LEDE Algorithms

Richard Dunks

Chase Davis

WEEK 3
CLASS 2

DO IT NOW

3-2_DoNow.ipynb

HOMEWORK

Issues?

Exercises

1. Write code necessary to analyze the relationship between median income and recycling rate in New York City Community Boards (using 2013_NYC_CD_MedianIncome_Recycle.xlsx). Calculate:
   - coefficient of correlation
   - coefficient of determination
2. What is the relationship between these two variables? Write a short Tumblr post outlining the relationship based on your findings

Exercises

3. Based on the outputs from Exercise 1, create a function that takes in a median income and outputs an estimated recycling rate.

Exercises

4. Using the height_weight_gender.csv data from class, filter the data by gender and create models for each gender (male and female). Write a function that takes in a person's height and gender, and outputs a prediction

Exercises

5. Using data from the FiveThirtyEight post http://53eig.ht/1e2aV6U, write code to calculate the correlation of the responses from the poll.
6. Write a short Tumblr post describing the results of your analysis

Goals for today

- Review p-values
- Discuss the difference between regression and classification
- Introduce the idea of feature engineering
- Discuss decision trees in machine learning
- Discuss evaluating decision tree classifiers in machine learning

AN APOLOGY

statsmodels was probably the way to go for linear regression

```
import statsmodels.formula.api as smf
```

```
lm = smf.ols(formula='Mortality~Exposure',data=df).fit()
```

"Y ~ X"

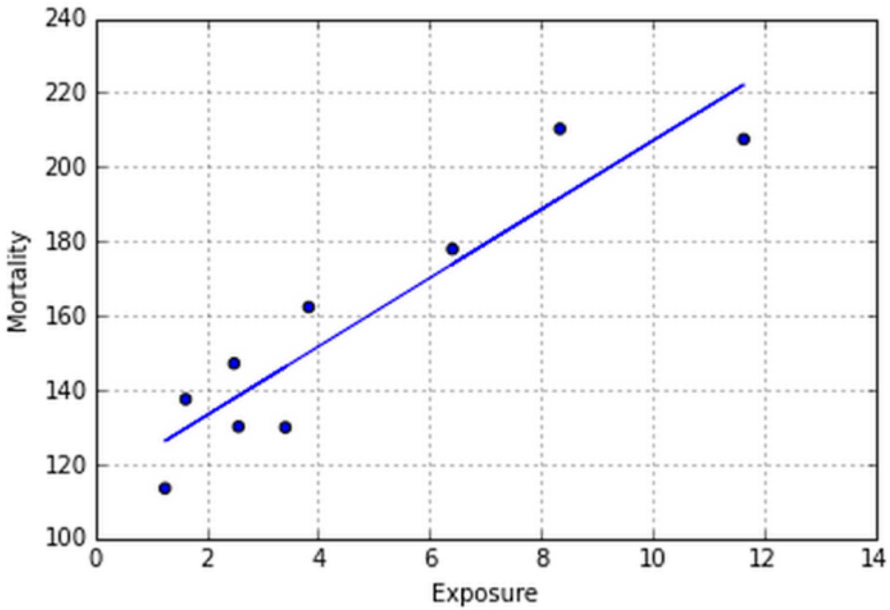
```
lm.params
```

```
Intercept    114.715631
Exposure       9.231456
dtype: float64
```

```
intercept, slope = lm.params
```

```
df.plot(kind='scatter',x='Exposure',y='Mortality')
plt.plot(df['Exposure'],slope*df['Exposure']+intercept,'-')
```

```
[<matplotlib.lines.Line2D at 0x1093571d0>]
```



```
lm.summary()
```

OLS Regression Results

Dep. Variable:	Mortality	**R-squared:**	0.858
Model:	OLS	**Adj. R-squared:**	0.838
Method:	Least Squares	**F-statistic:**	42.34
Date:	Wed, 29 Jul 2015	**Prob (F-statistic):**	0.000332
Time:	21:35:49	**Log-Likelihood:**	-35.397
No. Observations:	9	**AIC:**	74.79
Df Residuals:	7	**BIC:**	75.19
Df Model:	1		

	coef	**std err**	**t**	**P>|t|**	**[95.0% Conf. Int.]**
Intercept	114.7156	8.046	14.258	0.000	95.691 133.741
Exposure	9.2315	1.419	6.507	0.000	5.877 12.586

Omnibus:	2.914	**Durbin-Watson:**	1.542
Prob(Omnibus):	0.233	**Jarque-Bera (JB):**	0.915
Skew:	-0.030	**Prob(JB):**	0.633
Kurtosis:	1.439	**Cond. No.**	9.97

A CONFESSION

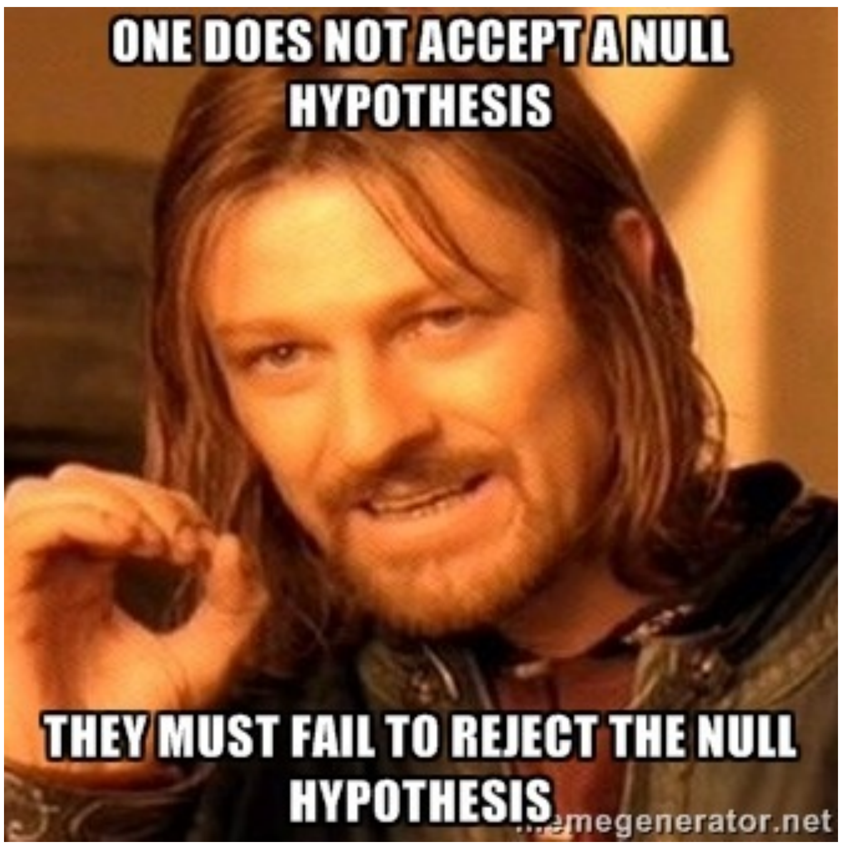
You're all my guinea pigs

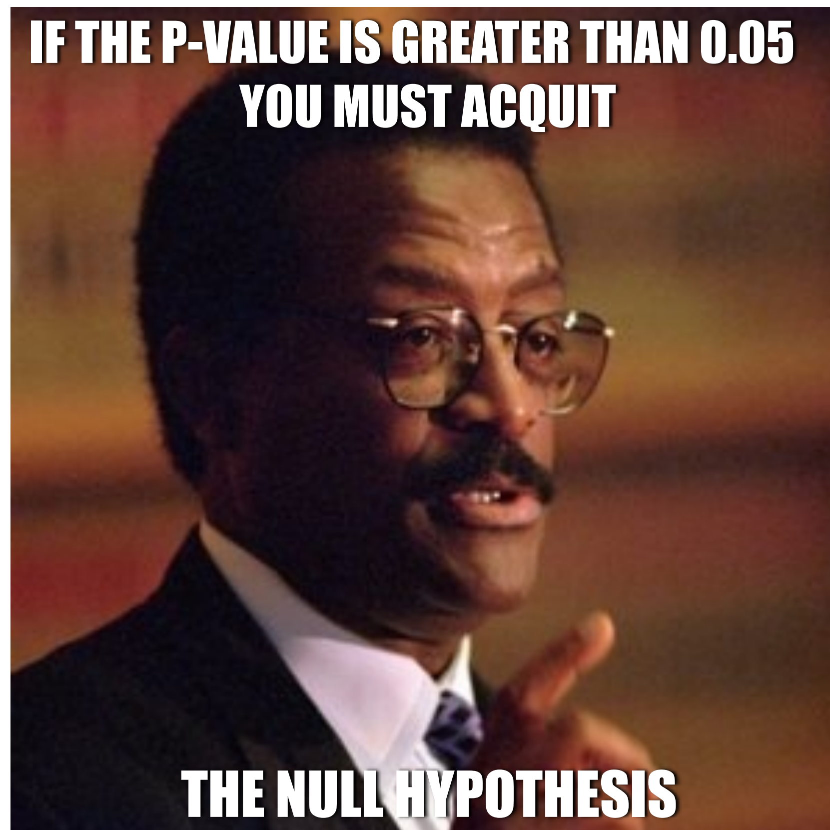
Null Hypothesis

- Start with the belief there is no relationship between variables
- Either reject the null hypothesis or fail to reject the hypothesis
- “Failing to reject” doesn’t mean we accept the null -> we may not have enough data
- The **alternative hypothesis** is that there is a relationship between the variables

p-values

- Help us determine whether we should reject the null hypothesis or fail to reject

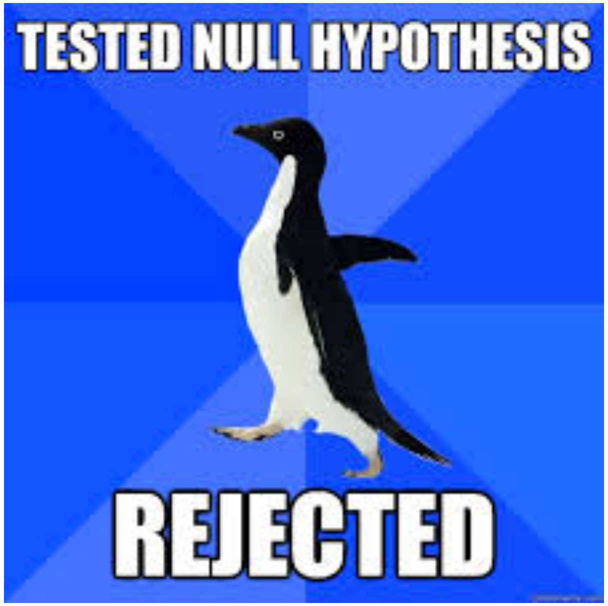




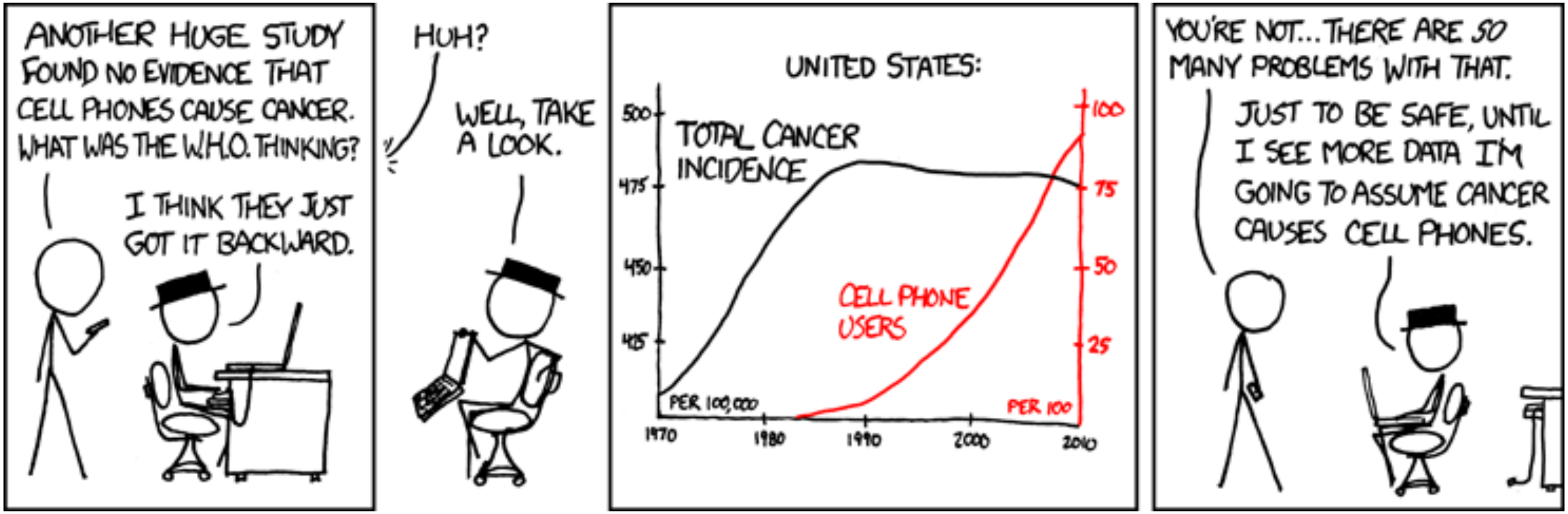
p-values

```
lm.pvalues
```

```
Intercept     0.000002
Exposure      0.000332
```



10 MIN BREAK



http://xkcd.com/925/

Feature Engineering

Actually the success of all Machine Learning algorithms depends on how you present the data.

— Mohammad Pezeshki

http://www.quora.com/What-are-some-general-tips-on-feature-selection-and-engineering-that-every-data-scientist-should-know

Feature Engineering

The process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data

http://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/

Feature Engineering

- Depends on
  - The data you’re using
  - The domain you’re working in
  - The models you’re working with
- Will impact the results
- More an art than a science

Data Types

- Nominal (Categorical)
  - Examples: ID numbers, eye color, zip codes
- Ordinal
  - Examples: rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in {tall, medium, short}
- Interval
  - Examples: calendar dates, temperatures in Celsius or Fahrenheit
- Ratio
  - Examples: temperature in Kelvin, length, time, counts

Discrete Attributes

- Has only a finite or countably infinite set of values
- Examples: zip codes, counts, or the set of words in a collection of documents
- Often represented as integer variables
- Note: binary attributes are a special case of discrete attributes

Continuous Attributes

- Has real numbers as attribute values
- Examples: temperature, height, or weight
- Practically, real values can only be measured and represented using a finite number of digits
- Continuous attributes are typically represented as floating-point variables

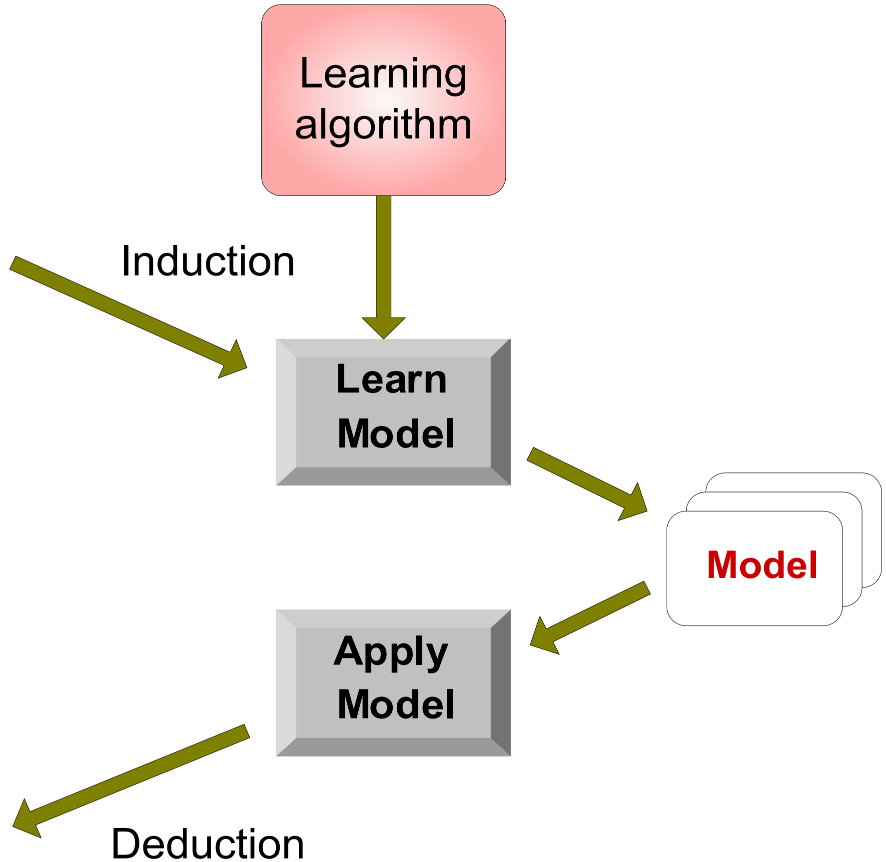
Supervised Learning

- Given a collection of records with a set of attributes and a known target value
- Find a model for the target value as a function of the values of other attributes
- Goal: previously unseen records should be assigned a class as accurately as possible

Supervised Learning

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

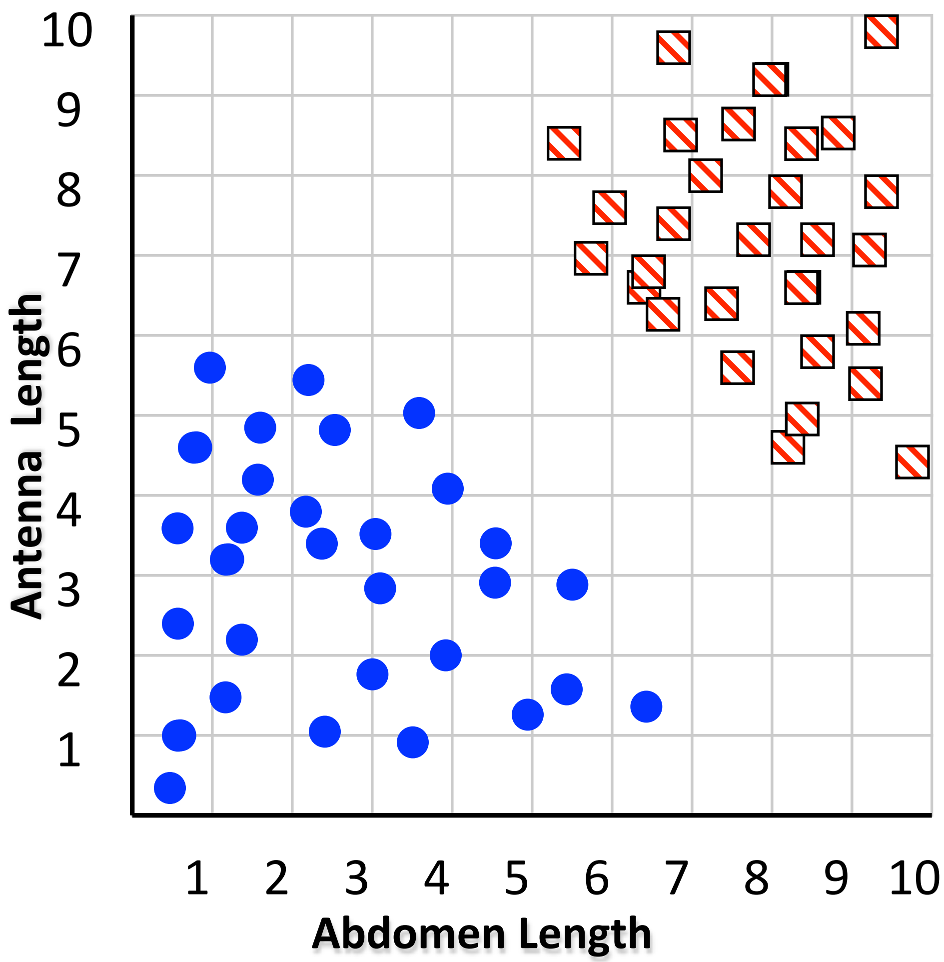
Regression vs Classification

- Regression -> predict continuous ordinal value
- Classification -> predict discrete categorical value

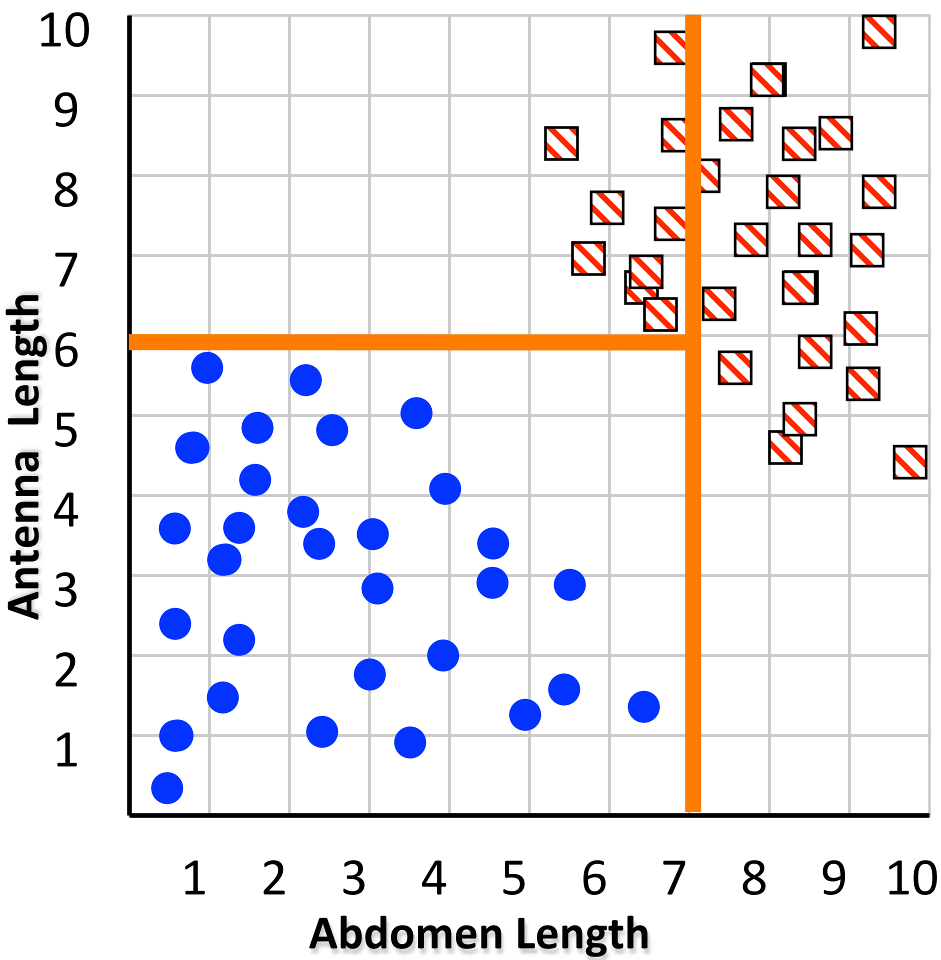
DECISION TREES

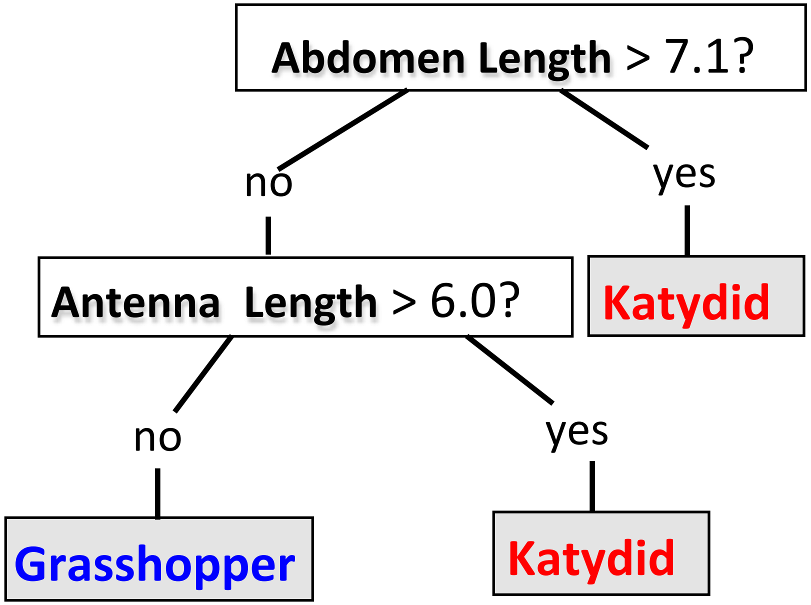
Intuition: Create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features

Decision Tree

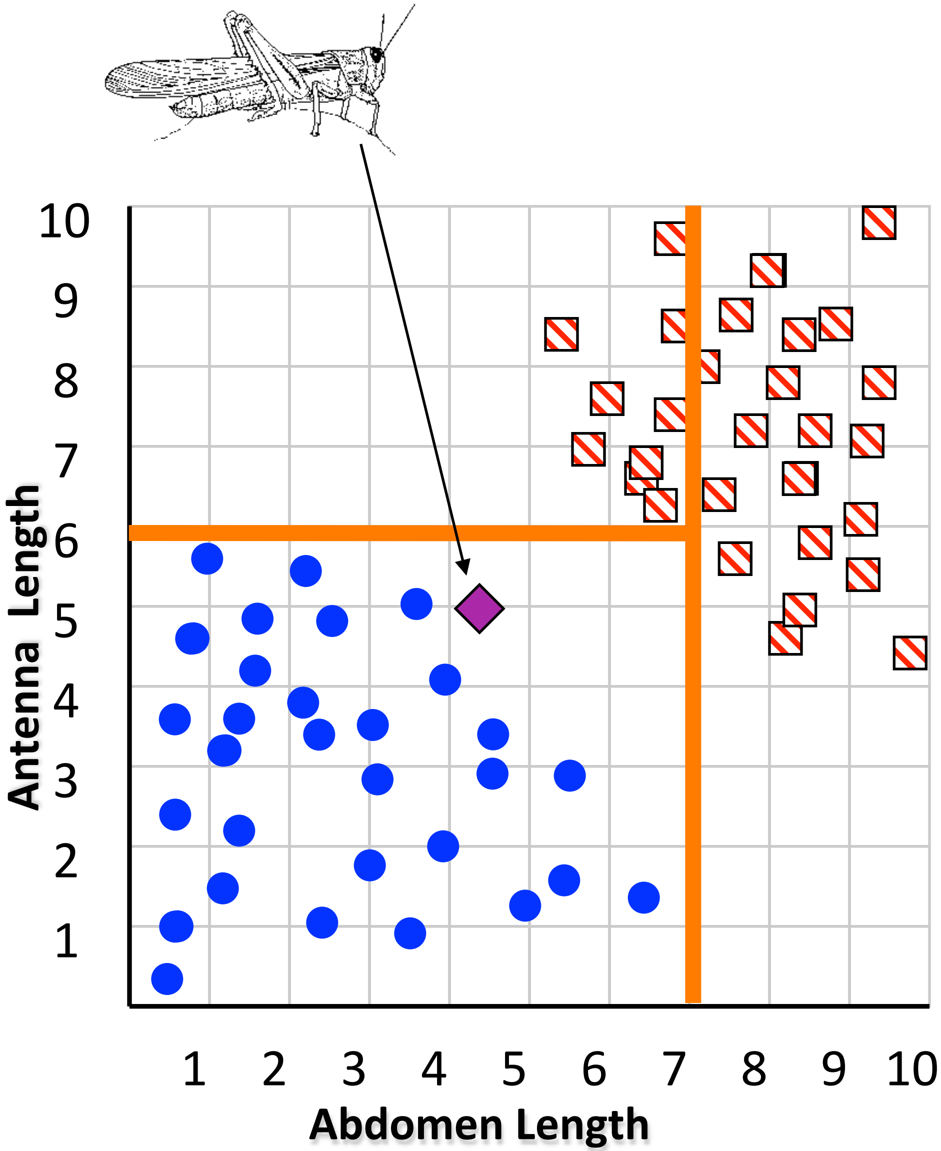


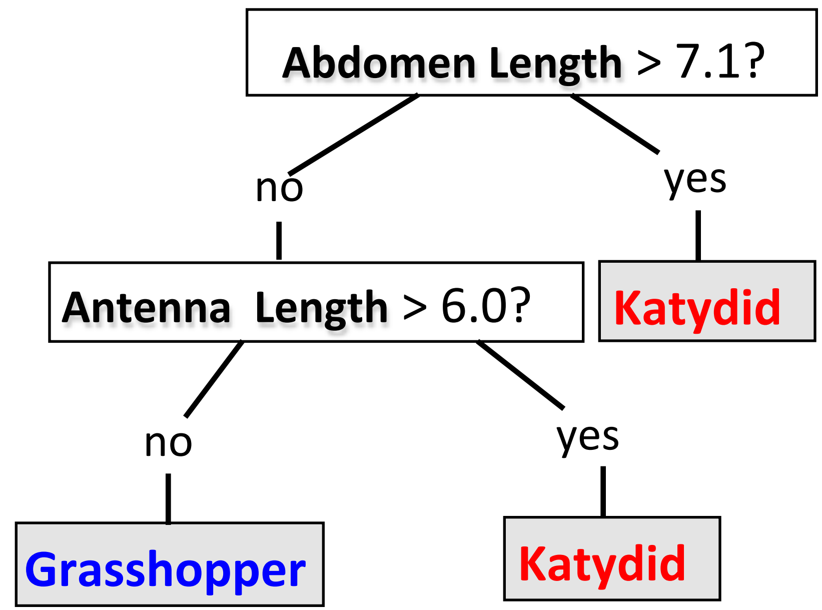
Decision Tree





Decision Tree





Decision Tree

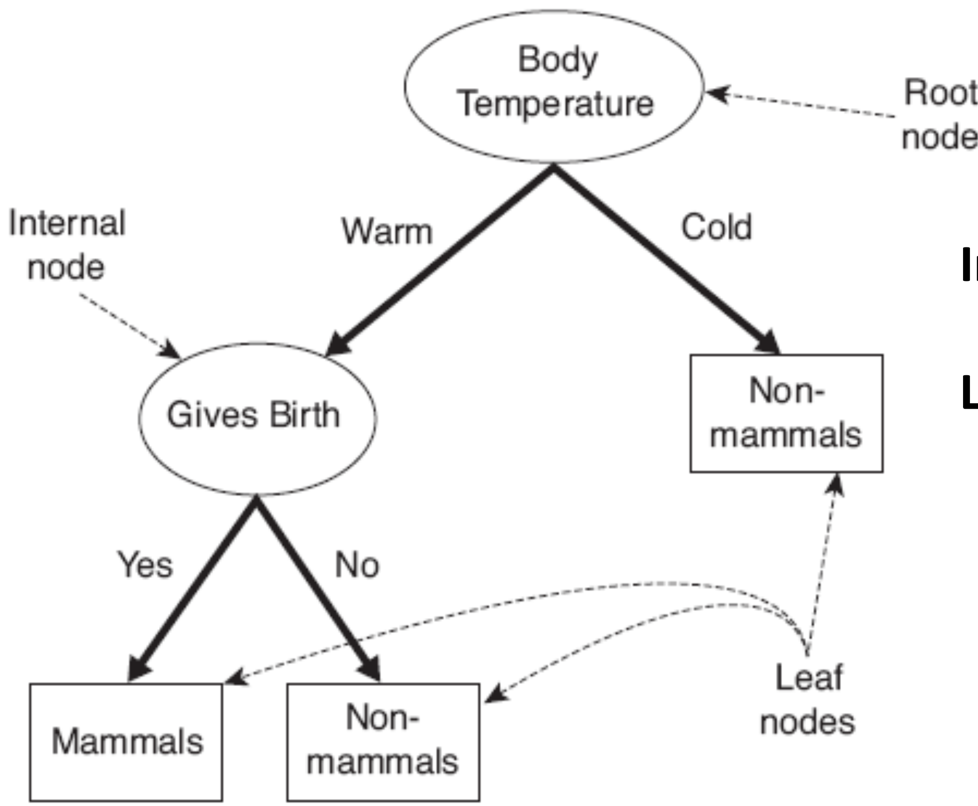


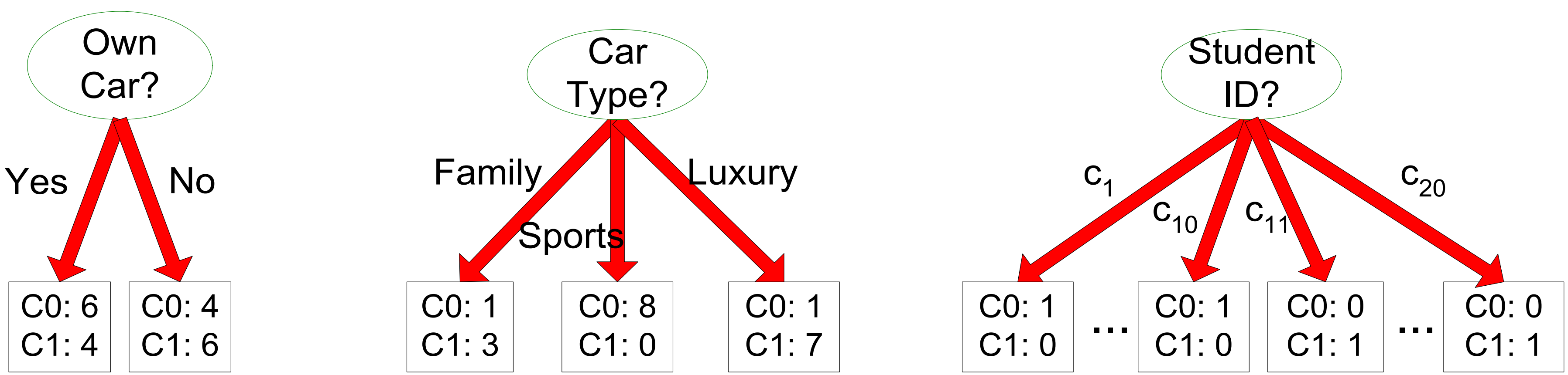
Figure 4.4. A decision tree for the mammal classification problem.

Internal/Decision node: specifies a test on a single attribute

Leaf node: indicates the value of the target attribute

Building a Decision Model

- Greedy strategy -> Split the records based on an attribute test that optimizes certain criterion
- Increase the "purity" of the groups after splitting



Building a Decision Model

C0: 5	C0: 9
C1: 5	C1: 1
Non-homogeneous,	Homogeneous,
High degree of impurity	Low degree of impurity

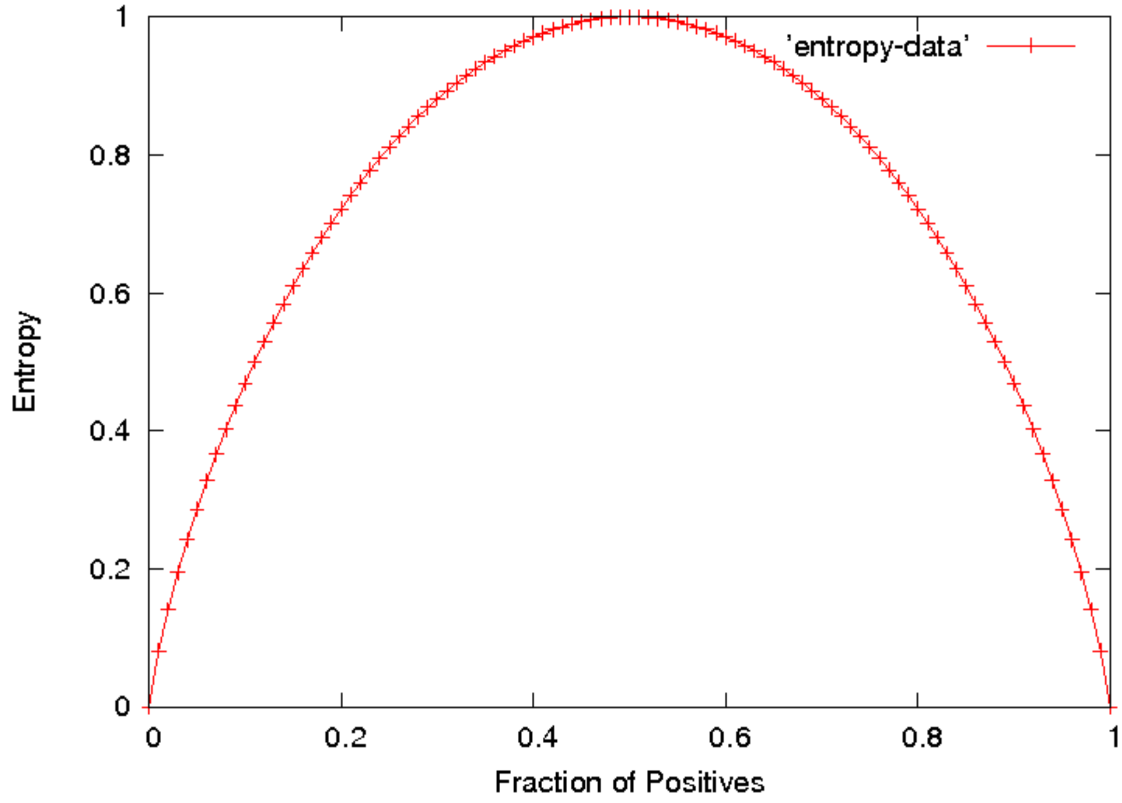
Building a Decision Model

- Select split based on highest information gain (reduction in entropy)

$$E(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

0 log(0) is defined as 0

Building a Decision Model



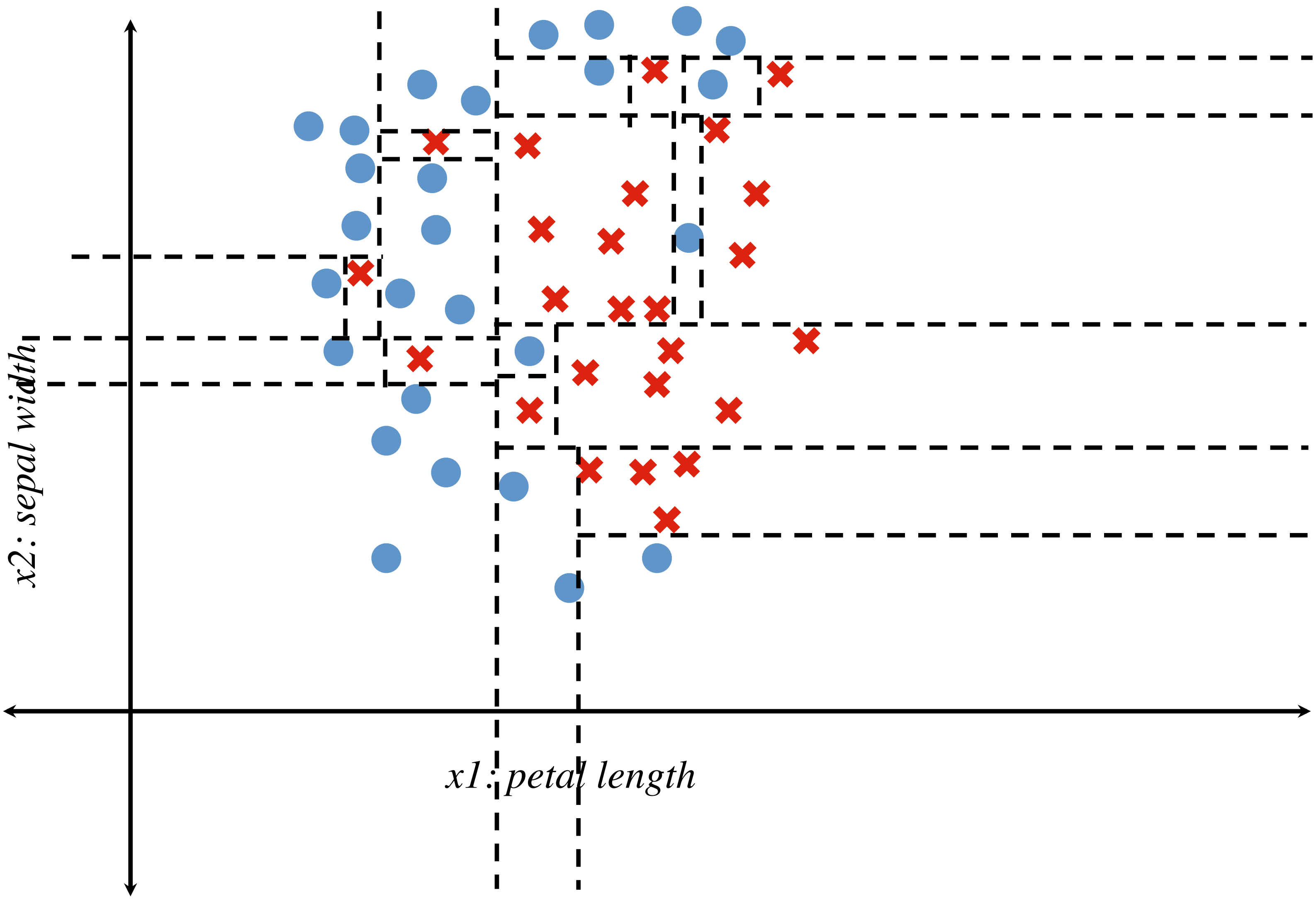
The entropy is 0 if the outcome is ***certain***

The entropy is maximum if we have no knowledge of the system (or any outcome is equally possible)

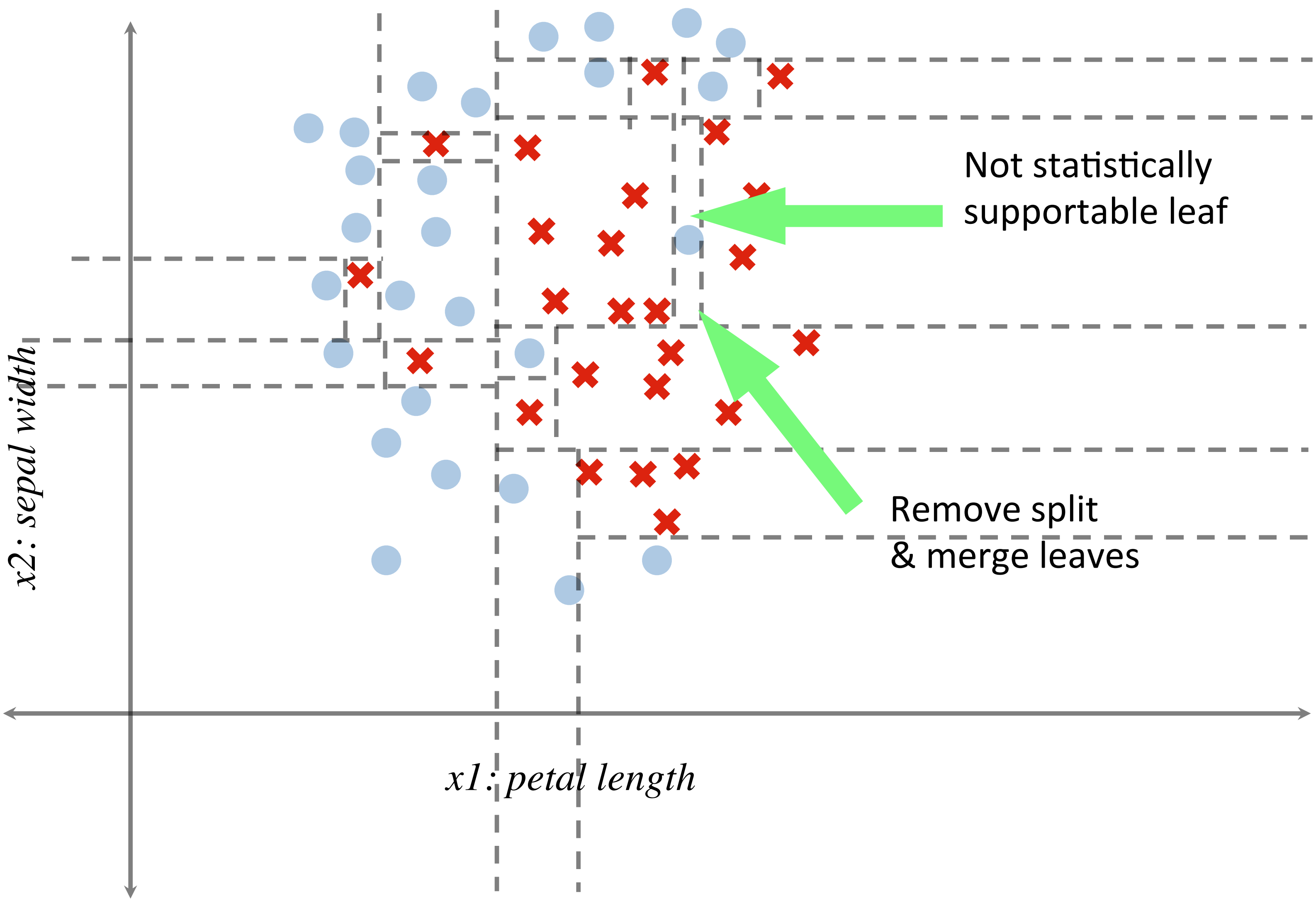
Building a Decision Model

- All training instances start at root
- Training instances are partitioned recursively to build the tree
- Tree pruning is done to remove branches that reflect noise or outliers

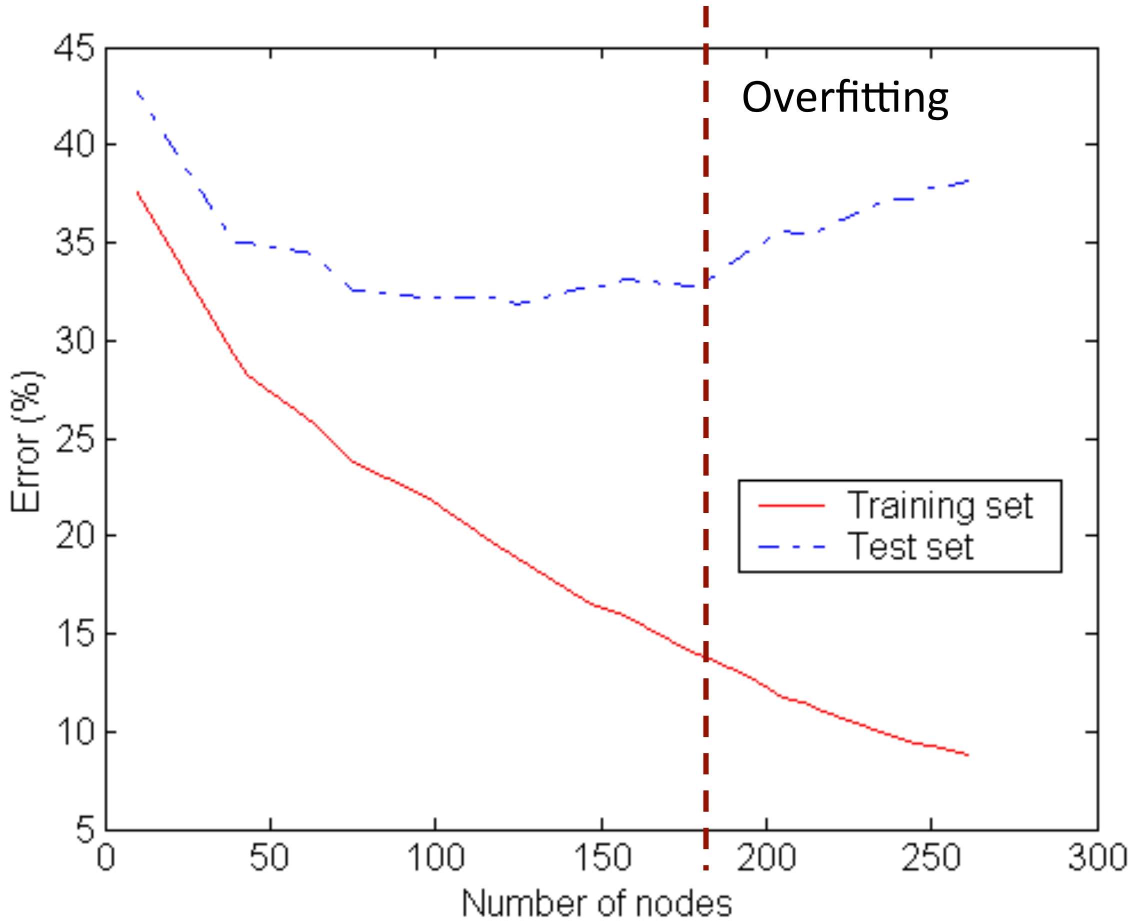
Overfitting Revisited



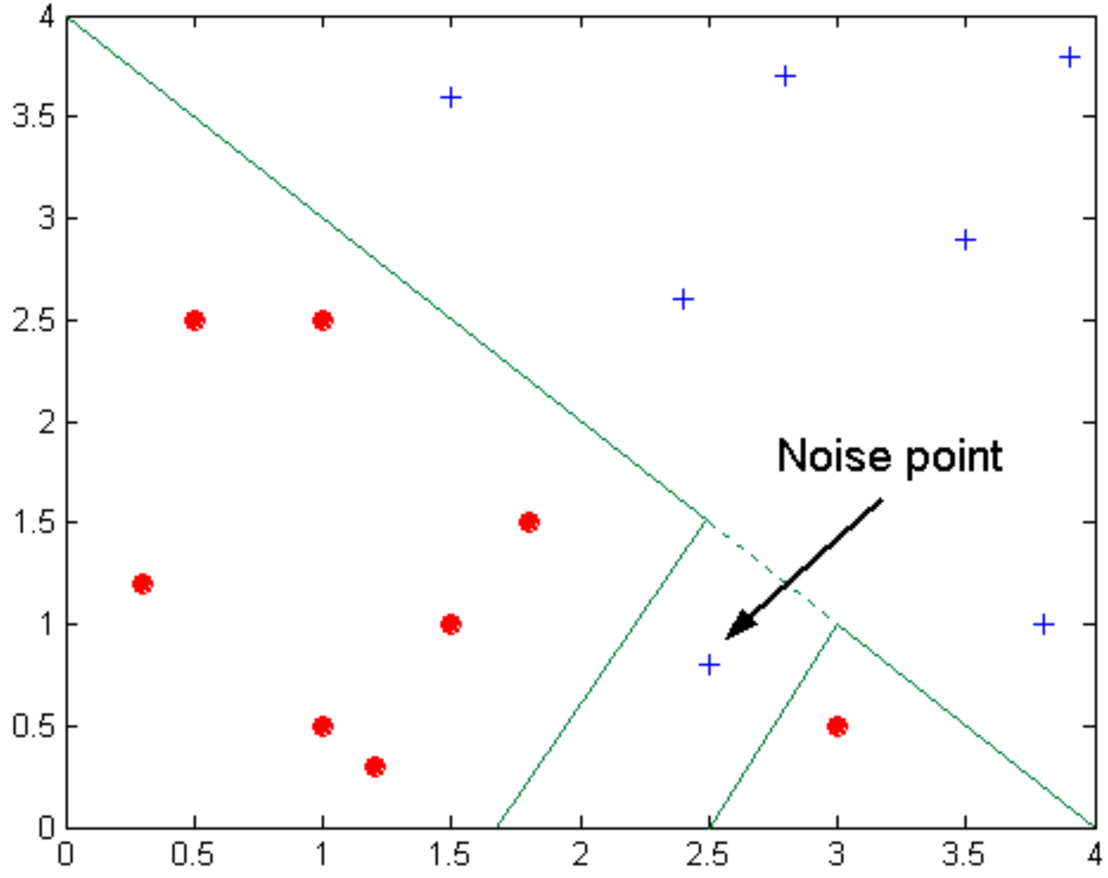
Overfitting Revisited



Overfitting and Underfitting



Overfitting Due to Noise



Dealing with Overfitting

- Pre-pruning
  - Identify overfitting as it happens
  - Limit creation of new nodes
  - Hard to know in advance when you need to prune
- Post-pruning
  - Go back after the fact to trim nodes
  - More used method

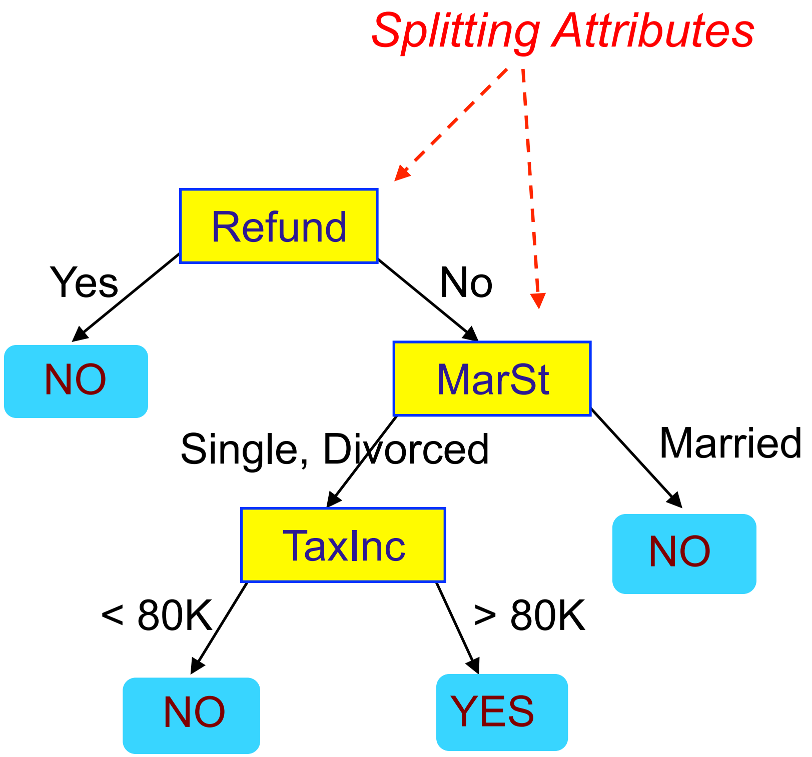
Stop Condition

- All samples for a given node belong to the same class
- There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
- There are no samples left (or only a predefined of samples left

Decision Tree

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
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	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

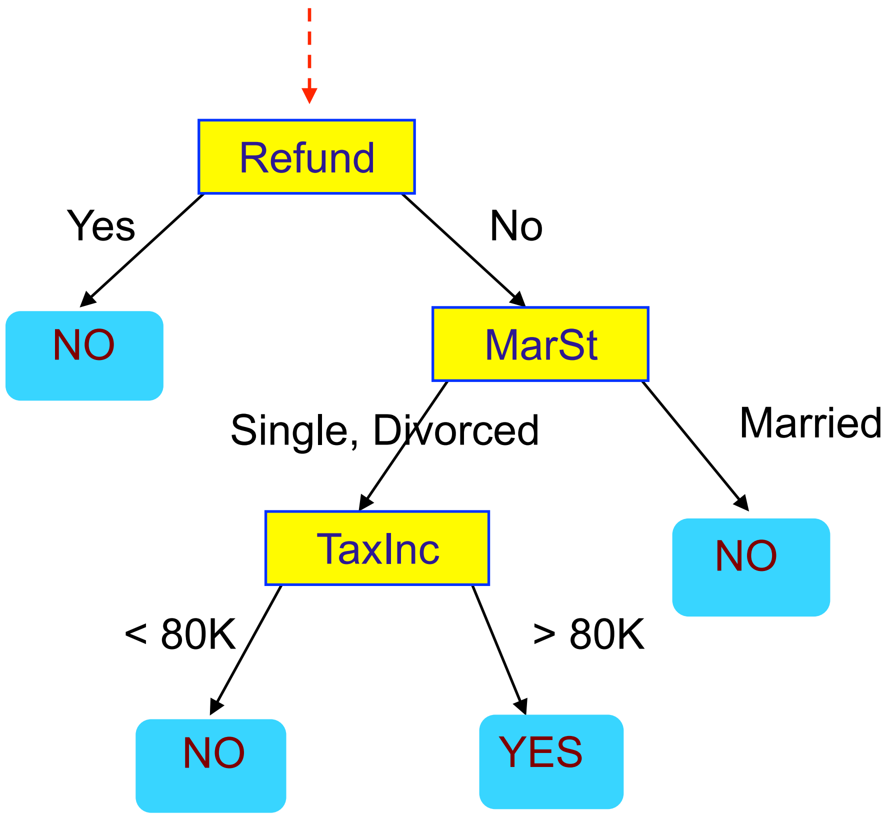
Training Data



Model: Decision Tree

Decision Tree

Start from the root of tree.



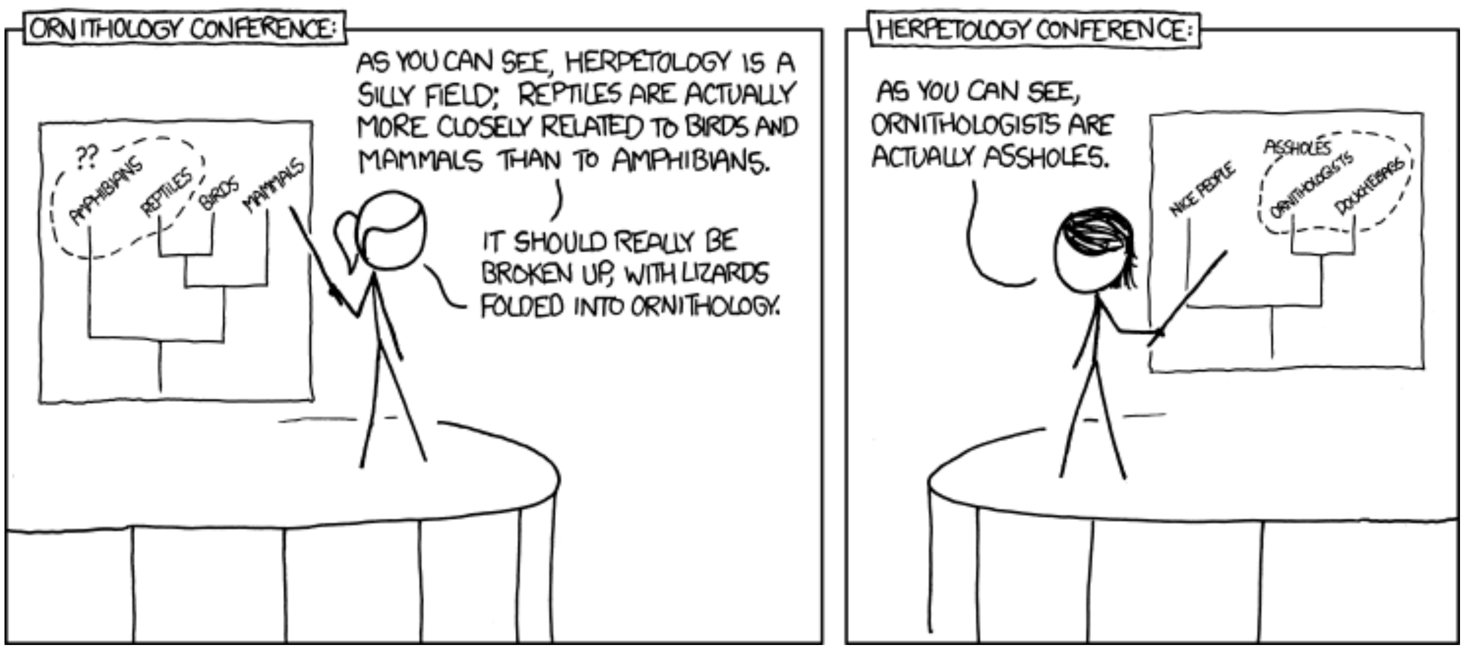
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

AND NOW THIS…

http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

5 MIN BREAK



http://xkcd.com/867/

LETS DO THIS

Decision_Tree.ipynb

EVALUATING CLASSIFIERS

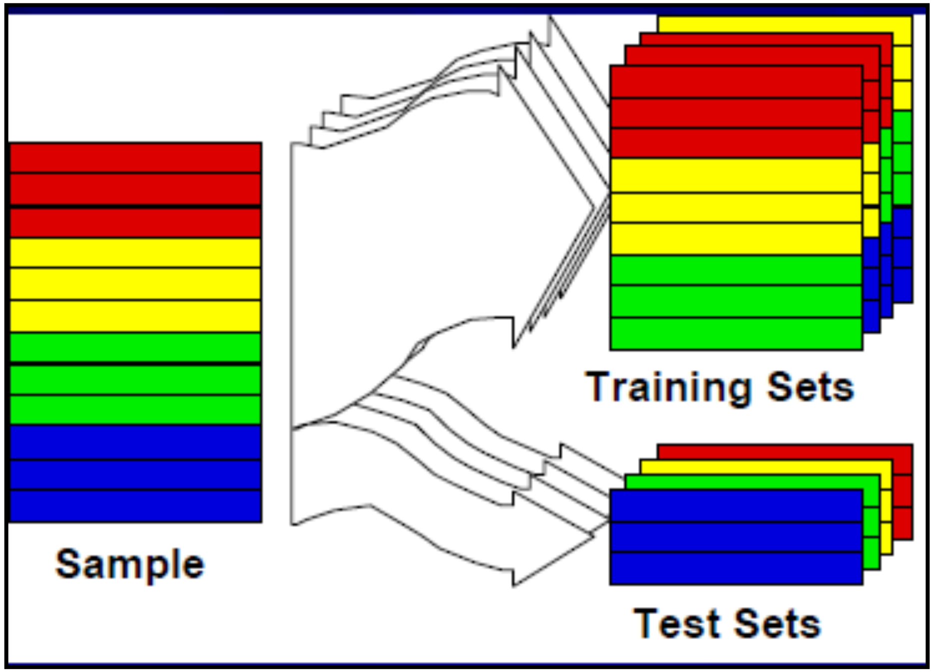
Evaluation

- A **training set** is used to build the model
- A **test set** is used to determine the accuracy of the model on unseen data
- Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it
- A **validation set** is another data set that can additionally be used to test the model

Evaluation Methods

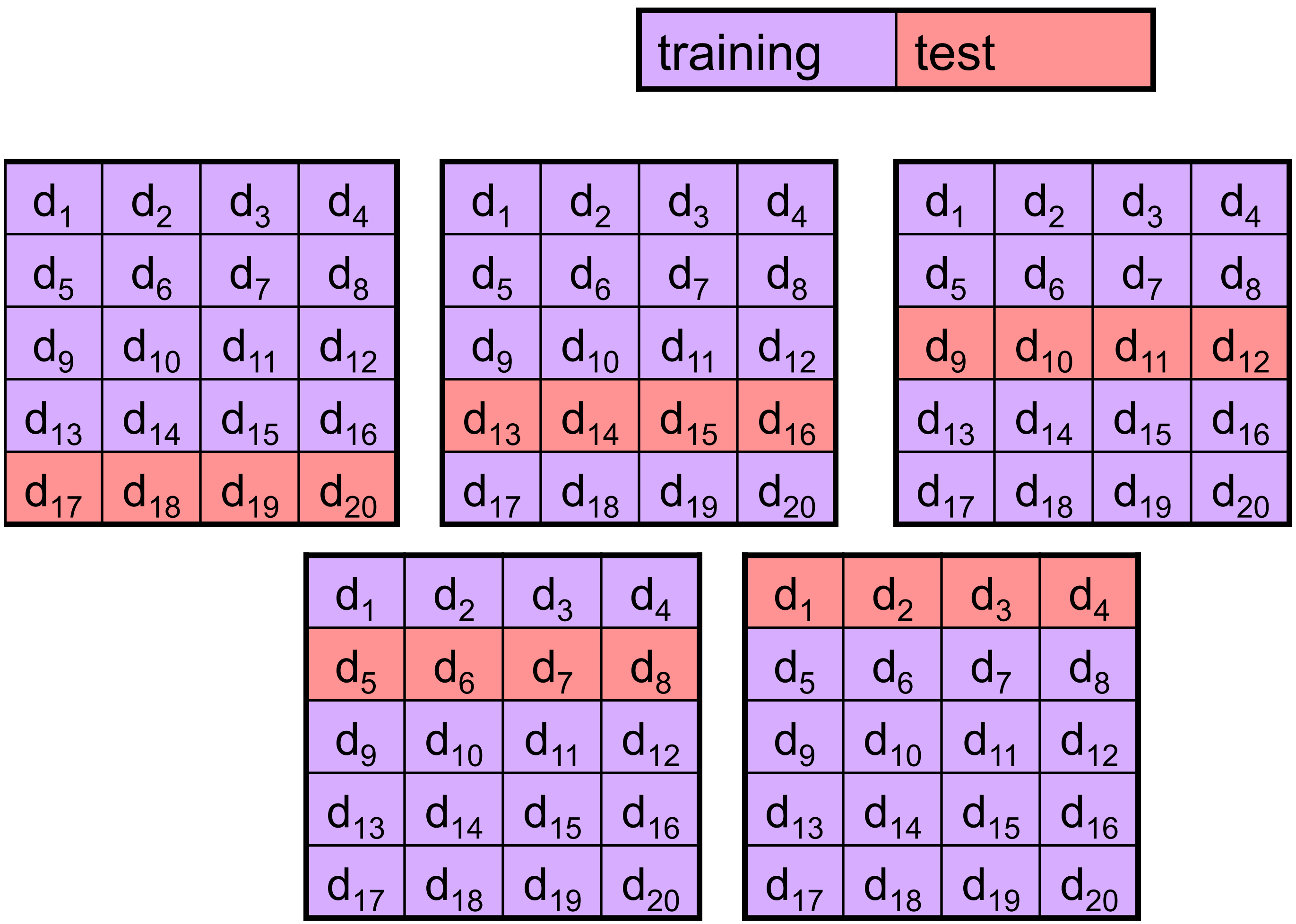
- Holdout: Reserve 2/3 for training and 1/3 for testing
- Random subsampling: Repeated holdout
- Cross validation: Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n

Cross Validation



Cross Validation

Example: data set with 20 instances, 5-fold cross validation



compute error rate for each fold → **then compute average error rate**

Can you average trees?
Solution?

Building a Training and Test Set in Python

```
sklearn.cross_validation.train_test_split(*arrays, **options)
```
[source]

Split arrays or matrices into random train and test subsets

Quick utility that wraps input validation and `next(iter(ShuffleSplit(n_samples)))` and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

Parameters:	***arrays** : sequence of arrays or scipy.sparse matrices with same shape[0] Python lists or tuples occurring in arrays are converted to 1D numpy arrays. **test_size** : float, int, or None (default is None) If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the value is automatically set to the complement of the train size. If train size is also None, test size is set to 0.25. **train_size** : float, int, or None (default is None) If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split. If int, represents the absolute number of train samples. If None, the value is automatically set to the complement of the test size. **random_state** : int or RandomState Pseudo-random number generator state used for random sampling.

Evaluation

		MODEL PREDICTED	
		NO EVENT	*EVENT*
GOLD STANDARD TRUTH	*NO EVENT*	TRUE NEGATIVE	B
	EVENT	C	TRUE POSITIVE

Evaluation

Two types of errors:

False positive (“false alarm”), FP alarm sounds but person is not carrying metal

False negative (“miss”), FN alarm doesn’t sound but person is carrying metal

		MODEL PREDICTED	
		NO EVENT	*EVENT*
GOLD STANDARD TRUTH	*NO EVENT*	A	FALSE POSITIVE (Type 1 Error)
	EVENT	FALSE NEGATIVE (Type 2 Error)	D

Evaluation Metrics

		PREDICTED CLASS	
		Class=P	Class=N
ACTUAL CLASS	Class=P	a (TP)	b (FN)
	Class=N	c (FP)	d (TN)

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Evaluation Metrics

	Positive (+)	Negative (-)
Predicted positive (Y)	***TP***	***FP***
Predicted negative (N)	***FN***	***TN***

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

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

$$\text{True Positive Rate} = \frac{TP}{TP+FN}$$

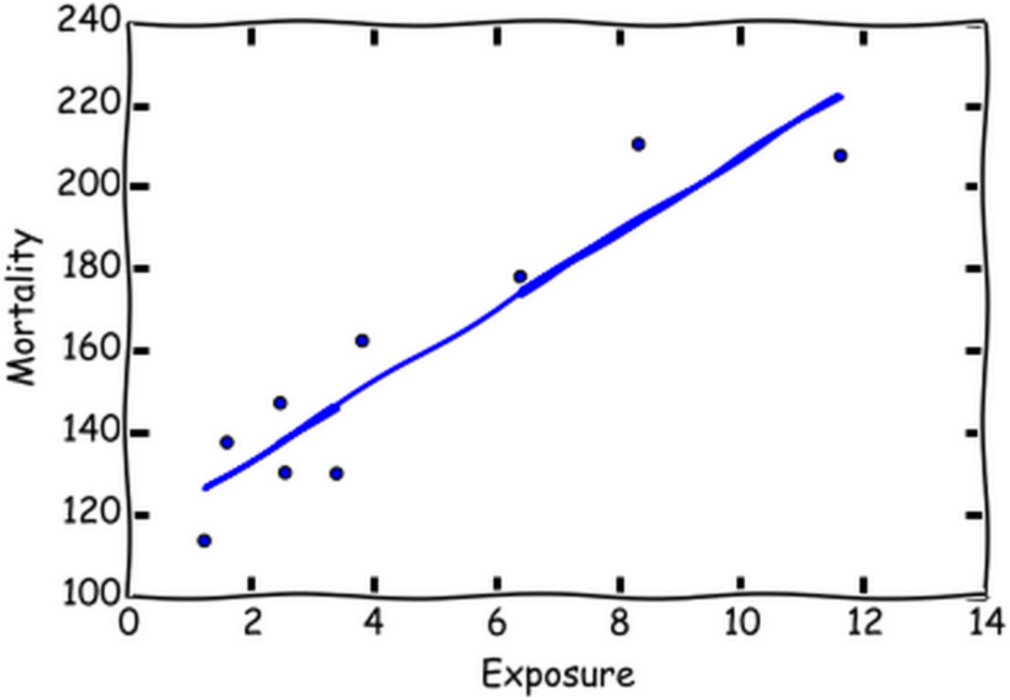
$$\text{False Positive Rate} = \frac{FP}{FP+TN}$$

WRAP-UP

As if you needed more proof Python was awesome…

```
plt.xkcd()
df.plot(kind='scatter',x='Exposure',y='Mortality')
plt.plot(df['Exposure'],slope*df['Exposure']+intercept,'-')
```

```
[<matplotlib.lines.Line2D at 0x109524d50>]
```



http://jakevdp.github.io/blog/2013/07/10/XKCD-plots-in-matplotlib/