69 | Yes | | 4 | 70 | Yes | | 14 | 71 | No | | 8 | 72 | No | | 12 | 72 | Yes | | 10 | 75 | Yes | | 11 | 75 | Yes | | 2 | 80 | No | | 13 | 81 | Yes | | 3 | 83 | Yes | | 1 | 85 | No |   **2. Calculating Gains and Gain Ratios for all values:**  **2.a. For Temp = 64**  We have 1 Yes & 0 No decisions at <= 64 and 8 Yes & 5 No decisions at > 64  **Entropy(Decision|Temp<=64)**  = – p(No) . log2p(No) – p(Yes) . log2p(Yes)  = -(0/1).log2(0/1) – (1/1).log2(1/1)  = 0  **Entropy(Decision|Temp>64)**  = -(5/13).log2(5/13) – (8/13).log2(8/13)  =0.530 + 0.431  = 0.961  **Gain(Decision, Temp <> 64)**  = 0.940 – (1/14).0 – (13/14).(0.961)  = **0.048**  **SplitInfo(Decision, Temp <> 64)** =  -(1/14).log2(1/14) -(13/14).log2(13/14)  = 0.371  **GainRatio(Decision, Temp <> 64)**  = 0.048/0.371  = **0.129**  **2.b. For Temp = 65**  We have 1 Yes & 1 No decisions at <= 65 and 8 Yes & 4 No decisions at > 65  **Entropy(Decision|Temp<=65)**  = – p(No) . log2p(No) – p(Yes) . log2p(Yes)  = -(1/2).log2(1/2) – (1/2).log2(1/2)  = 1  **Entropy(Decision|Temp>65)**  = -(4/12).log2(4/12) – (8/12).log2(8/12)  = 0.918  **Gain(Decision, Temp<> 65)**  = 0.940 – (2/14).1 – (12/14).(0.918)  = 0.010  **SplitInfo(Decision, Temp<> 65)**  = -(2/14).log2(2/14) -(12/14).log2(12/14)  = 0.591  **GainRatio(Decision, Temp<> 65)**  = 0.010/0.591  = 0.017  Similarly, calculate the Gains and Gain Ratios for all other values of Temp.  We found out that the Gain was **maximum for Temp = 83**  **Gain(Decision, Temp <> 83) = 0.113**  **GainRatio(Decision, Temp <> 83) = 0.305**  **Comparison of Gains and Gain Ratios**   |  |  |  | | --- | --- | --- | | **Attribute** | **Gain** | **Gain Ratio** | | Wind | 0.049 | 0.049 | | Outlook | 0.247 | 0.156 | | Humidity <> 80 | 0.101 | 0.107 | | Temp <> 83 | 0.113 | 0.305 |   If we use Gain, Outlook will be the root node. (Because it has the highest Gain value)  Similarly, if we use Gain Ratio, Temp will be the root node.  We will proceed using the Gain.  **Outlook = Sunny**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Day** | **Outlook** | **Temp.** | **Humidity** | **Wind** | **Decision** | | 1 | Sunny | 85 | 85 | Weak | No | | 2 | Sunny | 80 | 90 | Strong | No | | 8 | Sunny | 72 | 95 | Weak | No | | 9 | Sunny | 69 | 70 | Weak | Yes | | 11 | Sunny | 75 | 70 | Strong | Yes |   If humidity > 80, decision is ‘No’  If humidity <= 80, decision is ‘Yes’  https://insightimi.files.wordpress.com/2020/03/image-6-3.png?w=1024  **Outlook = Overcast**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Day** | **Outlook** | **Temp.** | **Humidity** | **Wind** | **Decision** | | 3 | Overcast | 83 | 78 | Weak | Yes | | 7 | Overcast | 64 | 65 | Strong | Yes | | 12 | Overcast | 72 | 90 | Strong | Yes | | 13 | Overcast | 81 | 75 | Weak | Yes |   All decisions are ‘Yes’  https://insightimi.files.wordpress.com/2020/03/image-7-2.png?w=1024  **Outlook = Rain**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Day** | **Outlook** | **Temp.** | **Humidity** | **Wind** | **Decision** | | 4 | Rain | 70 | 96 | Weak | Yes | | 5 | Rain | 68 | 80 | Weak | Yes | | 6 | Rain | 65 | 70 | Strong | No | | 10 | Rain | 75 | 80 | Weak | Yes | | 14 | Rain | 71 | 80 | Strong | No |   If Wind = Weak, decision is ‘No’  If Wind = Strong, decision is ‘Yes’  https://insightimi.files.wordpress.com/2020/03/image-8-2.png?w=1024  So, this is our final Decision Tree using C4.5 algorithm.  **Advantages of C4.5 over ID3**  C4.5 is an evolution of ID3 by the same author (Quinlan). He made sure that the bottlenecks are not repeated in C4.5 that were present in ID3. Following are the improvements he made in C4.5  1. It can handle both continuous and discrete variables.  2. It can handle missing values by marking them as ‘?’. They are not used in Gain and Entropy calculations.  3. Prunes the tree and thereby avoids ‘overfitting’. |

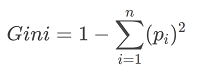
**CART Algorithm**

Classification and Regression Trees (CART) is a decision tree algorithm that is used for both classification and regression tasks. It is a supervised learning algorithm that learns from labelled data to predict unseen data.

* **Tree structure:** CART builds a tree-like structure consisting of nodes and branches. The nodes represent different decision points, and the branches represent the possible outcomes of those decisions. The leaf nodes in the tree contain a predicted class label or value for the target variable.
* **Splitting criteria:** CART uses a greedy approach to split the data at each node. It evaluates all possible splits and selects the one that best reduces the impurity of the resulting subsets.

For **classification tasks**, CART uses **Gini impurity or Gini index** as the splitting criterion. The lower the Gini impurity, the more pure the subset is.

The formula for Gini Index is as per the following:



where pi is the probability of a thing having a place with a specific class.

For **regression tasks**, CART uses **residual reduction** as the splitting criterion. The lower the residual reduction, the better the fit of the model to the data.

* **Pruning:** pruning is a technique used to remove the nodes that contribute little to the model accuracy.

To prevent overfitting (Overfitting happens due to several reasons, such as: • The training data size is too small and does not contain enough data samples to accurately represent all possible input data values. )of the data,

**Cost complexity pruning** and **information gain pruning** are two popular pruning techniques. Cost complexity pruning involves calculating the cost of each node and removing nodes that have a negative cost. Information gain pruning involves calculating the information gain of each node and removing nodes that have a low information gain.

**How does CART algorithm works?**

The CART algorithm works via the following process:

* The best-split point of each input is obtained.
* Based on the best-split points of each input in Step 1, the new “best” split point is identified.
* Split the chosen input according to the “best” split point.
* Continue splitting until a stopping rule is satisfied or no further desirable splitting is available.

