

# Data Science & ML Course

## Lesson #21 Decision Trees

Ivanovitch Silva  
December, 2018



# Update from repository

---

```
git clone https://github.com/ivanovitchm/datascience2machinelearning.git
```

Or ....

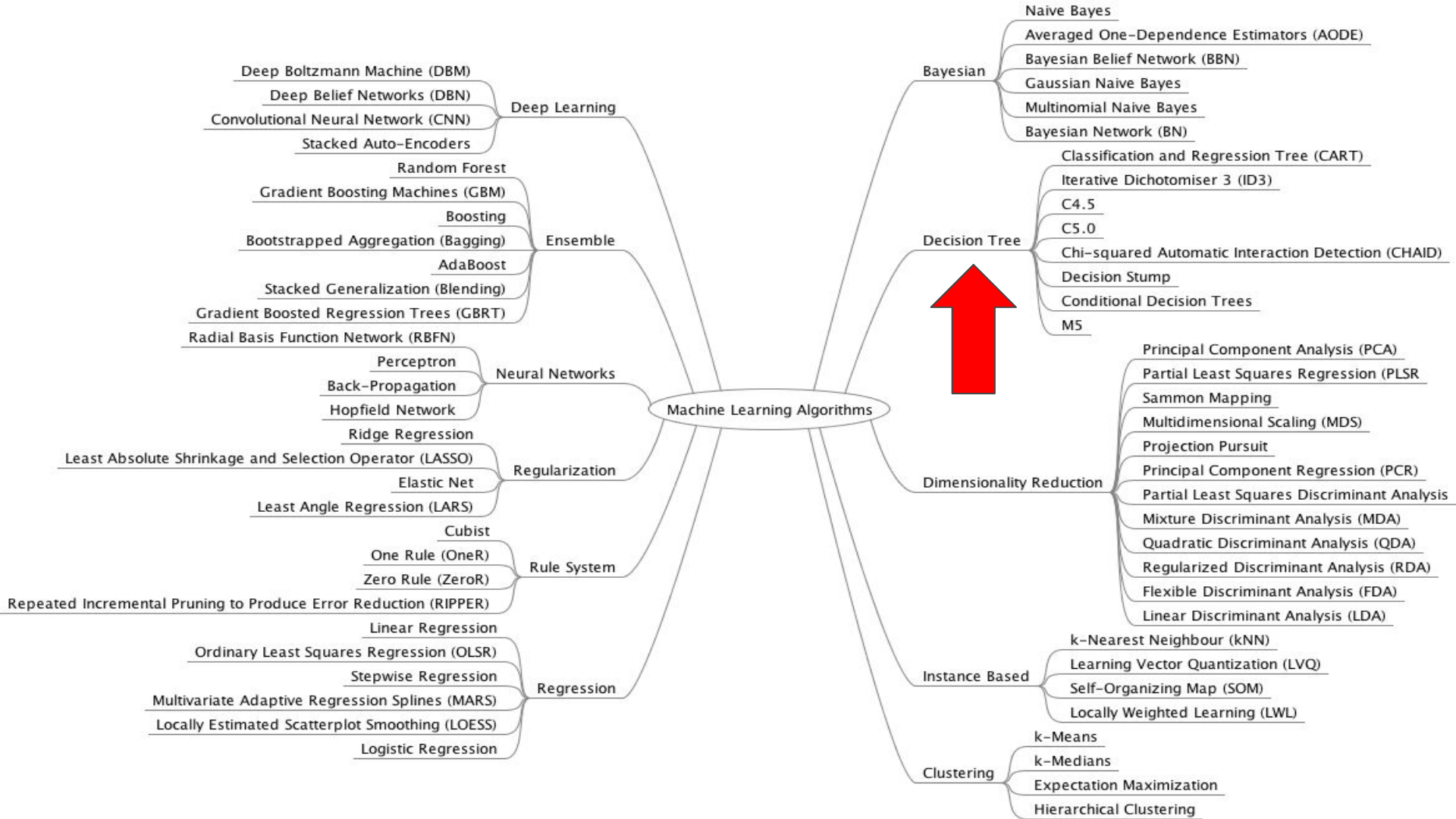
```
git pull
```



# Agenda

---

- Introduction to Decision Tree
- Converting categorical variables
- Splitting Data
- Decision Trees as flows of data
- Entropy
- Information gain
- Applying Decision Trees
- Overfitting problem



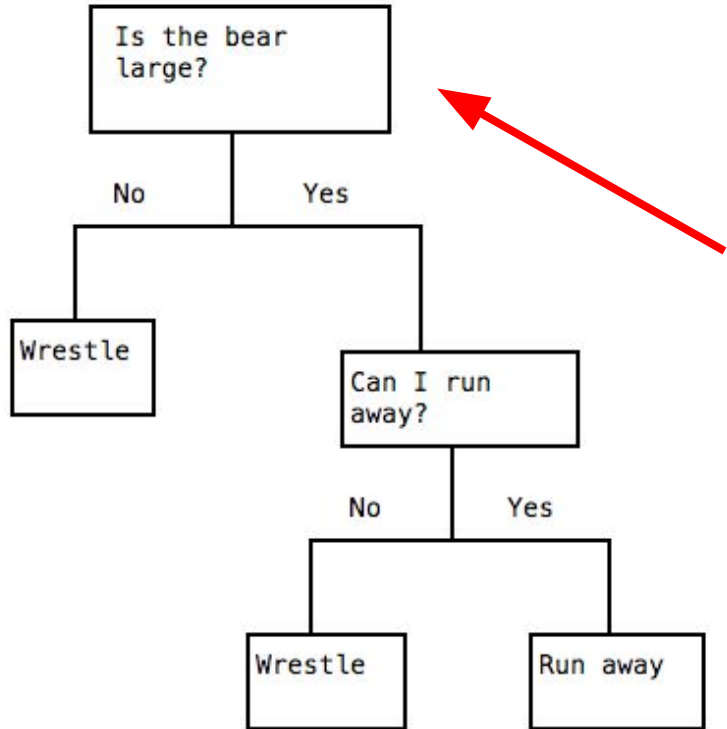


How can an  
algorithm be  
represented as a  
tree?



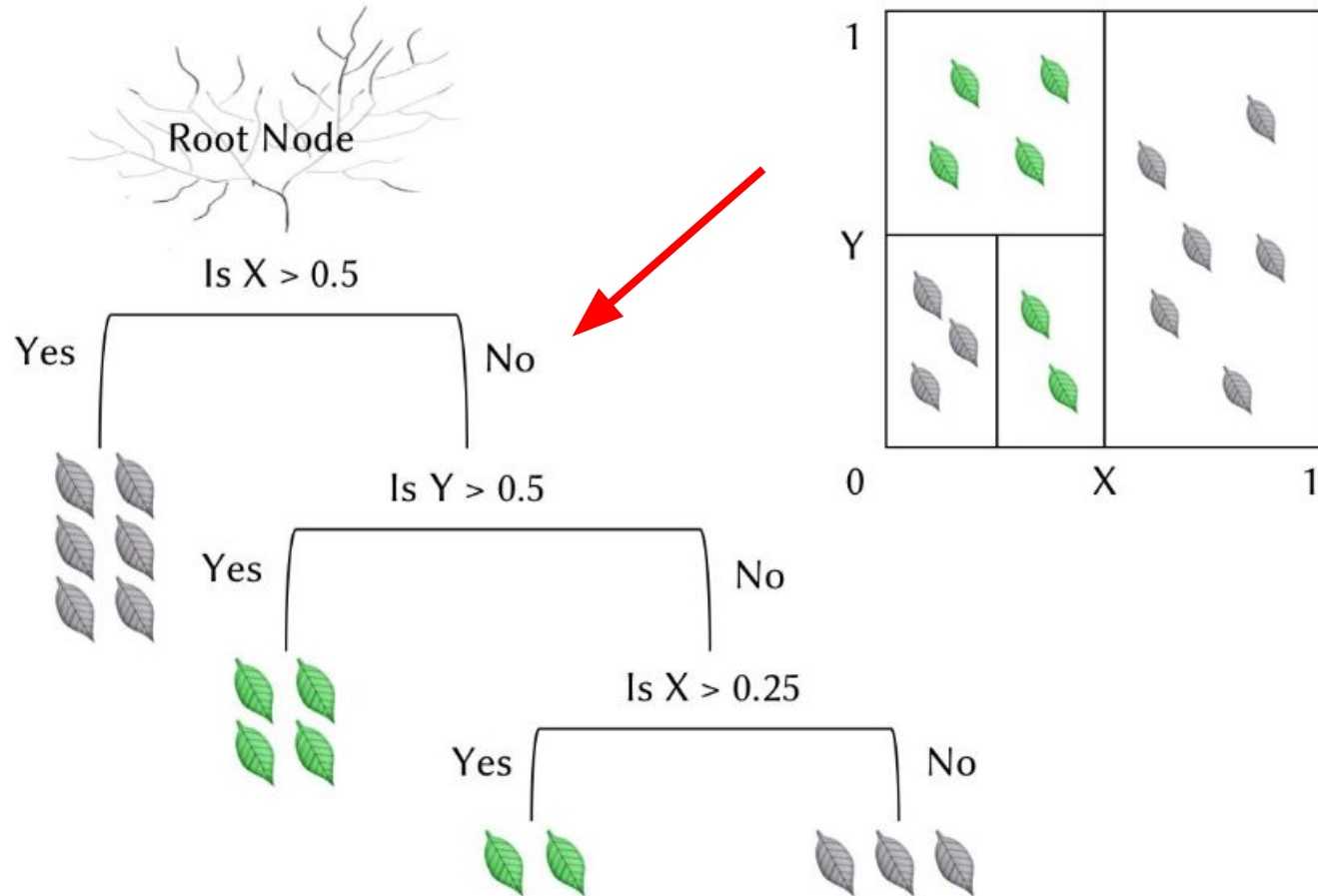
# Decision Tree (classification)

Should I wrestle this bear?



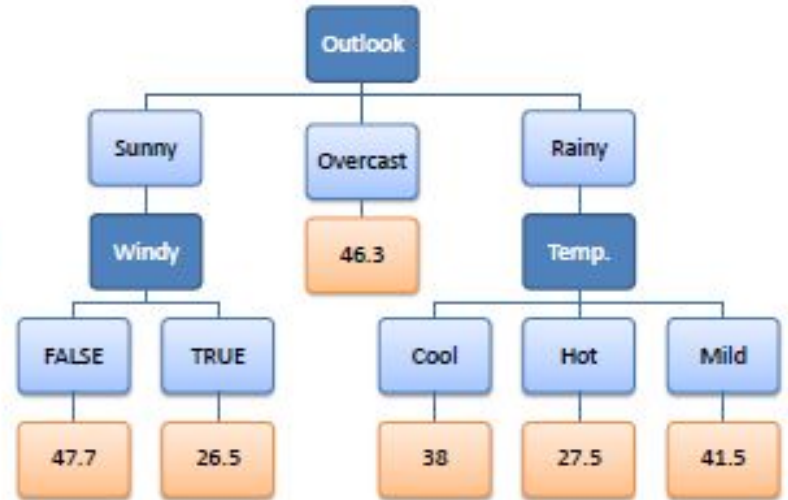
Bear name	Size	Escape possible?	Action
Yogi	Small	No	Wrestle
Winnie	Small	Yes	Wrestle
Baloo	Large	Yes	Run away
Gentle Ben	Large	No	Wrestle

# Decision Tree (classification)



# Decision Tree (regression)

Predictors				Target
Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30





# The Dataset

---

The target column, or what we want to predict, is whether individuals make less than or equal to 50k a year, or more than 50k a year.

US Census  
1994

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0

# Converting Categorical Variables

---

```
1 income.workclass.head()
```

```
0          State-gov
1    Self-emp-not-inc
2          Private
3          Private
4          Private
```

```
1 # Convert a single column from text categories to numbers
2 col = pd.Categorical(income["workclass"])
3 income["workclass"] = col.codes
4 income.workclass.head(5)
```

```
0    7
1    6
2    4
3    4
4    4
```

```
1 col.categories[7].
```

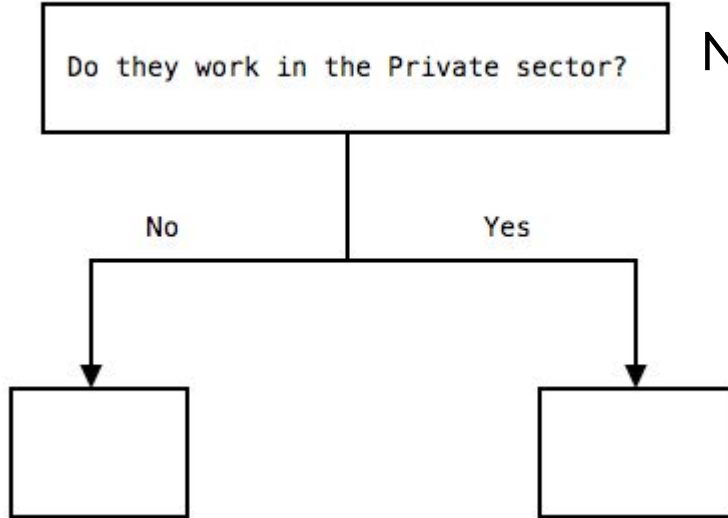
```
' State-gov'
```

```
1 col.categories
```

```
Index(['?', ' Federal-gov', ' Local-gov', ' Never-worked', ' Private',
       ' Self-emp-inc', ' Self-emp-not-inc', ' State-gov', ' Without-pay'],
      dtype='object')
```

# Splitting Data

What income do people make?

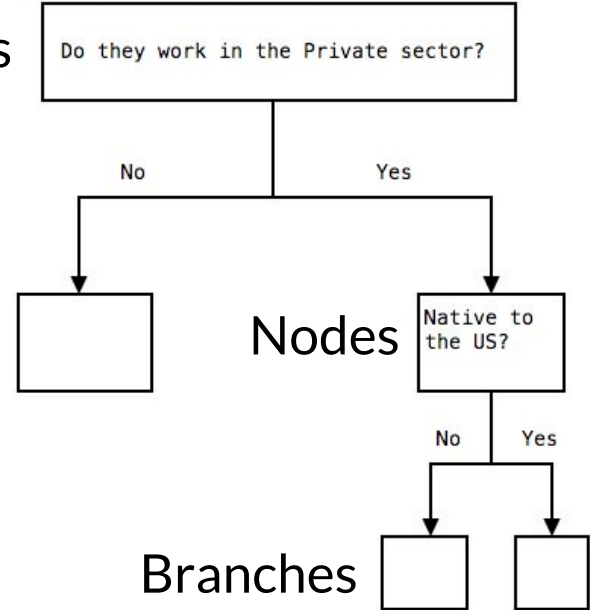


Nodes

Branches

What income do people make?

Nodes

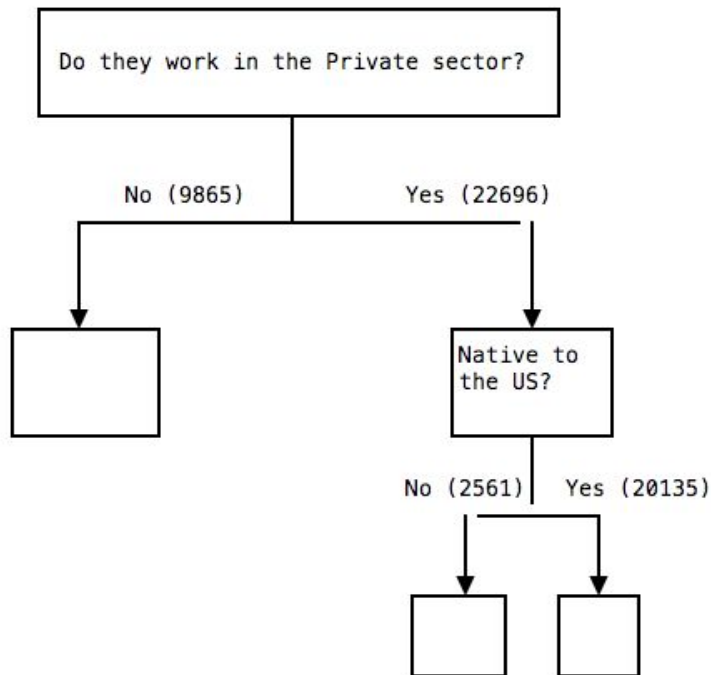


Nodes

Branches

# Decision Tree as Flows of Data

What income do people make?

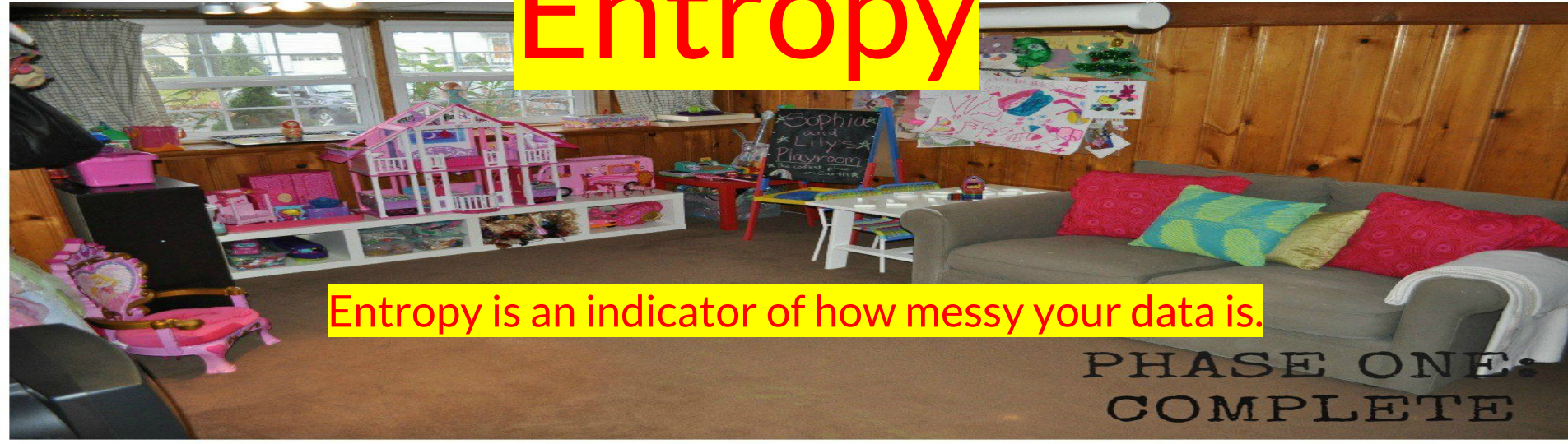


We'll need to continue splitting nodes until we get to a point where all of the rows in a node have the same value for **high\_income**.





# Entropy

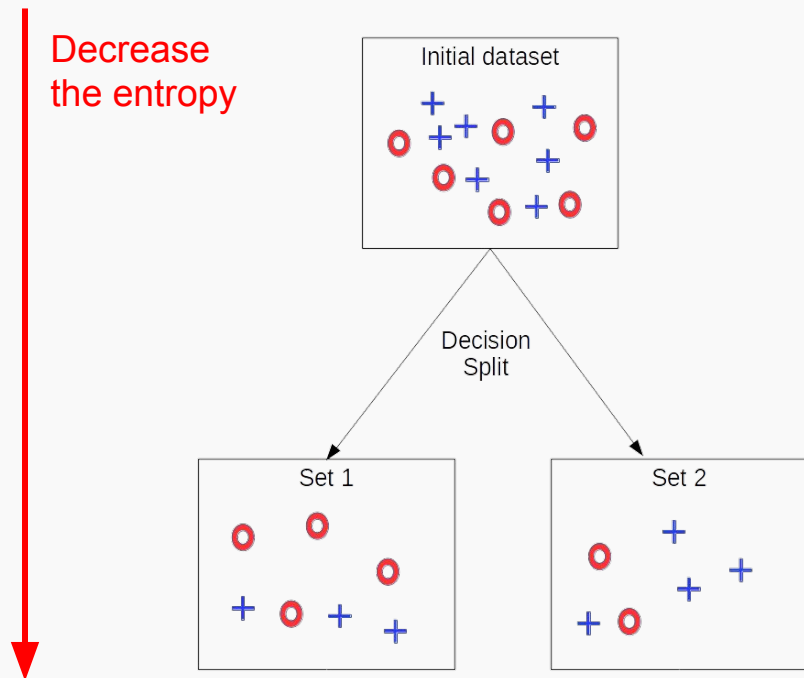


Entropy is an indicator of how messy your data is.

PHASE ONE:  
COMPLETE

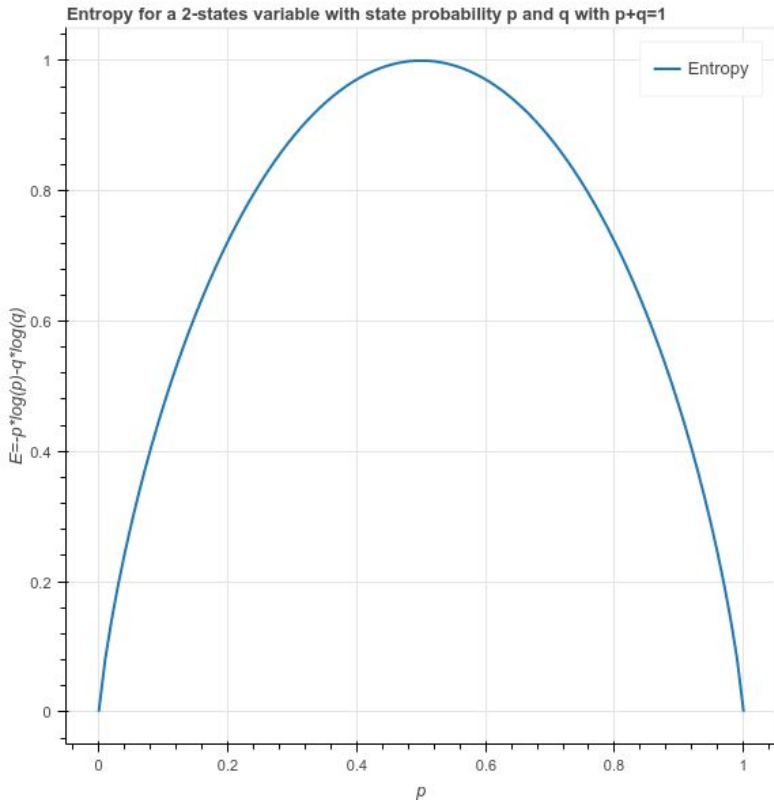


# Why Entropy in Decision Trees?



- The goal is to tidy the data.
- You try to separate your data and group the samples together in the classes they belong to.
- You maximize the purity of the groups as much as possible each time you create a new node of the tree
- Of course, at the end of the tree, you want to have a clear answer.

# Mathematical definition of entropy



- Suppose a set of  $N$  items, these items fall into two categories:
  - Label 1:  $n$  items
  - Label 2:  $m$  items
- $p = n/N$
- $q = m/N$
- $p + q = 1$
- $E = -p \log(p) - q \log(q)$

# Generalization

---

$$E(x) = - \sum_{i=1}^c P(x_i) \log_b P(x_i)$$

`c = pd.x.unique()`

# Entropy using the frequency table of one attribute

---

high\_income

1

1

0

0

1

$$E(x) = - \sum_{i=1}^c P(x_i) \log_b P(x_i)$$

$$E(\text{high\_income}) = - ((2/5 * \log_2 2/5) + (3/5 * \log_2 3/5))$$

$$E(\text{high\_income}) = - (-0.5287712379549449 + -0.44217935649972373)$$

$$E(\text{high\_income}) = 0.97$$

# Entropy using the frequency table of two attributes

---

age	high_income	split_age
25	1	0
50	1	0
30	0	0
50	0	0
80	1	1

$$E(T, X) = \sum_{c \in X} P(c)E(c)$$

$$E(\text{high\_income}, \text{split\_age}) = \frac{4}{5}E(\text{split\_age}, 0) + \frac{1}{5}E(\text{split\_age}, 1)$$

$$E(\text{split\_age}, 0) = -\left(\frac{1}{2} \times \log_2 \frac{1}{2} + \frac{1}{2} \times \log_2 \frac{1}{2}\right)$$

$$E(\text{split\_age}, 1) = -(0 \times \log_2 0 + 1 \times \log_2 1)$$

$$E(\text{high\_income}, \text{split\_age}) = \frac{4}{5}$$

split\_age is based on median of age (suppose equal to 50)



# Information Gain

---

The information gain is based on the **decrease in entropy after a dataset is split** on an attribute.

$$IG(T,X) = E(T) - E(T,X)$$

Constructing a decision tree is all about finding attribute that returns the **highest information gain** (i.e., the most homogeneous branches).

# Applying Decision Tree

---

## `sklearn.tree`.**DecisionTreeClassifier**

```
class sklearn.tree. DecisionTreeClassifier (criterion='gini', splitter='best', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None,  
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False)
```

[\[source\]](#)

## `sklearn.tree`.**DecisionTreeRegressor**

```
class sklearn.tree. DecisionTreeRegressor (criterion='mse', splitter='best', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None,  
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, presort=False) ¶
```

[\[source\]](#)

# Applying Decision Tree

---

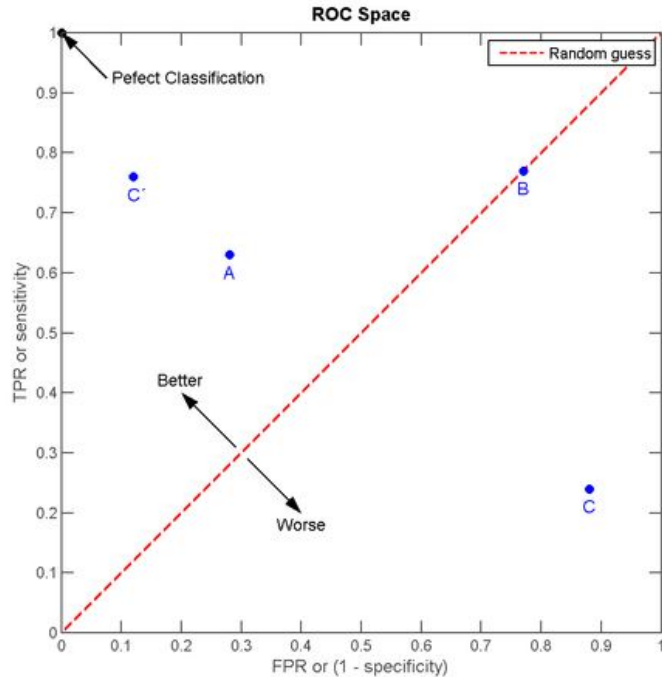
```
from sklearn.tree import DecisionTreeClassifier

# A list of columns to train with
# We've already converted all columns to numeric
columns = ["age", "workclass", "education_num", "marital_status",
           "occupation", "relationship", "race",
           "sex", "hours_per_week", "native_country"]

# Instantiate the classifier
# Set random_state to 1 to make sure the results are consistent
clf = DecisionTreeClassifier(random_state=1)

# fit using features and target
clf.fit(income[columns], income["high_income"])
```

# Receiver Operating Characteristic (ROC)



		True condition	
Total population		Condition positive	Condition negative
Predicted condition	Predicted condition positive	<b>True positive, Power</b>	<b>False positive, Type I error</b>
	Predicted condition negative	<b>False negative, Type II error</b>	<b>True negative</b>
		True positive rate (TPR), Recall, Sensitivity, probability of detection $= \frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\sum \text{False positive}}{\sum \text{Condition negative}}$

AUC - Area Under Curve

# Decision Tree Overfitting

---

```
1 from sklearn.metrics import roc_auc_score
2
3 clf = DecisionTreeClassifier(random_state=1)
4 clf.fit(train[columns], train["high_income"])
5
6 predictions = clf.predict(test[columns])
7 error = roc_auc_score(test["high_income"], predictions)
8 print(error)
```

0.6934656324746192

Splitting the data into training and testing sets doesn't prevent overfitting -- it just helps us detect and fix it.

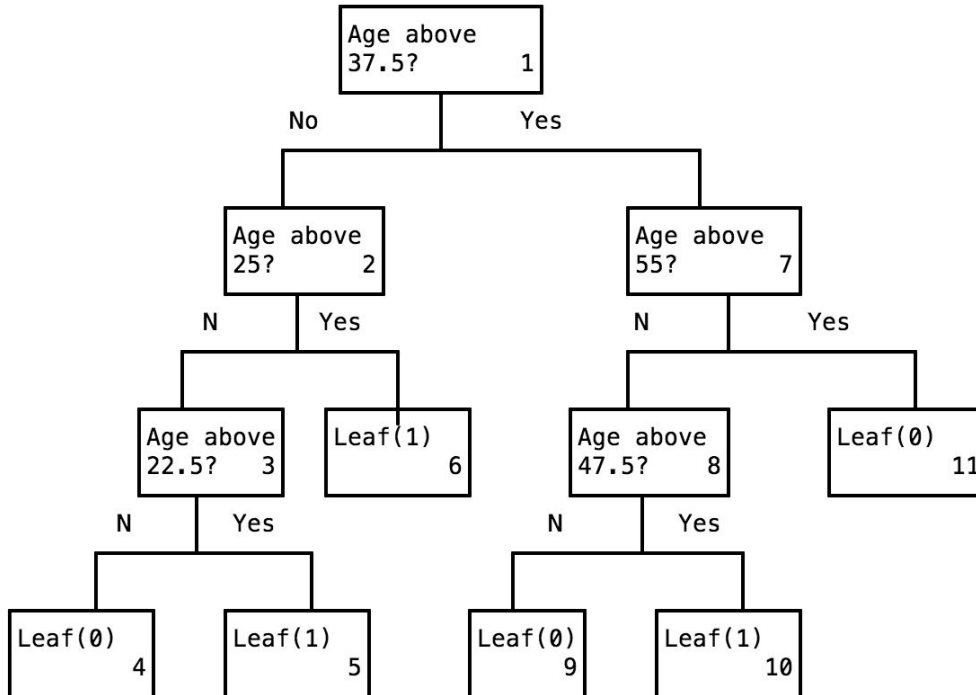
```
1 predictions = clf.predict(train[columns])
2 print(roc_auc_score(train["high_income"], predictions))
```

0.9471244501437455



# Decision Tree Overfitting

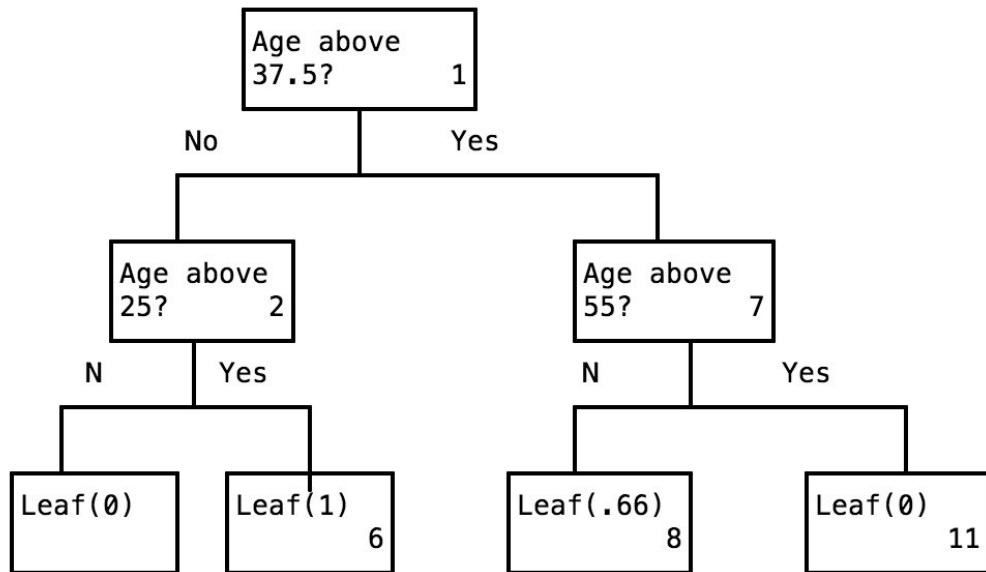
Full tree



- If you're under 22.5 years old, you have a low income
- If you're 22.5 - 37.5, you have a high income
- If you're 37.5 - 47.5, you have a low income
- If you're 47.5 to 55, you have a high income
- Finally, if you're above 55, you have a low income.

# Decision Tree Overfitting

Smaller tree



This version actually has lower accuracy on our training set, but will generalize to new examples better because it matches reality more closely.

# Reducing Overfitting with a Shallower Tree

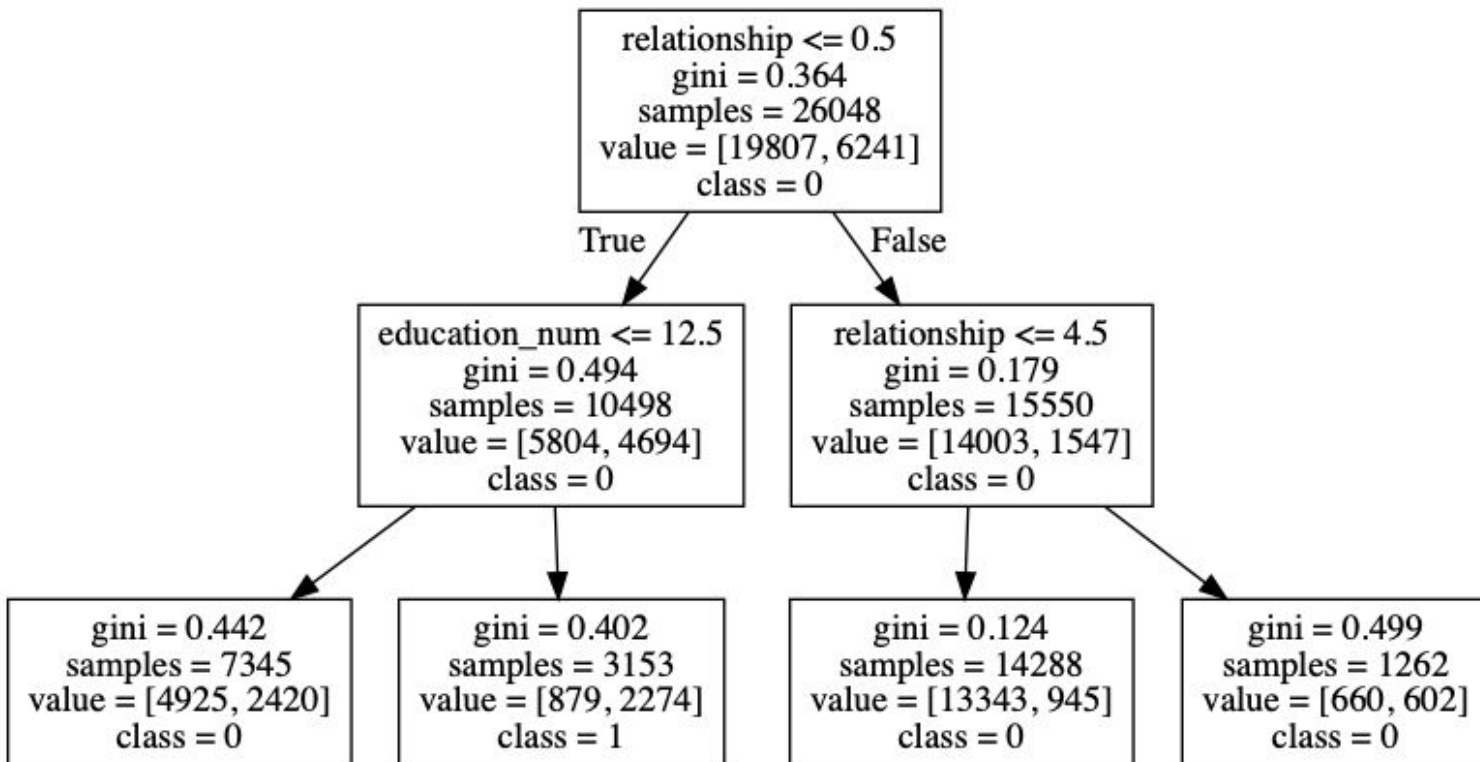
---

- **max\_depth** - Globally restricts how deep the tree can go
- **min\_samples\_split** - The minimum number of rows a node should have before it can be split; if this is set to 2, for example, then nodes with 2 rows won't be split, and will become leaves instead
- **min\_samples\_leaf** - The minimum number of rows a leaf must have
- **min\_weight\_fraction\_leaf** - The fraction of input rows a leaf must have
- **max\_leaf\_nodes** - The maximum number of total leaves; this will cap the count of leaf nodes as the tree is being built

settings	train AUC	test AUC
default	0.947	0.694
min_samples_split: 13	0.842	0.699

settings	train AUC	test AUC
default (min_samples_split: 2, max_depth: None)	0.947	0.694
min_samples_split: 13	0.842	0.699
min_samples_split: 13, max_depth: 7	0.748	0.743
min_samples_split: 100, max_depth: 2	0.662	0.655

# Visualizing a Decision Tree Model





# Knowing when to use decision trees

---

The main advantages of using decision trees is that they're:

- Easy to interpret
- Relatively fast to fit and make predictions
- Able to handle multiple types of data
- Able to pick up nonlinearities in data, and usually fairly accurate

The main disadvantage of using decision trees is their **tendency to overfit**.

# Lesson #21 - Decision Trees.ipynb

