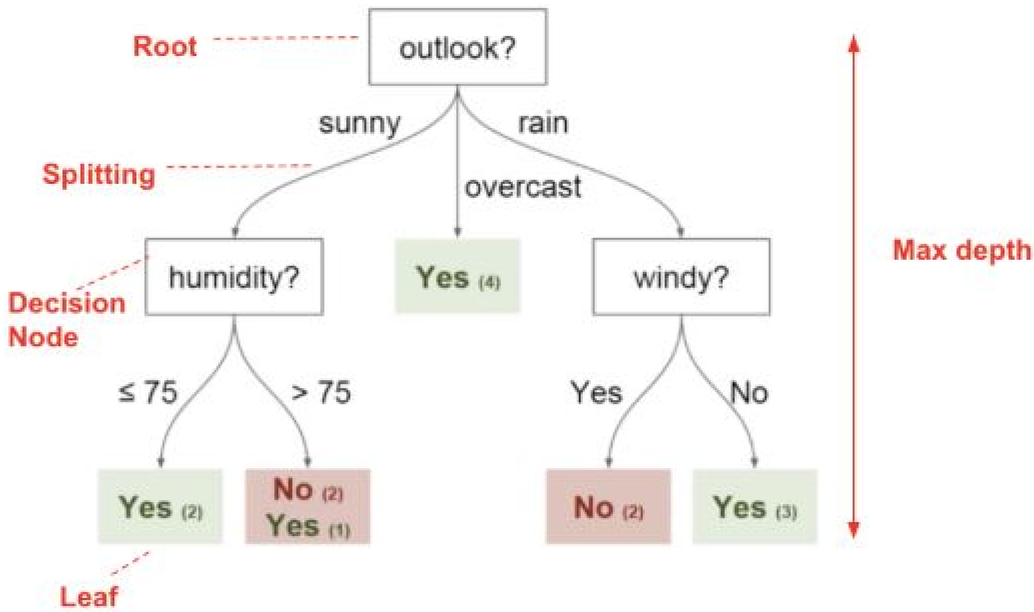
DECISION

TREES

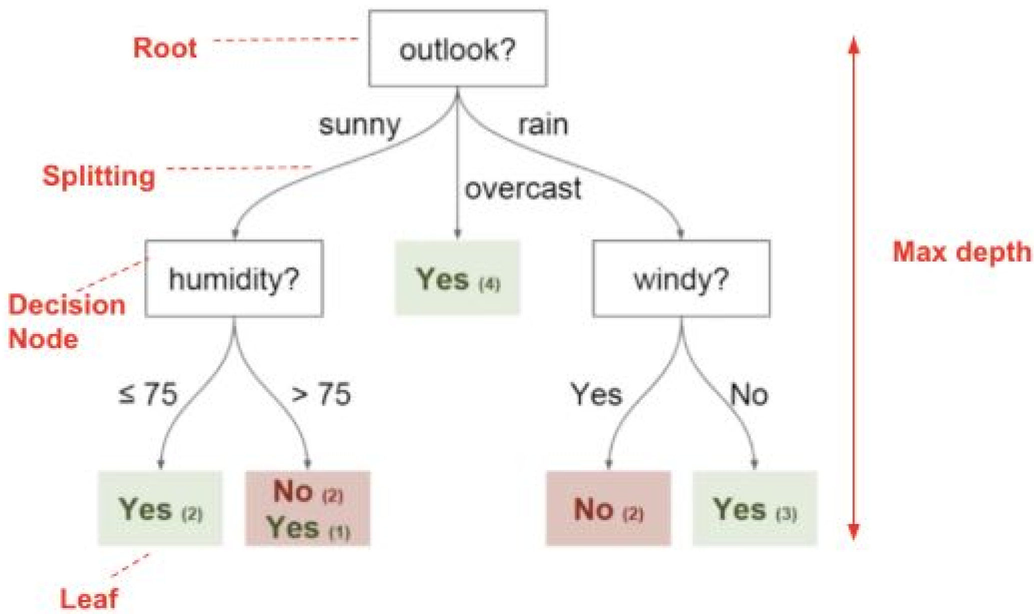
Decision Trees



- inputs and outputs
- want classification for labels
- decision tree classifier

figure reprinted from www.kdnuggets.com with explicit permission of the editor

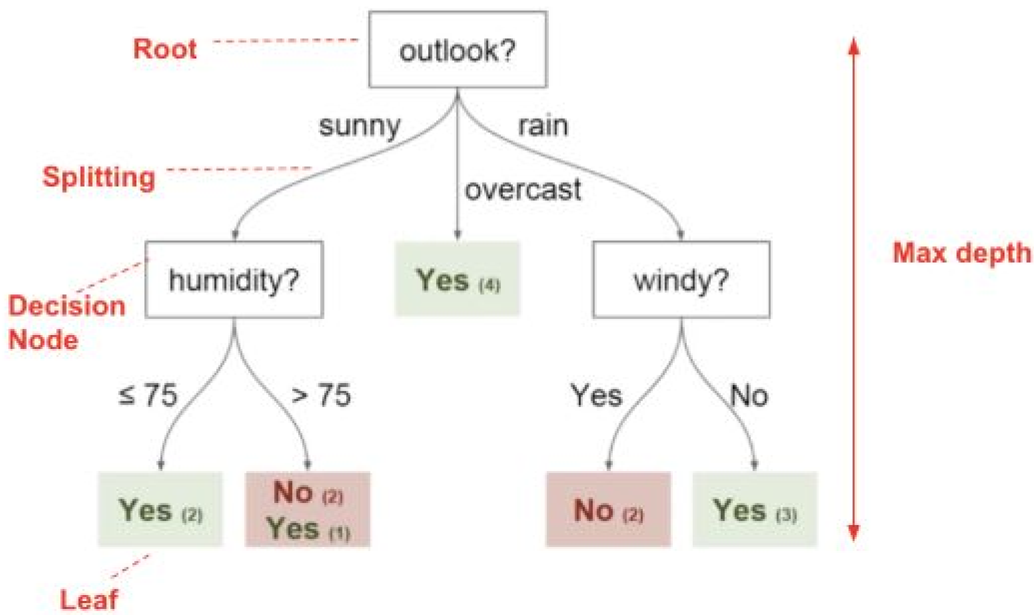
Decision Trees



- supervised learning
- constructed based on information gain
- classification or prediction

figure reprinted from www.kdnuggets.com with explicit permission of the editor

Representation

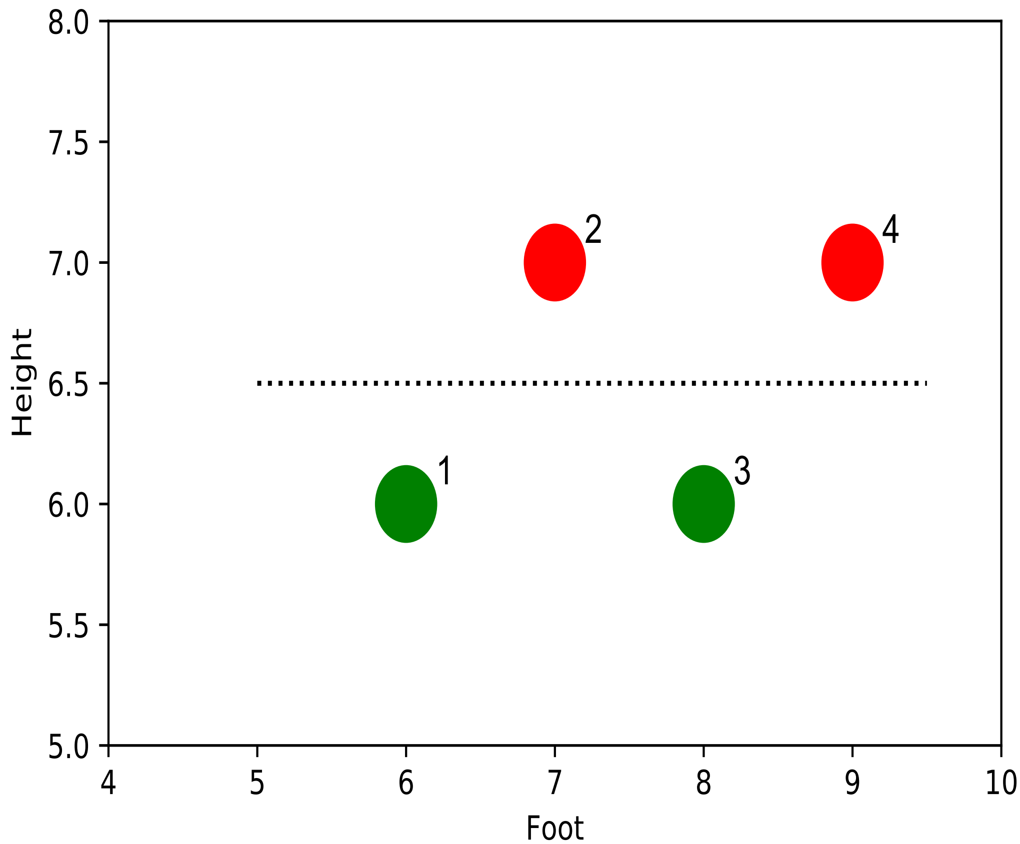


1. root node (initial test)
2. interior nodes (testing)
3. leaf nodes (predictions)

- edges are test outcomes

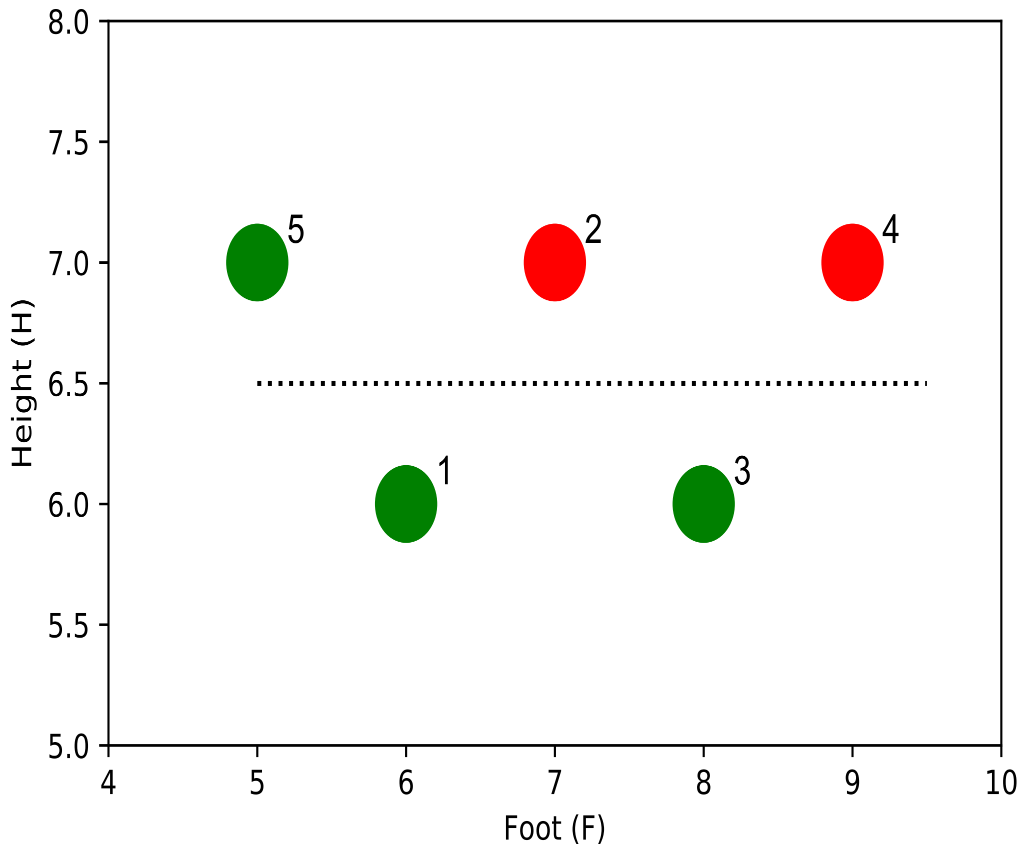
figure reprinted from www.kdnuggets.com with explicit permission of the editor

A Trivial Example



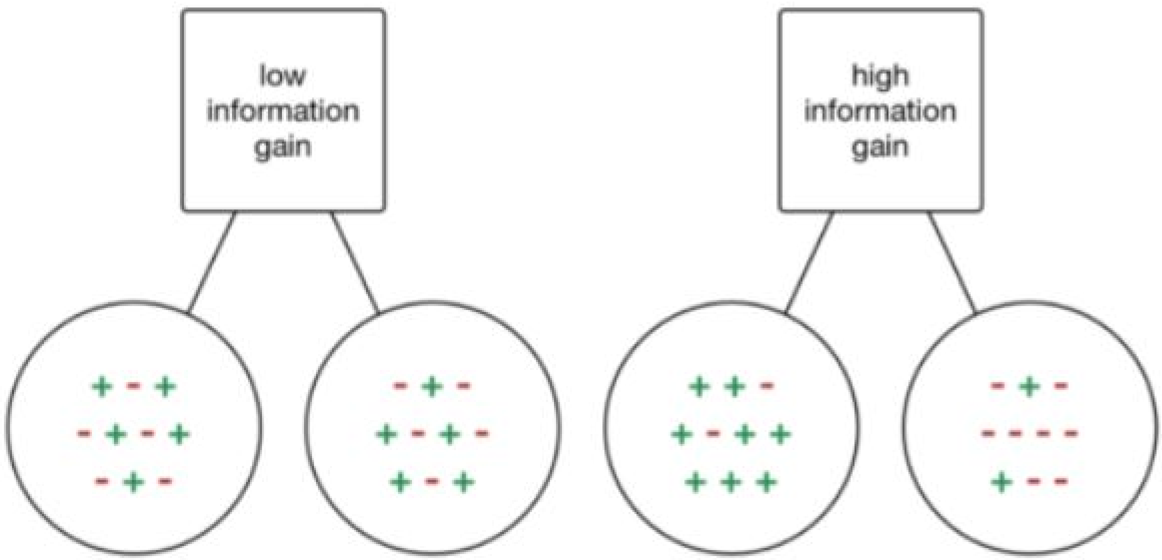
- a simple decision by height(H)

A Less Trivial Example



- decisions using both H & W

Constructing Tree



- want to grow simple trees
- each successor as pure as possible
- how: use "information gain" (defined by entropy)

figure reprinted from www.kdnuggets.com with explicit permission of the editor

Entropy

- measure of uncertainty
- $p_1, \ldots, p_n$ possible probabilities

$$H = -(p_1 \log_2 p_1 + \cdots + p_n \log_2 p_n)$$

Example 1: $n = 2, p_1 = 1, p_2 = 0$

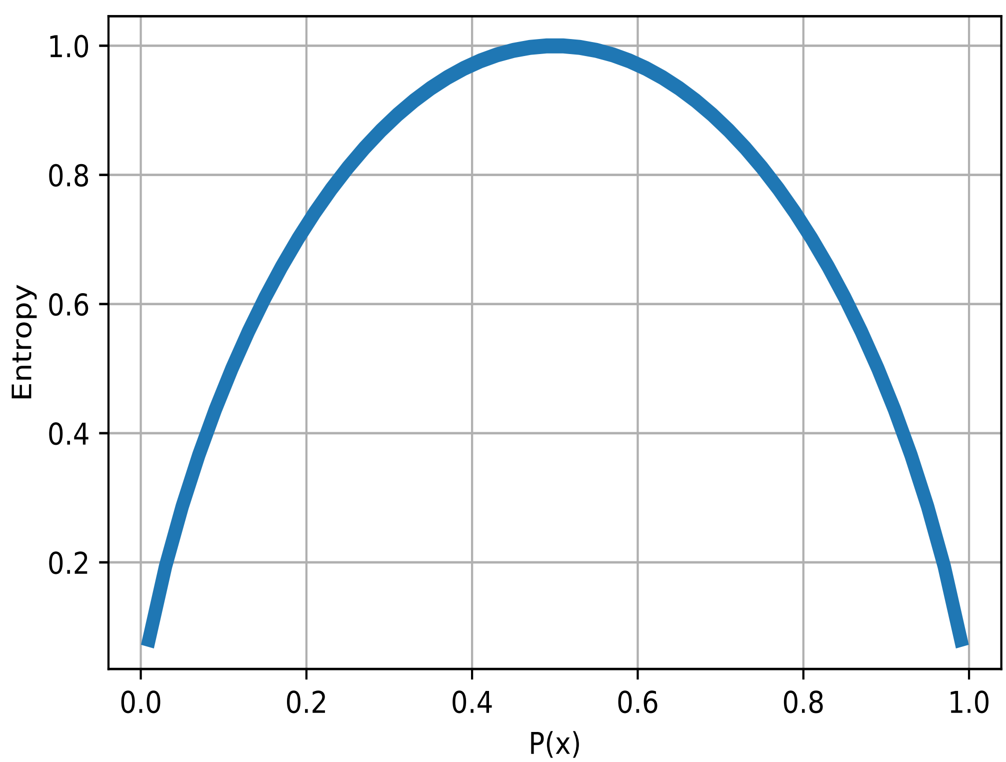
$$H_1 = -(\log_2 1 + 0) = 0$$

Example 2: $n = 2, p_1 = p_2 = 1/2$

$$H_2 = -(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}) = 1$$

Entropy vs. p

- $n = 2, p_1 = p, p_2 = 1 - p$
- H is maximized at $p = 0.5$



Entropy & Gini

1. entropy

$$H = -(p_1 \log_2 p_1 + \cdots + p_n \log_2 p_n)$$

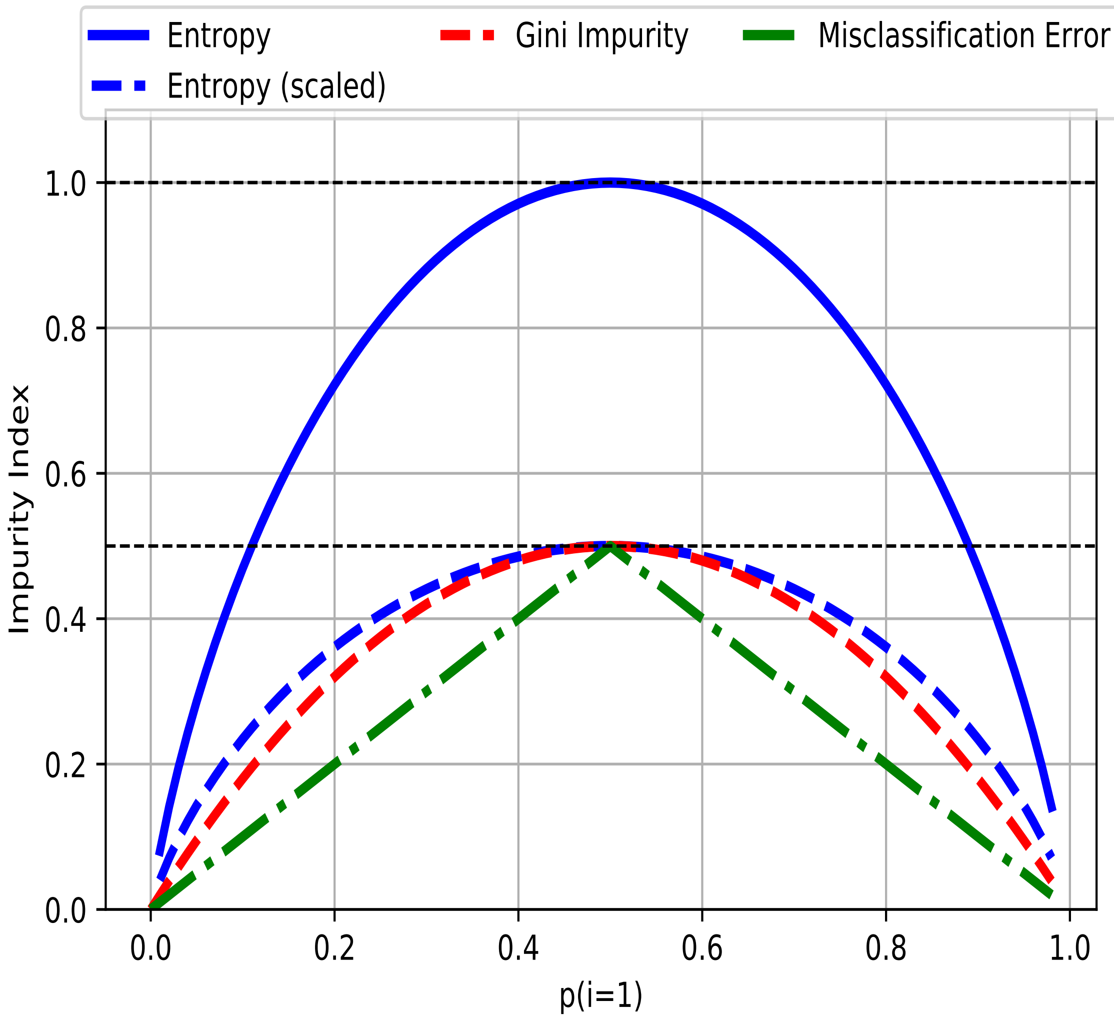
   - $H = 0$: all in one class
   - $H = 1$: uniform distribution

2. gini impurity

   - for binary classification
   - requires ranking
   - a measure of mis-classification

$$\begin{aligned}\text{Gini} &= p_1(1 - p_1) + \cdots + p_n(1 - p_n) \\ &= 1 - (p_1^2 + \cdots + p_n^2)\end{aligned}$$

Gini vs. Entropy



Information Gain

Day	Weather	Temperature	Wind	Play
1	sunny	hot	low	**no**
2	rainy	mild	high	**yes**
3	sunny	cold	low	**yes**
4	rainy	cold	high	**no**
5	sunny	cold	high	**yes**
6	overcast	mild	low	**yes**
7	sunny	hot	low	**yes**
8	overcast	hot	high	**yes**
9	rainy	hot	high	**no**
10	rainy	mild	low	**yes**

- P(Play = yes) = 0.7, P(Play = no) = 0.3

  $$\begin{aligned} H(\text{Play}) &= -0.7 \cdot \log_2(0.7) - 0.3 \cdot \log_2(0.3) \\ &= 0.8812 \end{aligned}$$

- what label for x* = (sunny, cold, low)?

Split on Weather

Day	Weather	Temperature	Wind	Play
1	sunny	hot	low	**no**
3	sunny	cold	low	**yes**
5	sunny	cold	high	**yes**
7	sunny	hot	low	**yes**

- $\text{P}(\text{Weather} = \text{sunny}) = 0.4$
- $\text{P}(\text{Play} = \text{yes}) = 0.75$
- $\text{P}(\text{Play} = \text{no}) = 0.25$

Split on Weather (cont'd)

Day	Weather	Temperature	Wind	Play
2	rainy	mild	high	**yes**
4	rainy	cold	high	**no**
9	rainy	hot	high	**no**
10	rainy	mild	low	**yes**

- P(Weather = rainy) = 0.4
- P(Play = yes) = 0.5
- P(Play = no) = 0.5

Split on Weather (cont'd)

Day	Weather	Temperature	Wind	Play
6	overcast	mild	low	**yes**
8	overcast	hot	high	**yes**

- P(Weather = overcast) = 0.2
- P(Play = yes) = 1
- P(Play = no) = 0

Weather I-Gain

$$\begin{aligned} H(W) = &- 0.4(0.75 \cdot \log_2 0.75 + 0.25 \cdot \log_2 0.25) \\ &- 0.4(0.5 \cdot \log_2 0.5 + 0.5 \cdot \log_2 0.5) \\ &- 0.2(\log_2 1 + 0) \\ =& 0.7245 \end{aligned}$$

- recall H(Play) = 0.8812
- information gain of splitting by weather:

$$\begin{aligned} \text{I-Gain}(W) &= H(Play) - H(W) \\ &= 0.8812 - 0.7245 \\ &= 0.1567 \end{aligned}$$

Split on Temperature

Day	Weather	Temperature	Wind	Play
1	sunny	hot	low	**no**
7	sunny	hot	low	**yes**
8	overcast	hot	high	**yes**
9	rainy	hot	high	**no**

- $\text{P}(\text{Temperature} = \text{hot}) = 0.4$
- $\text{P}(\text{Play} = \text{yes}) = 0.5$
- $\text{P}(\text{Play} = \text{no}) = 0.5$

Split on Temperature (cont'd)

Day	Weather	Temperature	Wind	Play
2	rainy	mild	high	**yes**
6	overcast	mild	low	**yes**
10	rainy	mild	low	**yes**

- P(temperature = mild) = 0.3
- P(Play = yes) = 1
- P(Play = no) = 0

Split on Temperature (cont'd)

Day	Weather	Temperature	Wind	Play
3	sunny	cold	low	**yes**
4	rainy	cold	high	**no**
5	sunny	cold	high	**yes**

- P(temperature = cold) = 0.3
- P(Play = yes) = 2/3
- P(Play = no) = 1/3

Temperature I-Gain

$$
\begin{aligned}
H(T) = & -0.4\left(\frac{1}{2}\cdot\log_2\frac{1}{2}+\frac{1}{2}\cdot\log_2\frac{1}{2}\right)\\
& -0.3\cdot 0\\
& -0.3\left(\frac{2}{3}\cdot\log_2\frac{2}{3}+\frac{1}{3}\cdot\log_2\frac{1}{3}\right)\\
=& 0.68
\end{aligned}
$$

- recall H(Play) = 0.88
- information gain of splitting by temperature

$$
\begin{aligned}
\text{I-Gain}(Temperature) &= H(Play) - H(W)\\
&= 0.88 - 0.68\\
&= 0.20
\end{aligned}
$$

Split on Wind

Day	Weather	Temperature	Wind	Play
1	sunny	hot	low	**no**
3	sunny	cold	low	**yes**
6	overcast	mild	low	**yes**
7	sunny	hot	low	**yes**
10	rainy	mild	low	**yes**

- $\text{P}(\text{Wind} = \text{low}) = 0.5$
- $\text{P}(\text{Play} = \text{yes}) = 0.8$
- $\text{P}(\text{Play} = \text{no}) = 0.2$

Split on Wind (cont'd)

Day	Weather	Temperature	Wind	Play
2	rainy	mild	high	**yes**
4	rainy	cold	high	**no**
5	sunny	cold	high	**yes**
8	overcast	hot	high	**yes**
9	rainy	hot	high	**no**

- P(Wind = high) = 0.5
- P(Play = yes) = 0.6
- P(Play = no) = 0.4

Wind I-Gain

$$\begin{aligned} H(Wind) = & -0.5(0.8 \cdot \log_2 0.8 + 0.2 \cdot \log_2 0.2) \\ & -0.5(0.6 \cdot \log_2 0.6 + 0.4 \cdot \log_2 0.4) \\ = & 0.85 \end{aligned}$$

- recall H(Play) = 0.88
- information gain of splitting by wind

$$\begin{aligned} \text{I-Gain}(Wind) &= H(Play) - H(Wind) \\ &= 0.88 - 0.85 \\ &= 0.03 \end{aligned}$$

Computation of I-Gain

- split according to feature A
- compute entropy of A
- compare with total entropy H
- choose feature that reduces H the most

Feature	(weighted) Entropy	I-Gain
Play	0.88	——
Weather	0.72	0.16
Temparature	0.68	0.20
Wind	0.85	0.03

- split by wind reduces entropy the most
- should be used as root node

Classification With Trees

Day	Weather	Temperature	Wind	Play
1	sunny	hot	low	**no**
2	rainy	mild	high	**yes**
3	sunny	cold	low	**yes**
4	rainy	cold	high	**no**
5	sunny	cold	high	**yes**
6	overcast	mild	low	**yes**
7	sunny	hot	low	**yes**
8	overcast	hot	high	**yes**
9	rainy	hot	high	**no**
10	rainy	mild	low	**yes**

- $x^* = (\text{sunny, cold, low}) \mapsto ?$
- need numeric values for attributes

Change to Dummy Variables

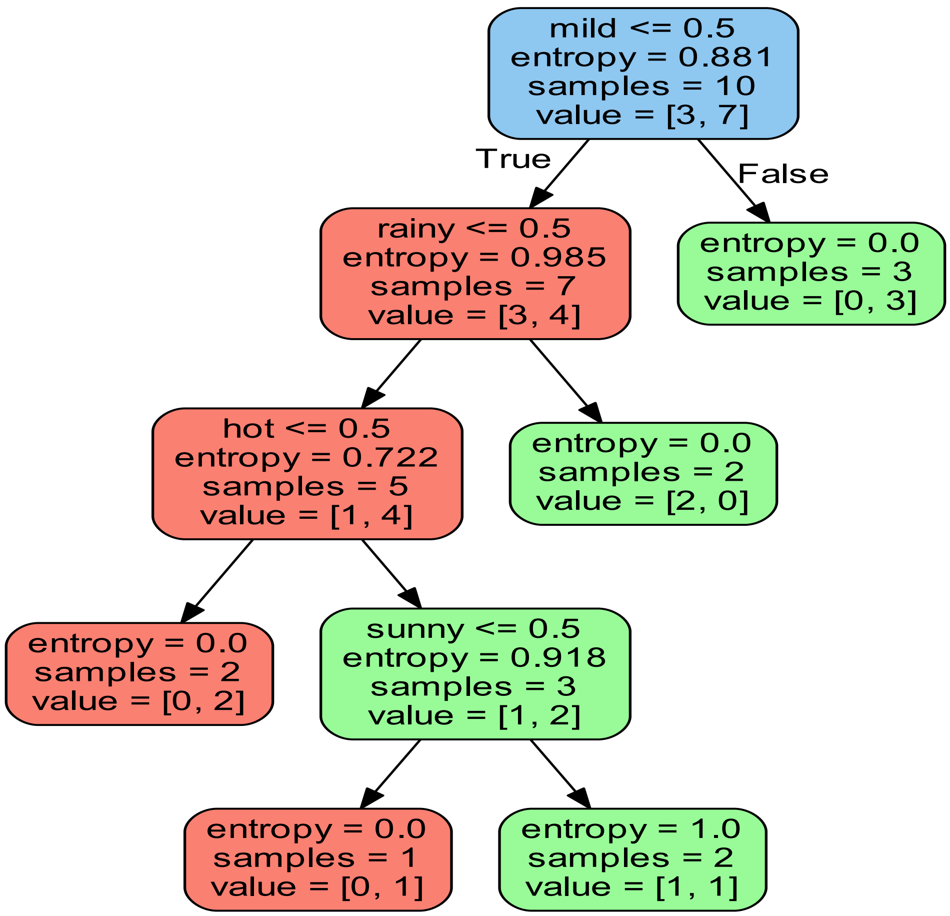
Day	Weather			Temp.			Wind	
	overcast	rainy	sunny	cold	hot	mild	high	low
1	0	0	1	0	1	0	0	1
2	0	1	0	0	0	1	1	0
3	0	0	1	1	0	0	0	1
4	0	1	0	1	0	0	1	0
5	0	0	1	1	0	0	1	0
6	1	0	0	0	0	1	0	1
7	0	0	1	0	1	0	0	1
8	1	0	0	0	1	0	1	0
9	0	1	0	0	1	0	1	0
10	0	1	0	0	0	1	0	1

Python Code

```
import numpy as np
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder

data = pd.DataFrame(
        {'Day':         [1,2,3,4,5,6,7,8,9,10],
        'Weather':      ['sunny','rainy','sunny','rainy',
                         'sunny','overcast','sunny','overcast',
                         'rainy','rainy'],
        'Temperature': ['hot', 'mild', 'cold','cold','cold',
                         'mild','hot','hot', 'hot','mild'],
        'Wind':         ['low','high','low','high','high',
                         'low','low', 'high','high','low'],
        'Play':         ['no', 'yes','yes','no','yes',
                         'yes','yes','yes','no','yes']},
        columns = ['Day','Weather','Temperature','Wind','Play'])

input_data = data[['Weather', 'Temperature', 'Wind']]
dummies = [pd.get_dummies(data[c]) for c in input_data.columns]
binary_data = pd.concat(dummies, axis=1)
X = binary_data[0:10].values
le = LabelEncoder()
Y = le.fit_transform(data['Play'].values)
clf = tree.DecisionTreeClassifier(criterion='entropy', max_features=8)
clf = clf.fit(X,Y)

# sunny -> (0,0,1), cold-> (0,1,0), low -> (0,1)
new_instance = np.asmatrix([0,0,1,1,0,0,0,1])
prediction = clf.predict(new_instance)
```

```
ipdb> prediction[0]
1
```

A Decision Tree for Categorical Dataset



Using Decision Tree

- label for x* = (sunny, cold, low)?
- dummy x**=(0, 0, 1, 1, 0, 0, 0, 1)
- labels: 0 ("no") and 1 ("yes")
  (a) mild ≤ 0.5, we take the left branch
  (b) rainy ≤ 0.5, we take the left branch
  (c) high ≤ 0.5, we take the left branch
  (d) cold ≥ 0.5, we take the right branch
  (e) leaf node $\mapsto$ "yes"
- can get results in $O(\log(N)$ steps

A Numerical Dataset

object x_i	Height (H)	Weight (W)	Foot (F)	Label (L)
x_1	5.00	100	6	green
x_2	5.50	150	8	green
x_3	5.33	130	7	green
x_4	5.75	150	9	green
x_5	6.00	180	13	red
x_6	5.92	190	11	red
x_7	5.58	170	12	red
x_8	5.92	165	10	red

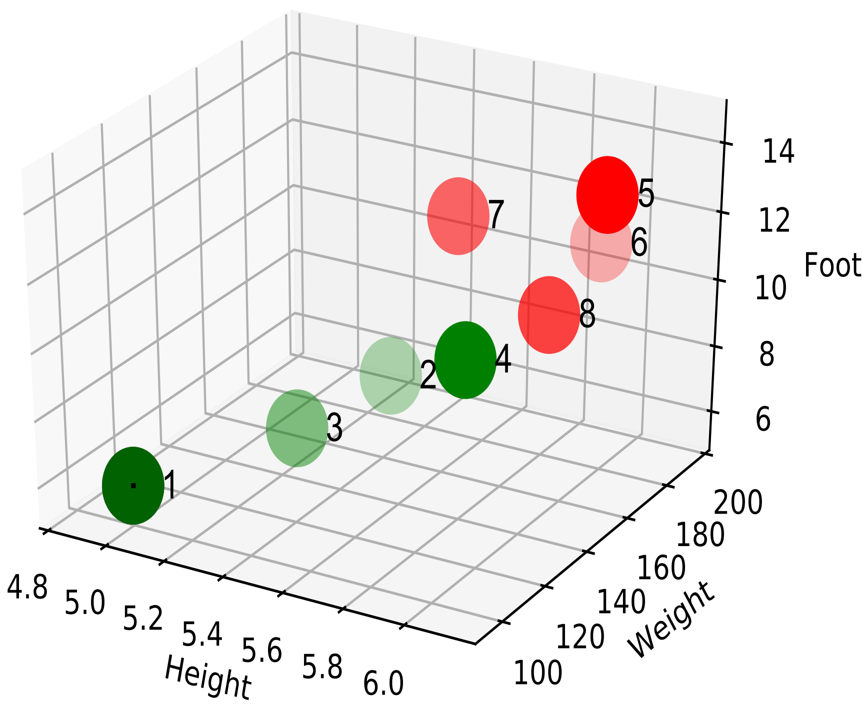
- $N = 8$ items
- $M = 3$ (unscaled) attributes

Code for the Dataset

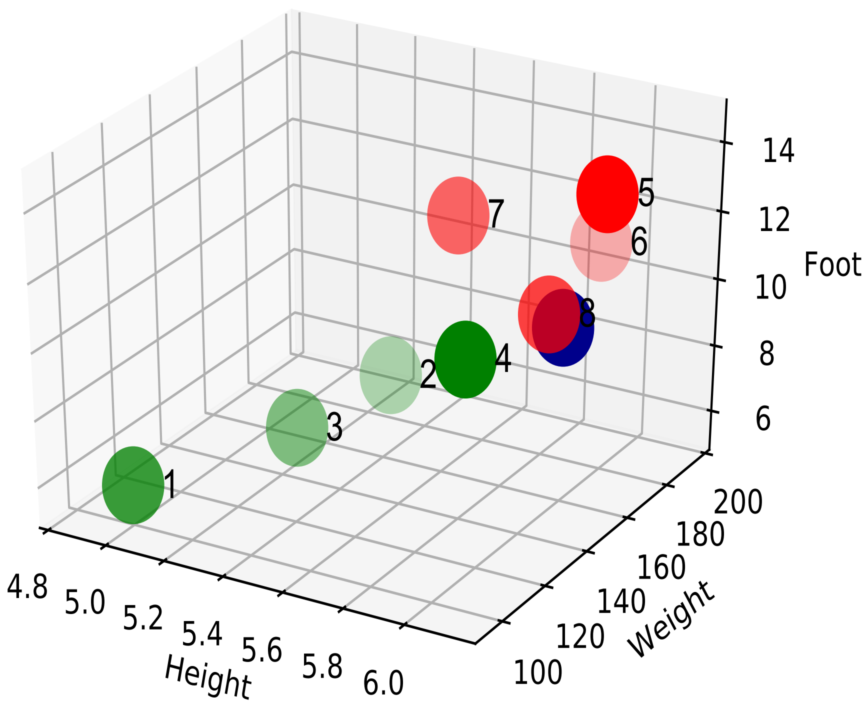
```
import pandas as pd
data = pd.DataFrame(
        {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green','green','green','green',
                   'red','red','red','red'],
        'Height': [5, 5.5, 5.33, 5.75,
                   6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                   180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                      'Foot', 'Label'] )
```

```
ipdb> data
  id Height Weight Foot Label
0  1   5.00    100    6 green
1  2   5.50    150    8 green
2  3   5.33    130    7 green
3  4   5.75    150    9 green
4  5   6.00    180   13   red
5  6   5.92    190   11   red
6  7   5.58    170   12   red
7  8   5.92    165   10   red
```

A Dataset Illustration

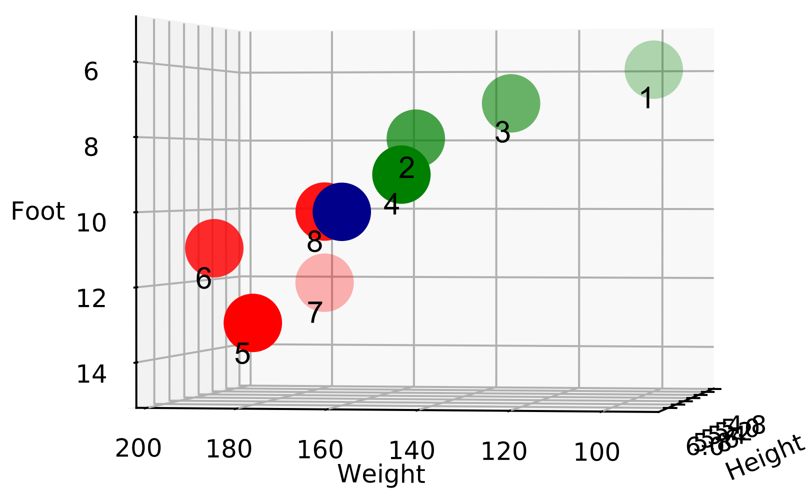


A New Instance



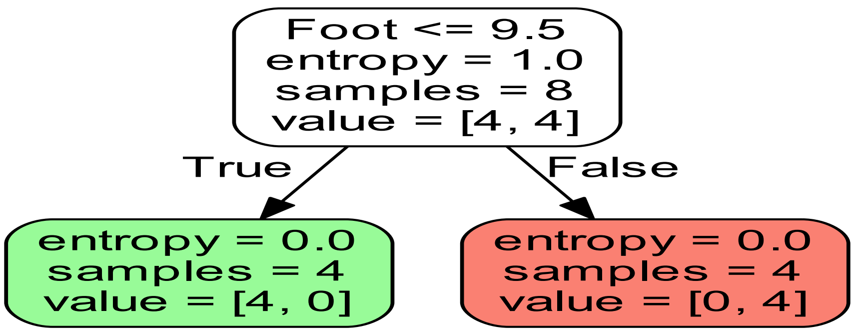
$$(\text{H}=6,\ \text{W}=160,\ \text{F}=10) \mapsto ?$$

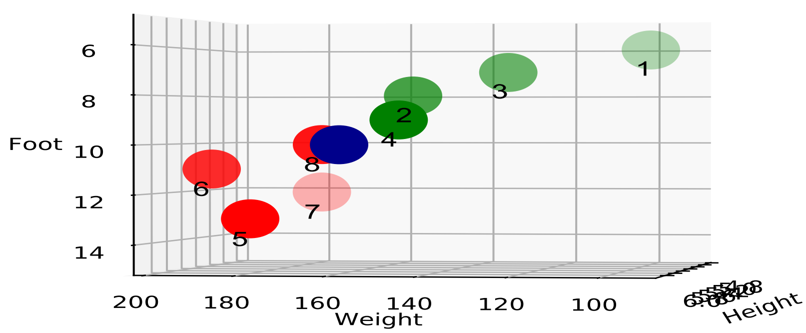
Decision Logic for Labels



- (H=6, W=160, F=10) $\mapsto$ red
- can decide by foot size

Decision Tree





$$(H{=}6,\ W{=}160,\ F{=}10) \mapsto \text{red}$$

Decision Tree in Python

```
import numpy as np
import pandas as pd
from sklearn import tree

data = pd.DataFrame( {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green', 'green', 'green', 'green',
                         'red', 'red', 'red', 'red'],
        'Height': [5, 5.5, 5.33, 5.75,
                            6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                                 180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                             'Foot', 'Label'] )

X = data[['Height', 'Weight', 'Foot']].values
Y = data[['Label']].values
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X,Y)

prediction = clf.predict(np.asmatrix([6, 160, 10]))
```

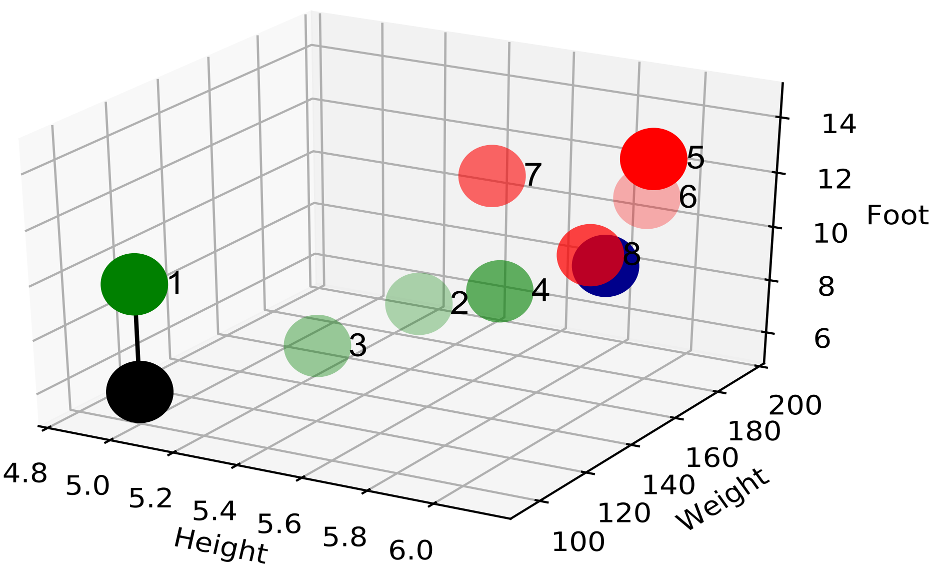
```
ipdb> prediction[0]
'red'
```

A Modified Dataset

- change foot size

id	Height	Weight	Foot	Label
1	5.00	100	$6 \mapsto 10$	green
2	5.50	150	8	green
3	5.33	130	7	green
4	5.75	150	9	green
5	6.00	180	13	red
6	5.92	190	11	red
7	5.58	170	12	red
8	5.92	165	10	red

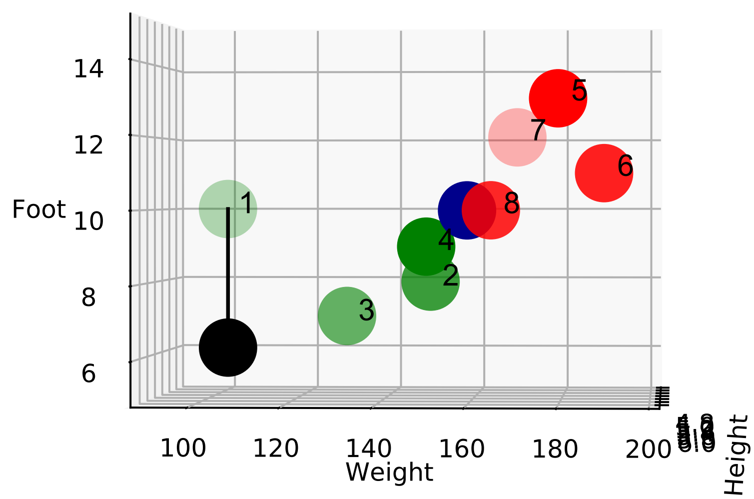
Foot Size Change



id	Height	Weight	Foot	Label
1	5	100	$6 \mapsto 10$	green

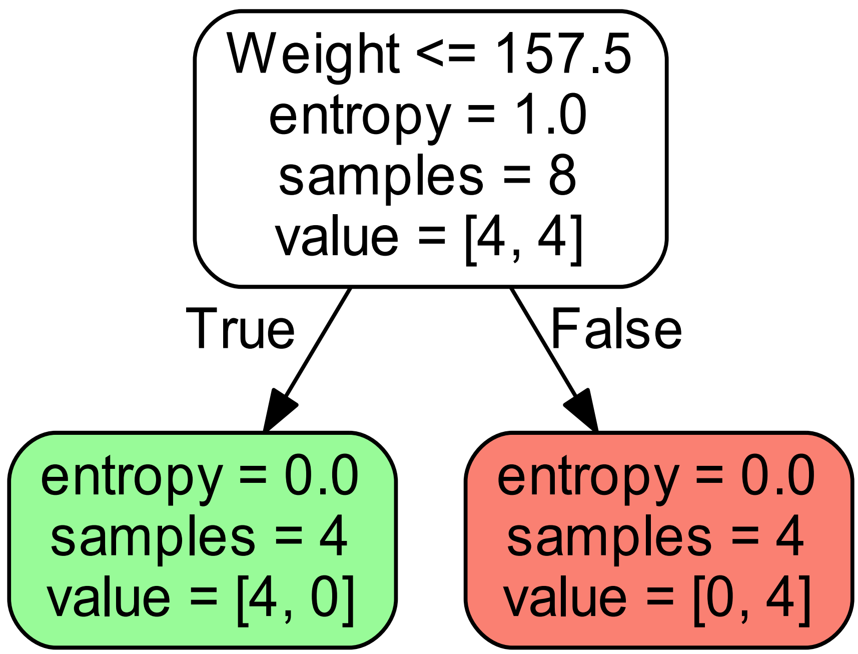
$$(H=6,\ W=160,\ F=10) \mapsto ?$$

Decision Logic for: Foot Size Change



- (H=6, W=160, F=10) $\mapsto$ red
- decide by weight, not by height

Decision Tree: Foot Size Change



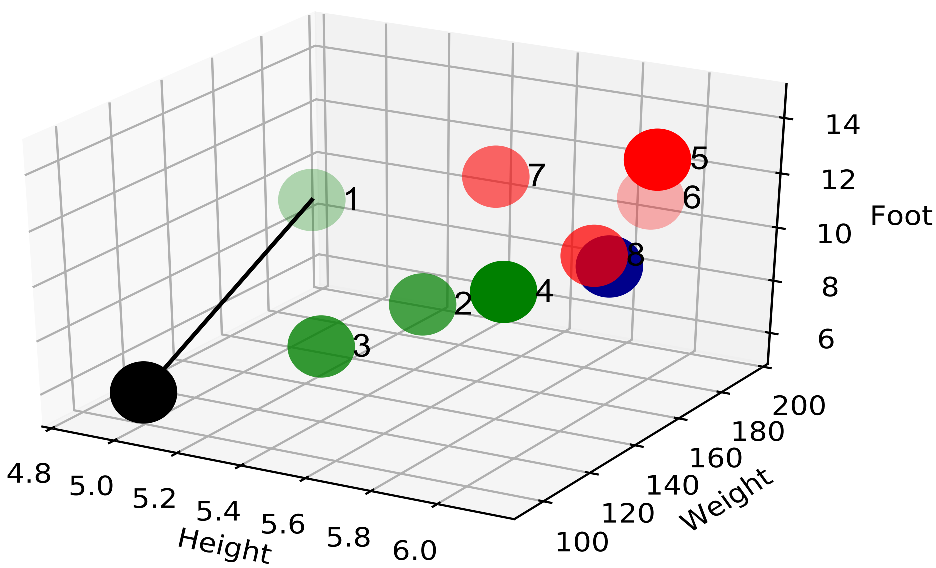
$$(\text{H}=6,\ \text{W}=160,\ \text{F}=10) \mapsto \text{red}$$

Code for Foot Change

```
import numpy as np
import pandas as pd
from sklearn import tree

data = pd.DataFrame( {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green', 'green', 'green', 'green',
                          'red', 'red', 'red', 'red'],
        'Height': [5, 5.5, 5.33, 5.75,
                             6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                                  180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                               'Foot', 'Label'] )

data['Foot'].iloc[1] = 10   # change foot from 6 to 10!!!

X = data[['Height', 'Weight', 'Foot']].values
Y = data[['Label']].values
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X,Y)

prediction = clf.predict(np.asmatrix([6, 160, 10]))
```

```
ipdb> prediction[0]
'red'
```

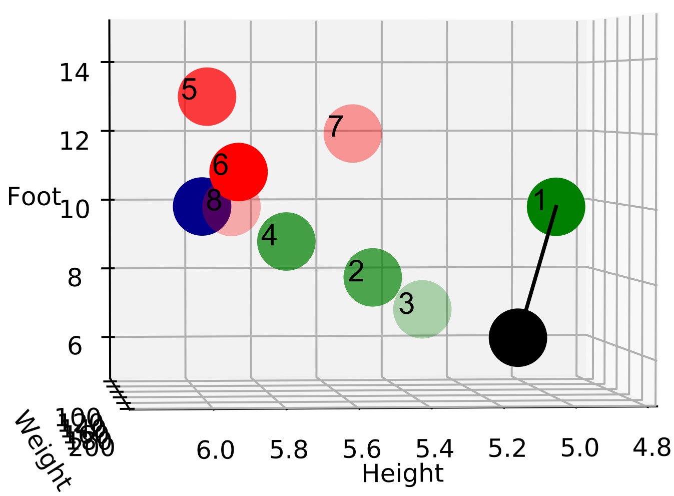
Foot/Weight Change



id	Height	Weight	Foot	Label
1	5	$100 \mapsto 170$	$6 \mapsto 10$	green

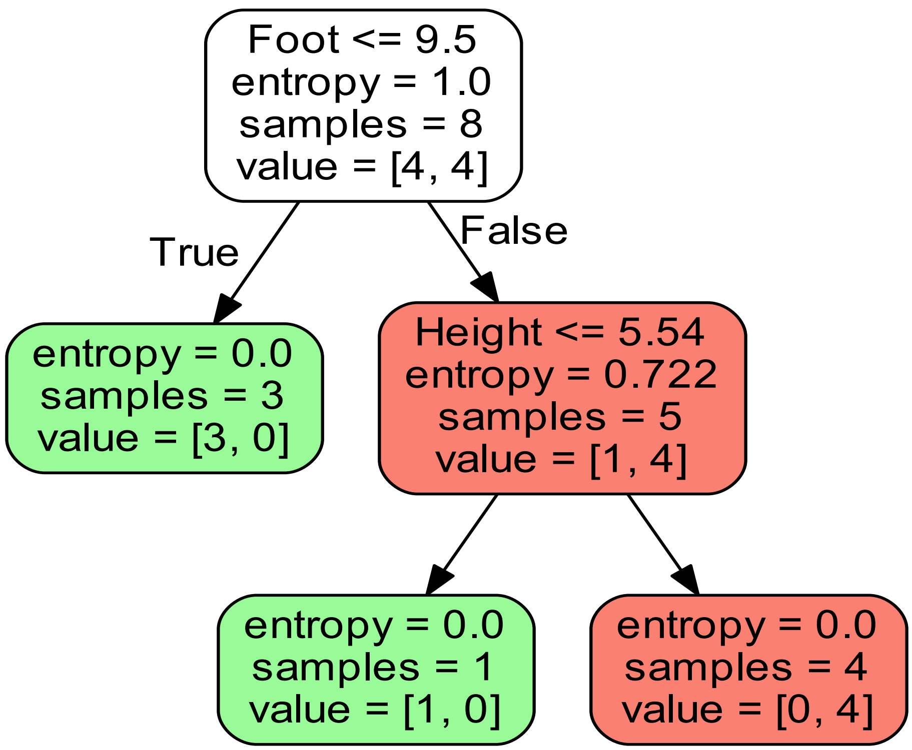
$$(\text{H}=6,\ \text{W}=160,\ \text{F}=10) \mapsto \ ?$$

Decision Logic for: F/W Change



- (H=6, W=160, F=10) $\mapsto$ green
- decide by foot and height

Decision Tree for F/W Change



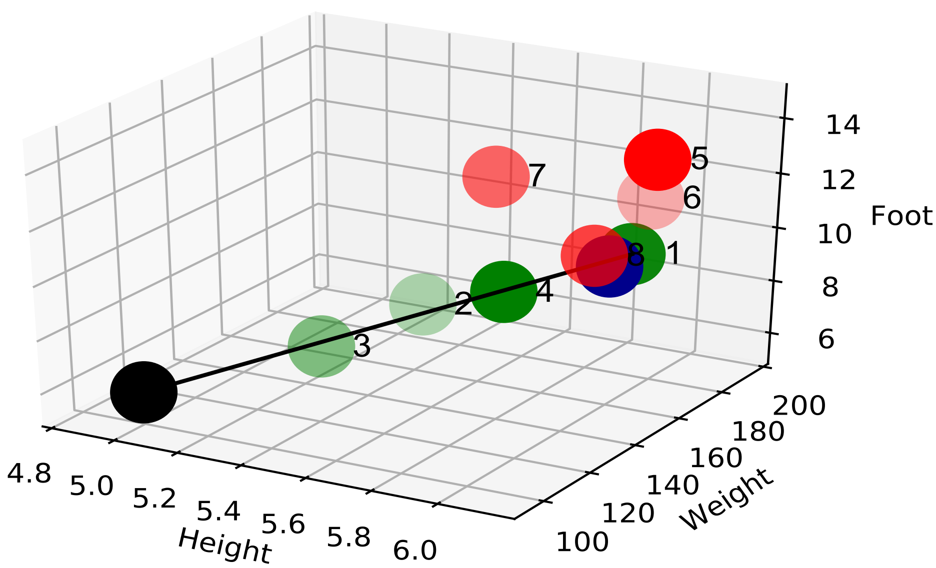
$$(\text{H=6,W=160,F=10}) \mapsto \text{green}$$

Code for F/W Change

```
import numpy as np
import pandas as pd
from sklearn import tree

data = pd.DataFrame( {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green', 'green', 'green', 'green',
                         'red', 'red', 'red', 'red'],
        'Height': [5, 5.5, 5.33, 5.75,
                            6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                                 180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                              'Foot', 'Label'] )

data['Foot'].iloc[1] = 10    # change foot from 6 to 10!
data['Weight'].iloc[1] = 170 # weight from 100 to 170

X = data[['Height', 'Weight', 'Foot']].values
Y = data[['Label']].values
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X,Y)
prediction = clf.predict(np.asmatrix([6, 160, 10]))
```

```
ipdb> prediction[0]

'green'
```

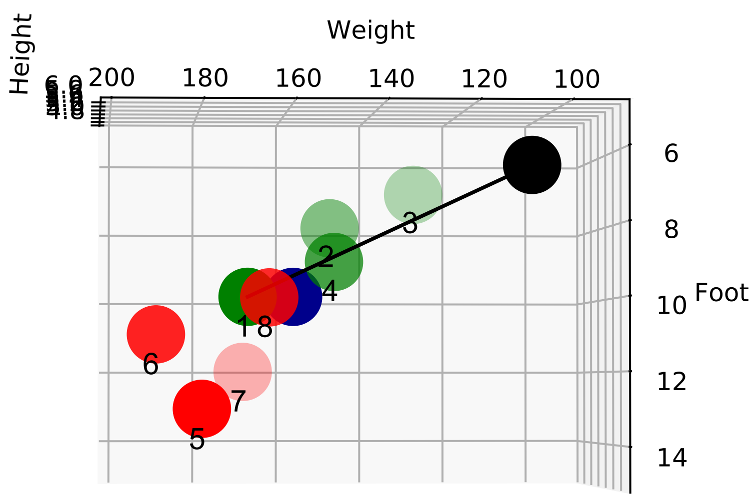
F/W/H Change



id	Height	Weight	Foot	Label
1	$5 \mapsto 6$	$100 \mapsto 170$	$6 \mapsto 10$	green

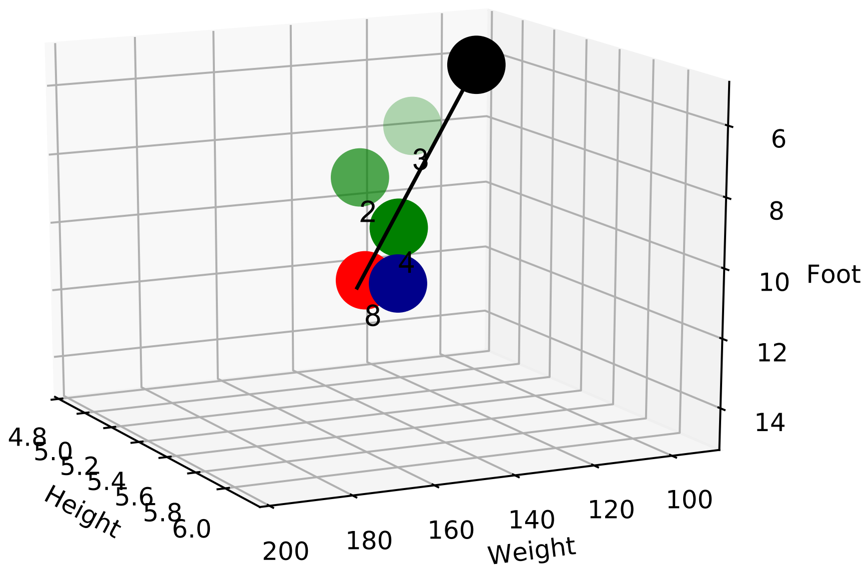
$$(\mathrm{H}=6,\ \mathrm{W}=160,\ \mathrm{F}=10) \mapsto ?$$

Decision Logic for: F/W/H Change



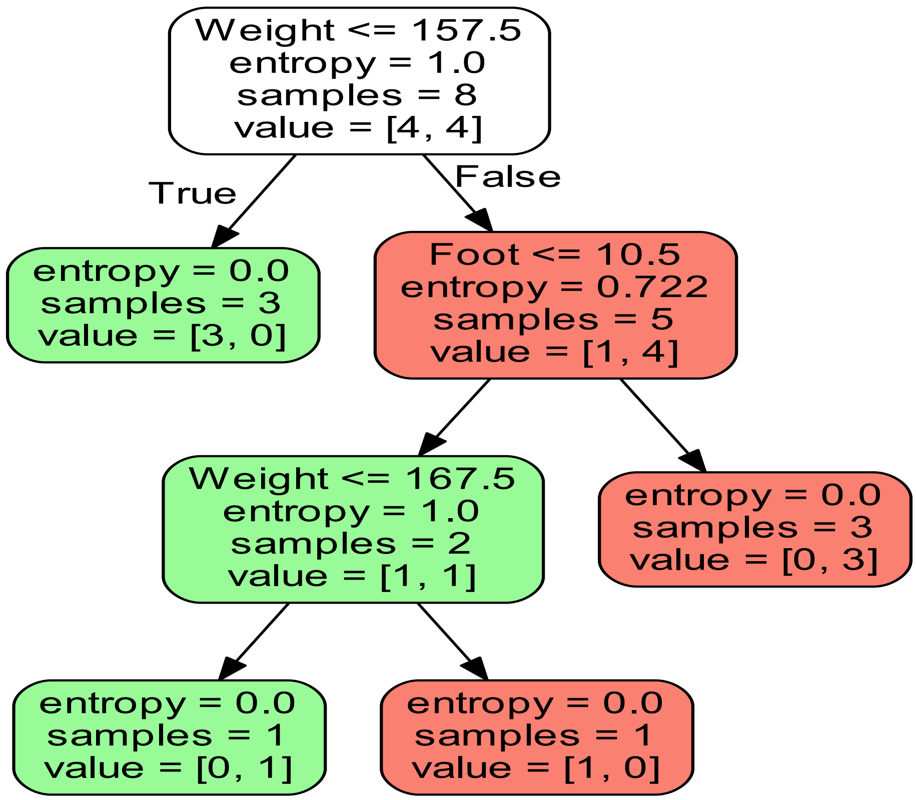
- (H=6, W=160, F=10) $\mapsto$ green
- decide by foot,weight,height

Intermediate Decision



- (H=6, W=160, F=10) $\mapsto$ green
- decide by foot, weight, height

Decision Tree for F/W/H Change



$$(\mathrm{H}{=}6,\mathrm{W}{=}160,\mathrm{F}{=}10) \mapsto \text{green}$$

Code for F/W/H Change

```
import numpy as np
import pandas as pd
from sklearn import tree

data = pd.DataFrame( {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green', 'green', 'green', 'green',
                         'red', 'red', 'red', 'red'],
        'Height': [5, 5.5, 5.33, 5.75,
                            6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                                 180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                              'Foot', 'Label'] )

data['Foot'].iloc[1] = 10    # change foot from 6 to 10!
data['Weight'].iloc[1] = 170 # weight from 100 to 170
data['Height'].iloc[1] = 6   # height from 5 to 6

X = data[['Height', 'Weight', 'Foot']].values
Y = data[['Label']].values
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X,Y)
prediction = clf.predict(np.asmatrix([6, 160, 10]))
```

```
ipdb> prediction[0]
'green'
```

Decision Tree: IRIS

```
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

url = r'https://archive.ics.uci.edu/ml/' + \
          r'machine-learning-databases/iris/iris.data'

iris_feature_names = ['sepal-length', 'sepal-width',
                             'petal-length', 'petal-width']
data = pd.read_csv(url, names=['sepal-length', 'sepal-width',
                         'petal-length', 'petal-width', 'Class'])

class_labels = ['Iris-versicolor', 'Iris-virginica']
data = data[data['Class'].isin(class_labels)]

X = data[iris_feature_names].values
le = LabelEncoder()
Y = le.fit_transform(data['Class'].values)

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,
                                   test_size=0.5,random_state=3)

tree_classifier = tree.DecisionTreeClassifier(criterion = 'entropy')
tree_classifier = tree_classifier.fit(X, Y)

prediction = tree_classifier.predict(X_test)
error_rate = np.mean(prediction != Y_test)
```

```
ipdb> error_rate
0.12
```

Final Remarks

- advantages:

(a) considers all alternatives
(b) fast (once constructed)
(c) easy to use and explain

- disadvantages:

(a) small change in data can lead to a large change
(b) overfitting
(c) computationally intensive if many features are linked

Concepts Check:

(a) root, interior and leaf nodes

(b) entropy and information gain

(c) gini impurity

(d) decision trees for categorical data

(e) advantages and disadvantages of trees