K Nearest Neighbours

KNN Majority Vote Activity

- Consider the training set:

Instance / Object	a_1	a_2	y (label / class)
x_1	2	2	-1
x_2	1.5	1	-1
x_3	4.5	3	+1
x_4	4	4	-1
x_5	5	4.5	+1
x_6	3	6	+1

- Predict the label for the test object $x = (4.5, 5)$
- Problem: Use KNN to classify x when $k = 3$
- Distance Metric: Euclidean
- Step 1: Compute Distance

$$d(x, x_1) = \sqrt{(4.5-2)^2 + (5-2)^2} = \sqrt{2.5^2 + 3^2} = \sqrt{16.25} \approx 3.91$$

$$d(x, x_2) = \sqrt{(4.5-1.5)^2 + (5-1)^2} = \sqrt{3^2 + 4^2} = \sqrt{25} = 5$$

$$d(x, x_3) = \sqrt{(4.5-4.5)^2 + (5-3)^2} = \sqrt{0^2 + 2^2} = \sqrt{4} = 2$$

$$d(x, x_4) = \sqrt{(4.5-4)^2 + (5-4)^2} = \sqrt{0.5^2 + 1^2} = \sqrt{1.25} \approx 1.12$$

$$d(x, x_5) = \sqrt{(4.5-5)^2 + (5-4.5)^2} = \sqrt{(-0.5)^2 + 0.5^2} = \sqrt{0.5} \approx 0.71$$

$$d(x, x_6) = \sqrt{(4.5-3)^2 + (5-6)^2} = \sqrt{1.5^2 + (-1)^2} = \sqrt{3.25} \approx 1.8$$

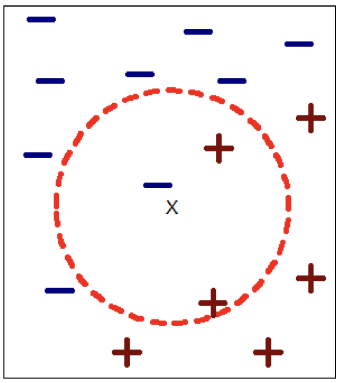
- Step 2: Identify 3 Nearest Neighbours $\{x_4, x_5, x_6\}$
- Step 3: Classify x (Major vote): +1

Improving KNN Efficiency

- Avoid having to compute distance to all objects in the training set
  - Multi-dimensional access methods (k-d trees)
- Condensing
  - Determine a smaller set of objects that give the same performance
  - Editing
  - Remove objects to improve efficiency

Potential Issue with Majority Vote

- Determine the class from nearest neighbour list
  - Take the majority vote of class labels among the k-nearest neighbours
  - A potential issue is that all the points have the same weight



Distance-Weighted Voting

- In the majority vote, each neighbour has the same influence, making KN sensitive to the choice of k.
- For distance-weighted voting:
  - Weight the influence of x_i according to its distance: $w_i = \frac{1}{d(x,x_i)^2}$
  - Training objects far away from the test sample x have a weaker impact
  - In this case, the label assigned to x is the label that corresponds to the points whose sum of weights is the largest

Weighted KNN Activity

- Consider the training set:

Instance / Object	a_1	a_2	y (label / class)
x_1	2	2	-1
x_2	1.5	1	-1
x_3	4.5	3	+1
x_4	4	4	-1
x_5	5	4.5	+1
x_6	3	6	+1

- Predict the label for the test object $x = (4.5, 5)$
- Problem: Use a weighted KNN to classify x when $k = 3$
- Distance Metric: Euclidean

- Step 1: Compute Distance

$$d(x, x_1) = \sqrt{(4.5-2)^2 + (5-2)^2} = \sqrt{2.5^2 + 3^2} = \sqrt{16.25} \approx 3.91$$

$$d(x, x_2) = \sqrt{(4.5-1.5)^2 + (5-1)^2} = \sqrt{3^2 + 4^2} = \sqrt{25} = 5$$

$$d(x, x_3) = \sqrt{(4.5-4.5)^2 + (5-3)^2} = \sqrt{0^2 + 2^2} = \sqrt{4} = 2$$

$$d(x, x_4) = \sqrt{(4.5-4)^2 + (5-4)^2} = \sqrt{0.5^2 + 1^2} = \sqrt{1.25} \approx 1.12$$

$$d(x, x_5) = \sqrt{(4.5-5)^2 + (5-4.5)^2} = \sqrt{(-0.5)^2 + 0.5^2} = \sqrt{0.5} \approx 0.71$$

$$d(x, x_6) = \sqrt{(4.5-3)^2 + (5-6)^2} = \sqrt{1.5^2 + (-1)^2} = \sqrt{3.25} \approx 1.8$$

- Step 2: Identify 3 Nearest Neighbours $\{x_4, x_5, x_6\}$
- Step 3: Calculated the weighted sum of the neatest neighbours

Object	$w_i = \frac{1}{d(x,x_i)^2}$
x_4	0.80
x_5	1.98
x_6	0.31

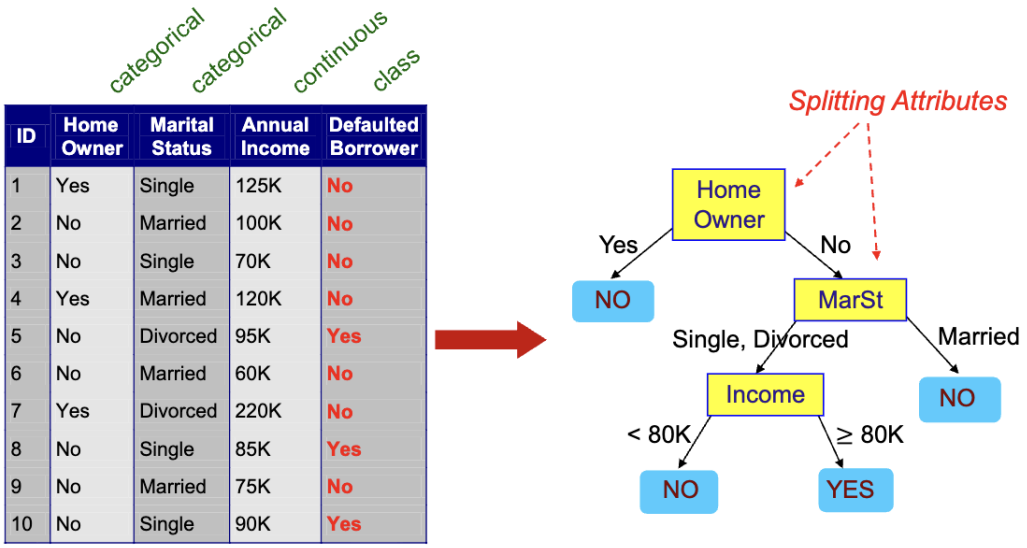
$$(-1) = w_4 = 0.8$$

$$(+1) = w_5 + w_6 = 1.98 + 0.31 = 2.29$$

- Step 4: Classify x as the label with the largest sum: $+1$

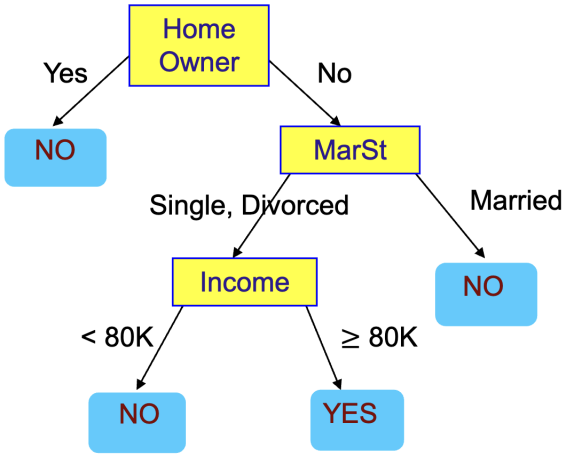
Decision Trees

- We want to be able to turn training data into a decision tree that can be followed to classify new data



Applying Trees to Test Data

- Consider the tree

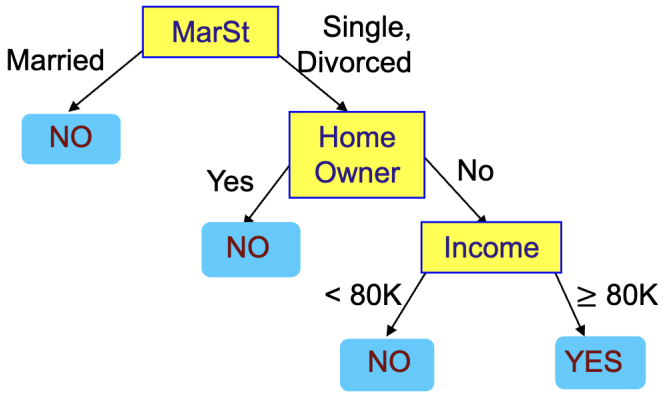


- We will use this tree to classify the test data:

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

- First, we begin at the root of the tree
- Since our test data is not a home owner, we go to the right
- Then, our test data is married, so we go to the right once again
- Now, we have reached a leaf labeled "No"
- So, we assign the label (Defaulted) to "No"

Another Example of a Decision Tree



- There could be more than one tree that fits the same data

Terminology

- Root Note: Does not have parents and sits at the top of the tree

- Internal Node: Has a parent and at least one child
- Leaf Node: Has a parent but does not have children

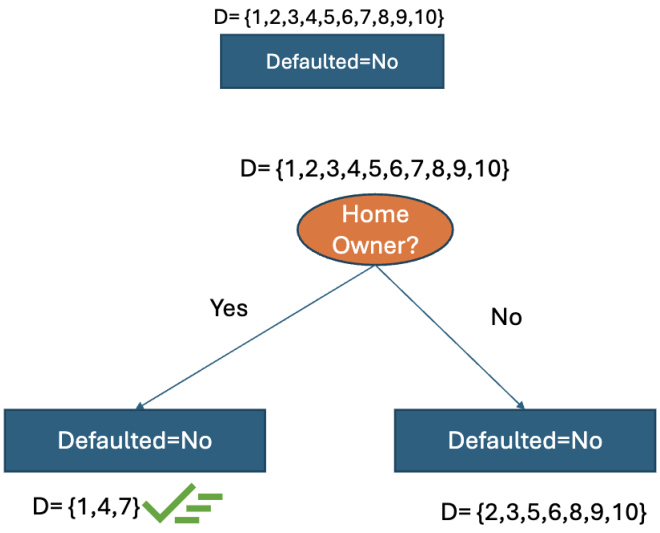
Hunt's Algorithm

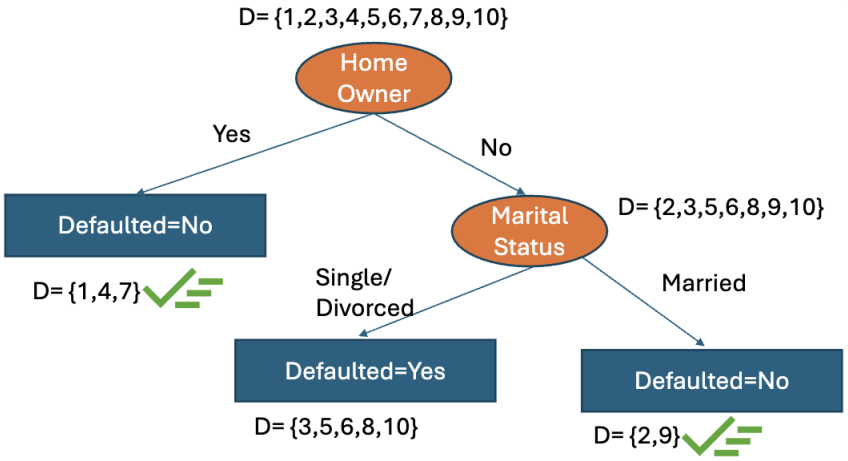
- Step 1: If D_t contains instances that all belong to the same class y_t, then t is a leaf node labeled and y_t
- Step 2: If D_t contains instances that belong to more than one class, use an attribute test condition to split the data into smaller subsets (creating children)
- Step 3: Recursively apply the procedure to each subset

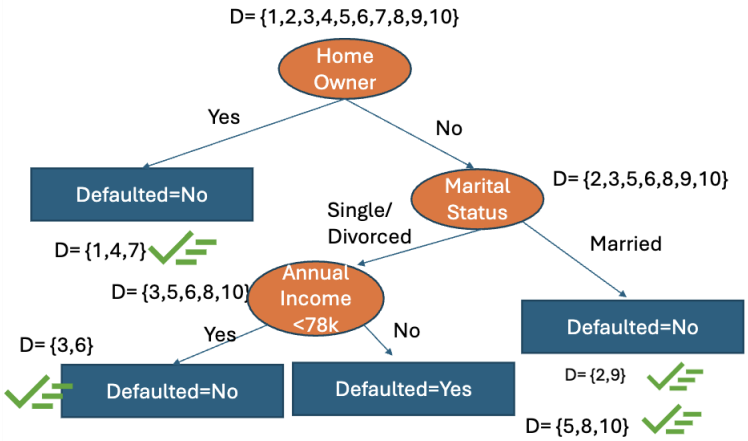
Applying Hunt's Algorithm

- Consider the data

ID	Home Owner	Marital Status	Annual Income	Defaulted on Loan
1	Yes	Single	125k	No
2	No	Married	100k	No
3	No	Single	70k	No
4	Yes	Married	120k	No
5	No	Divorced	95k	Yes
6	No	Single	60k	No
7	Yes	Divorced	220k	No
8	No	Single	85k	Yes
9	No	Married	75k	No
10	No	Single	90k	Yes



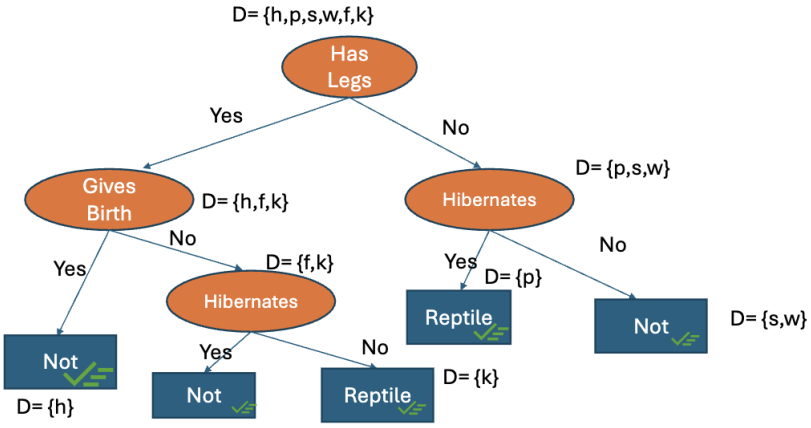




Classifying Vertebrates Exercise (Reptile / Not)

Name	Body Temperature	Skin Cover	Gives Birth	Aerial Creature	Has Legs	Class Label
Human	Warm-blooded	Hair	Yes	No	Yes	Not
Python	Cold-blooded	Scales	No	No	No	Reptile
Salmon	Cold-blooded	Scales	No	No	No	Not
Whale	Warm-blooded	Hair	Yes	No	No	Not
Frog	Cold-blooded	None	No	No	Yes	Not
Komodo	Cold-blooded	Scales	No	No	Yes	Reptile

- A potential solution would be (Note this solution uses a version of the table with additional attributes):


D= {h,p,s,w,f,k}
Has Legs
Yes
No
Gives Birth
D= {h,f,k}
Hibernates
D= {p,s,w}
Yes
No
No
Yes
D= {f,k}
Yes
D= {p}
Hibernates
Reptile
Not
D= {s,w}
Not
Yes
No
Not
Reptile
D= {k}
D= {h}