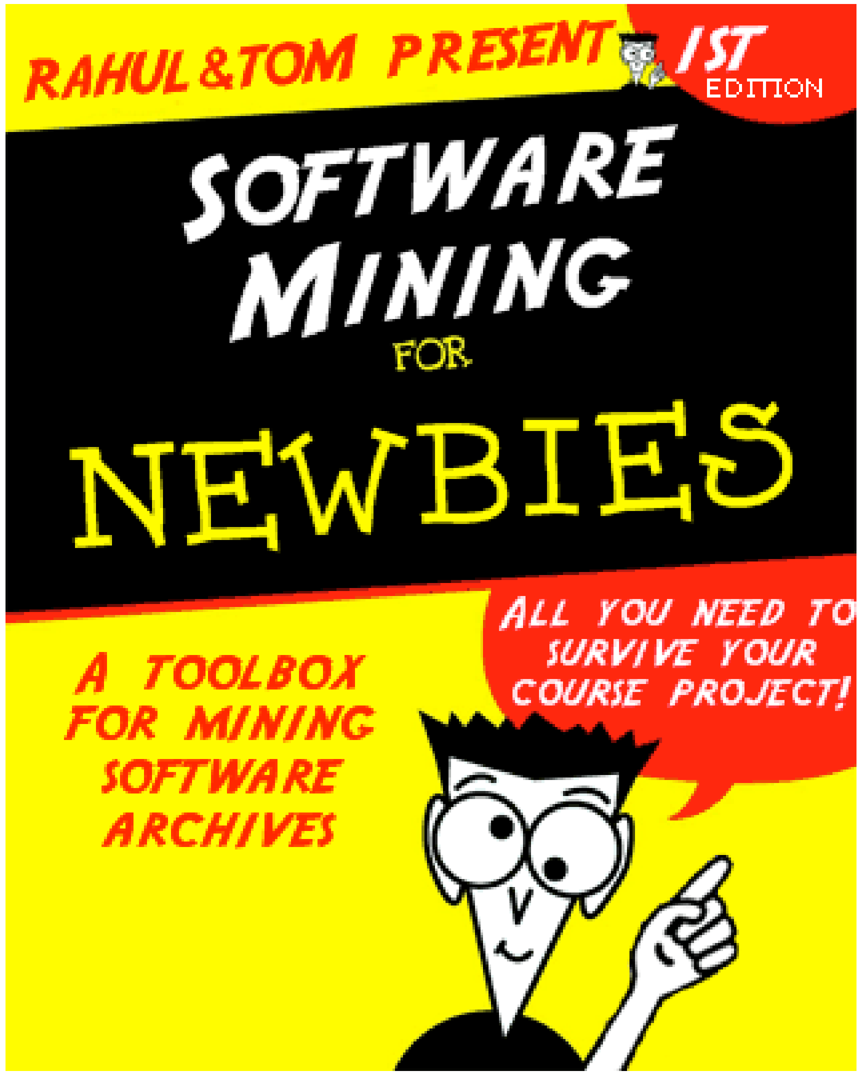


A Toolbox for Software Mining

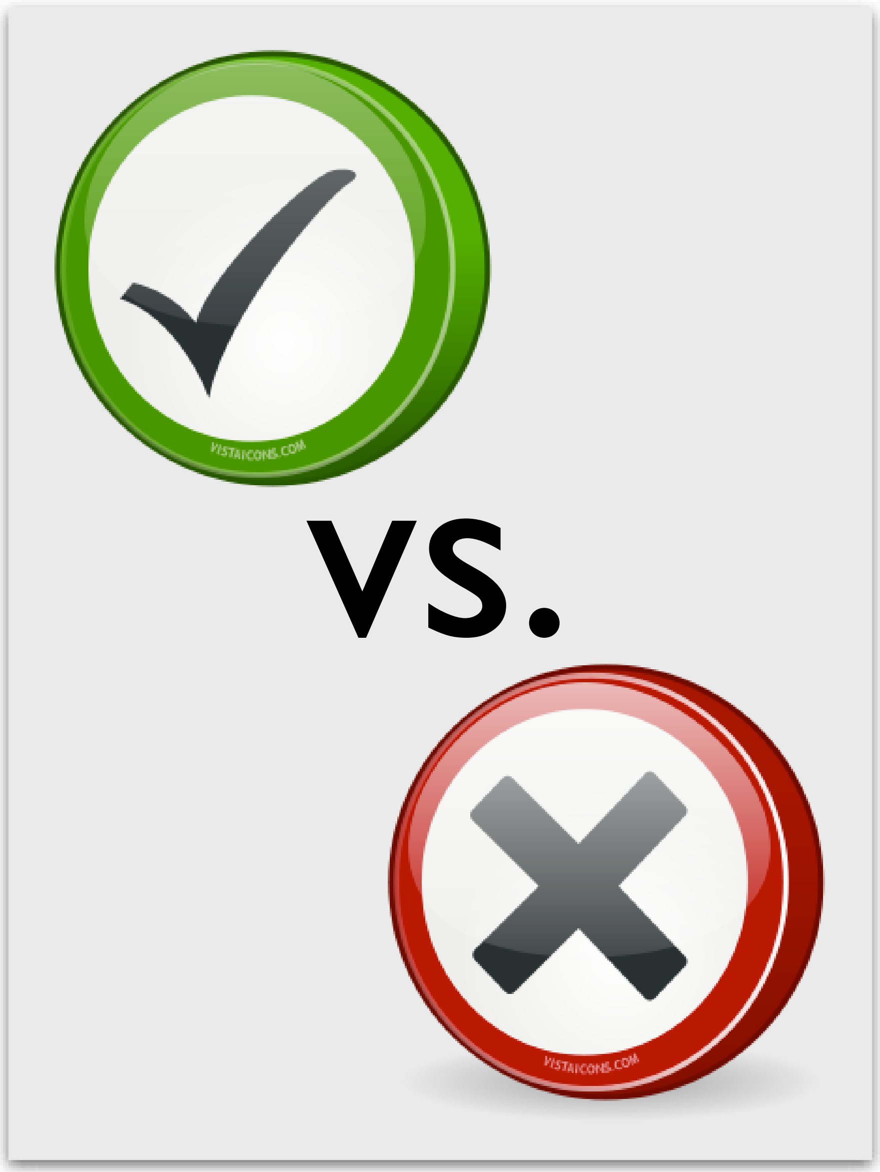
Rahul Premraj
Tom Zimmermann

Project #1
Quality of bug reports

What makes a good bug report?

How do they differ from bad ones?

Related:
Cool Paper Title
[Your names here, Big SE conference]



Project #2

Predicting lifetime of bug reports

How long will a bug exist?

What influences its time to fix?

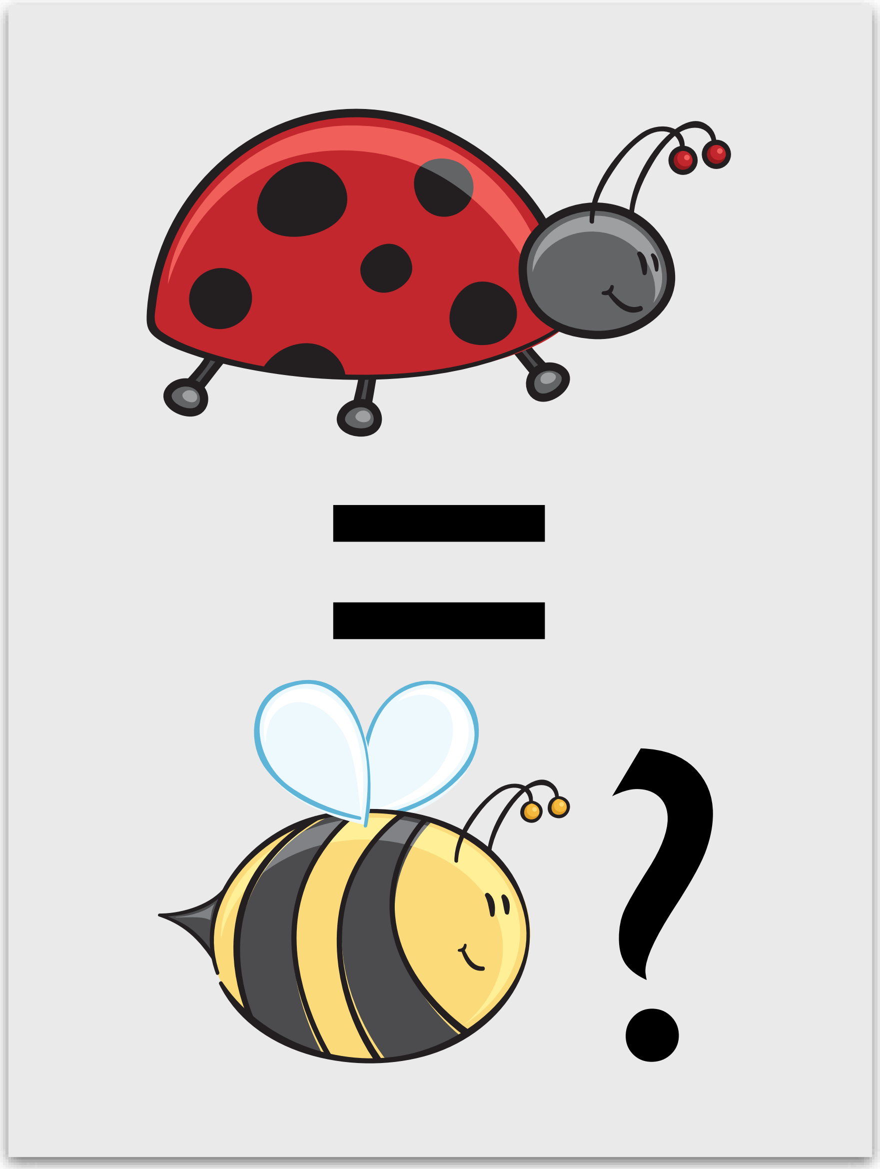
Related:
Predicting Eclipse Bug Lifetimes
[Panjer, Mining Challenge 2007]

Project #3

Detecting duplicates of bug reports

Is a new bug report a duplicate of another?

Related:
Detection of Duplicate Defect Reports Using Natural Language Processing [Runeson et al., ICSE 2007]



Project #4

Triaging bug reports

How can we assign bugs to developers?

Related:
Who Should Fix This Bug?
[Anvik et al., ICSE 2006]

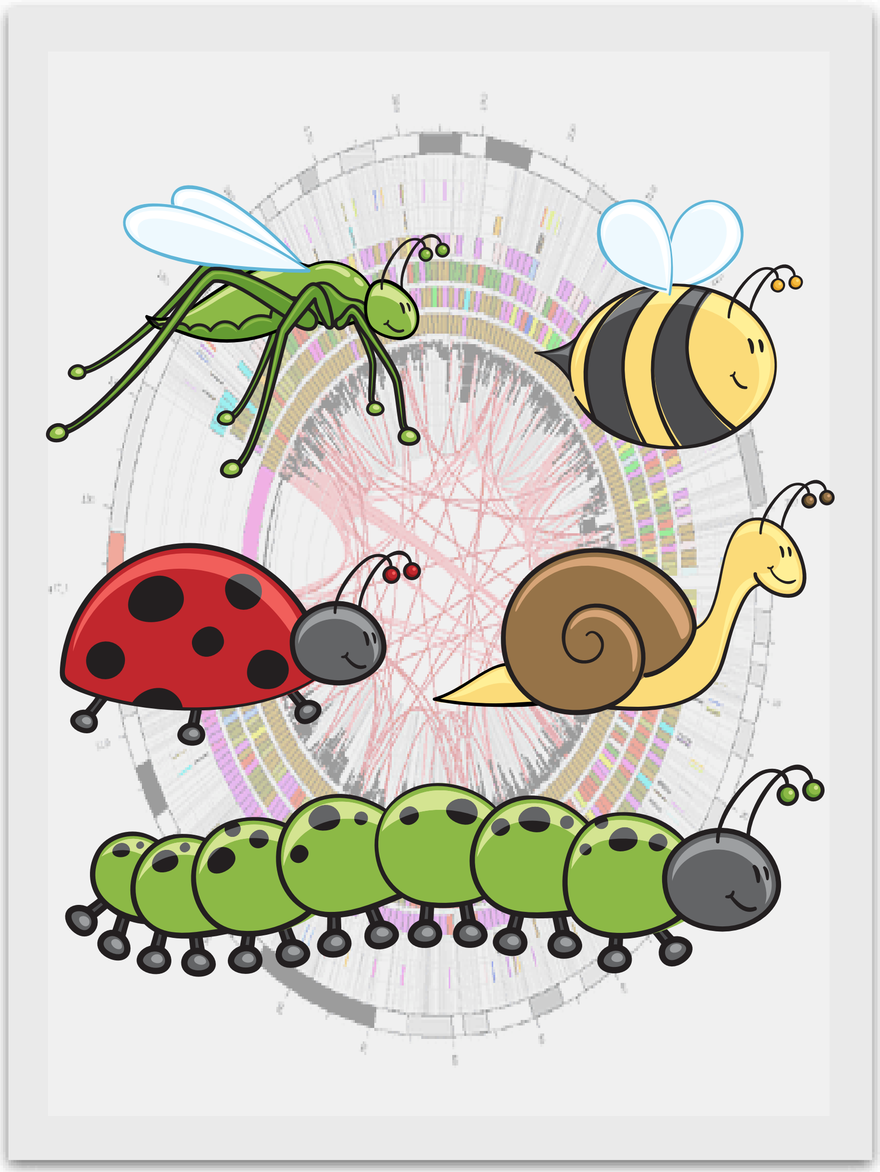
Project #5

Visualizing bug reports (networks)

How can we visualize individual bugs?

How can we visualize all bugs?

Related:
Software Bugs and Evolution: A Visual Approach to Uncover Their Relationships [D'Ambros et al., CSMR 2006]

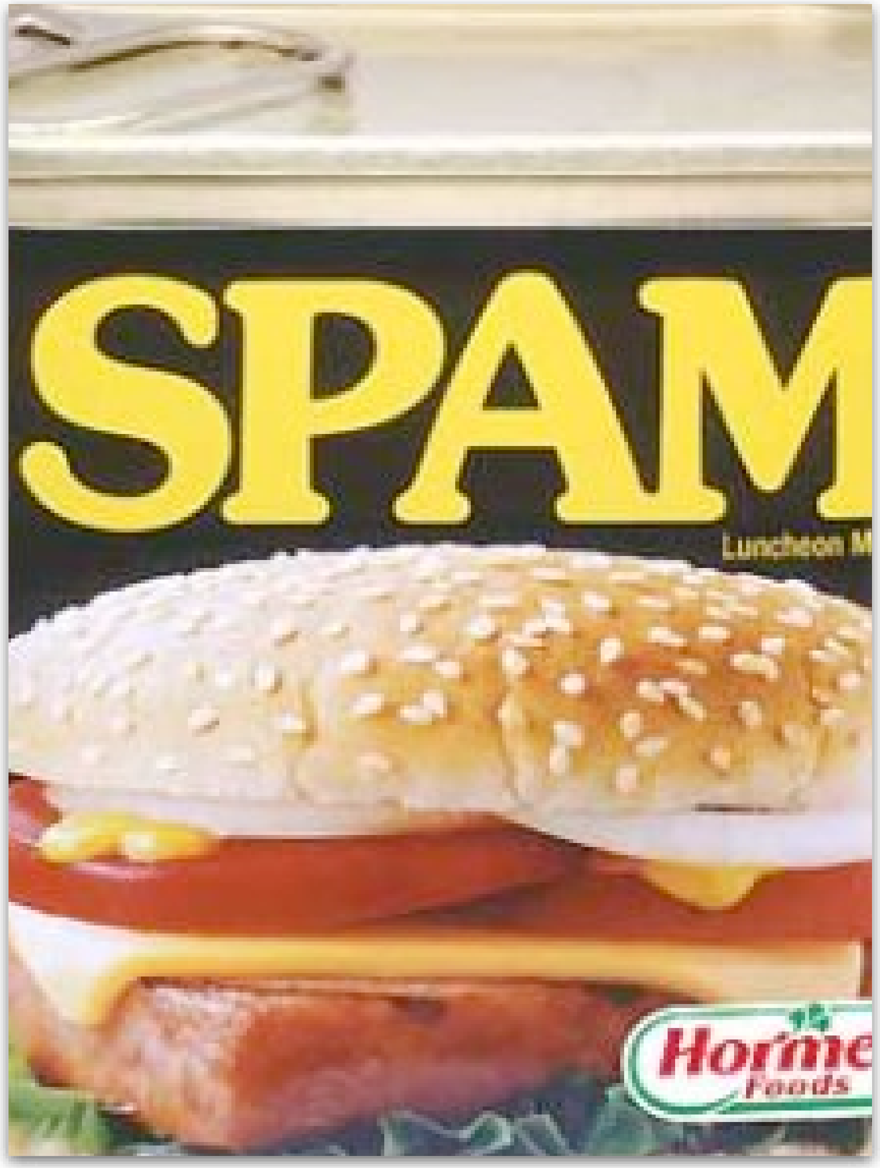


Project #6

Predicting defects with spam filters

Which files have bugs?
Which ones don't?

Related:
Spam Filter Based Approach for Finding Fault-Prone Software Modules
[Mizuno et al., MSR 2007]



Pattern mining

Mining Version Histories to Guide Software Changes
(Lecture 3)

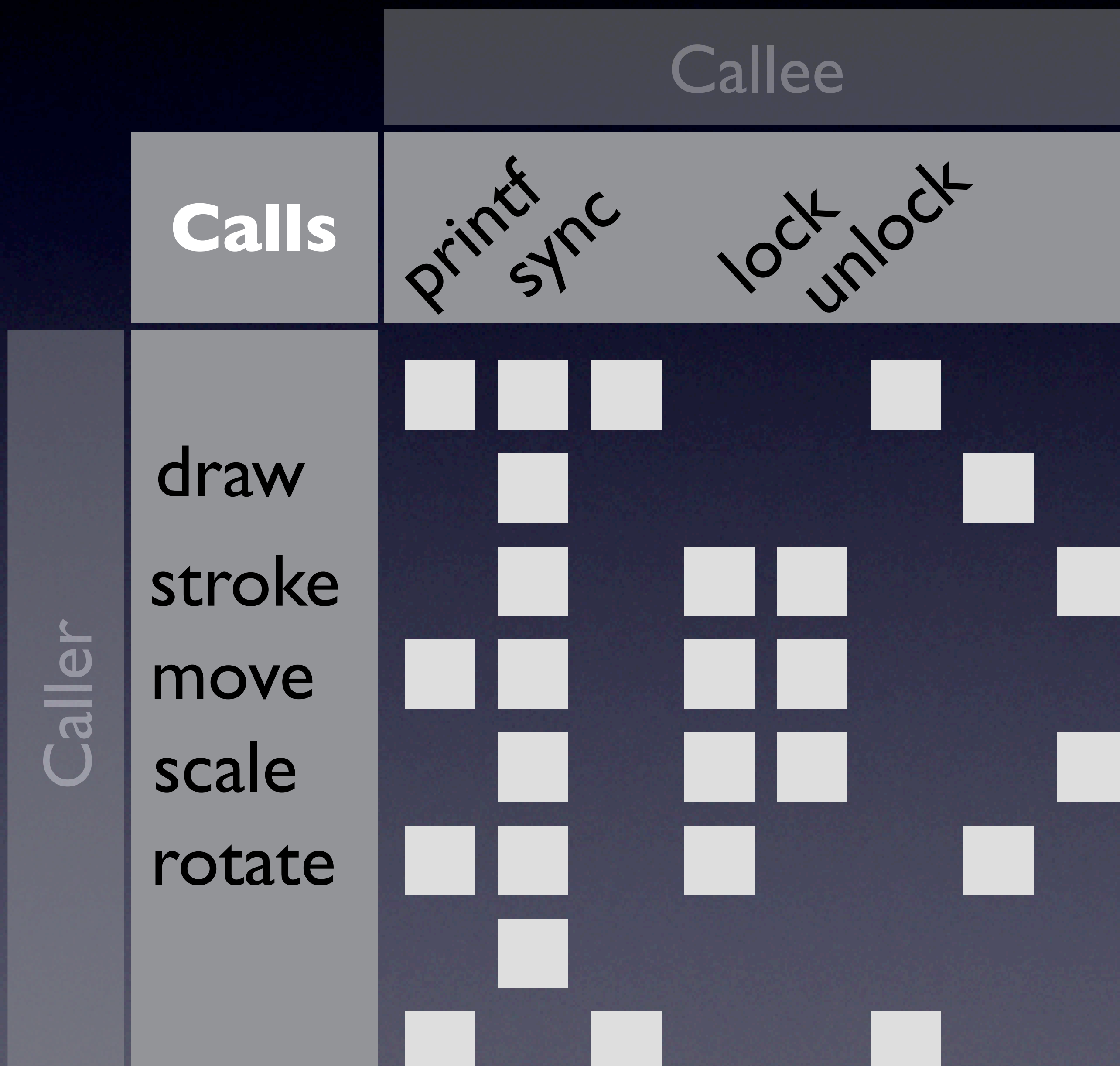
Xelopes Library

Commercial?

http://www.prudsys.com/Produkte/Algorithmen/Xelopes/

Slide provided by C. Lindig

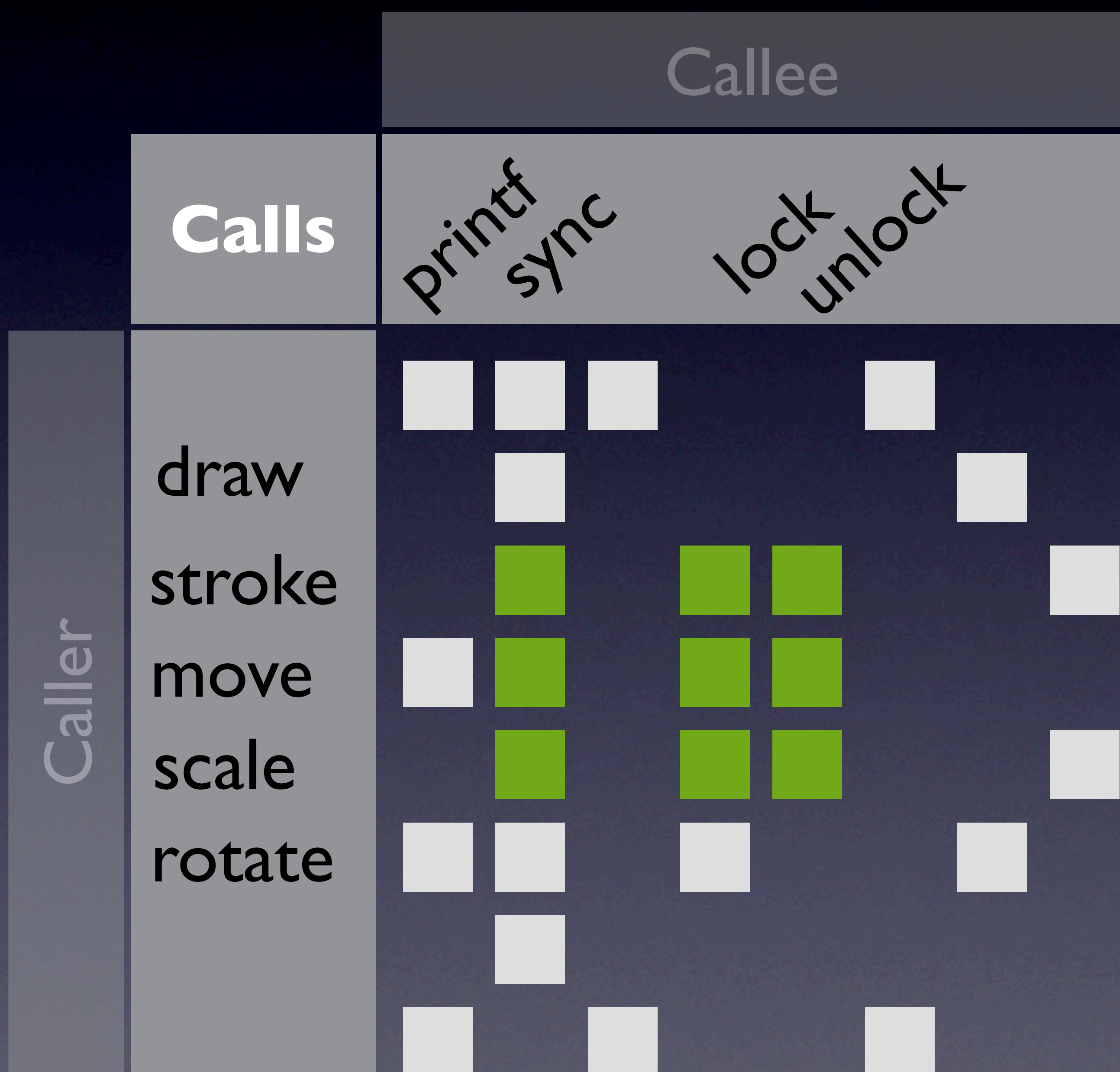
The Call relation



The call relation is easy to obtain by disassembling executable C code.

Slide provided by C. Lindig

The Call relation

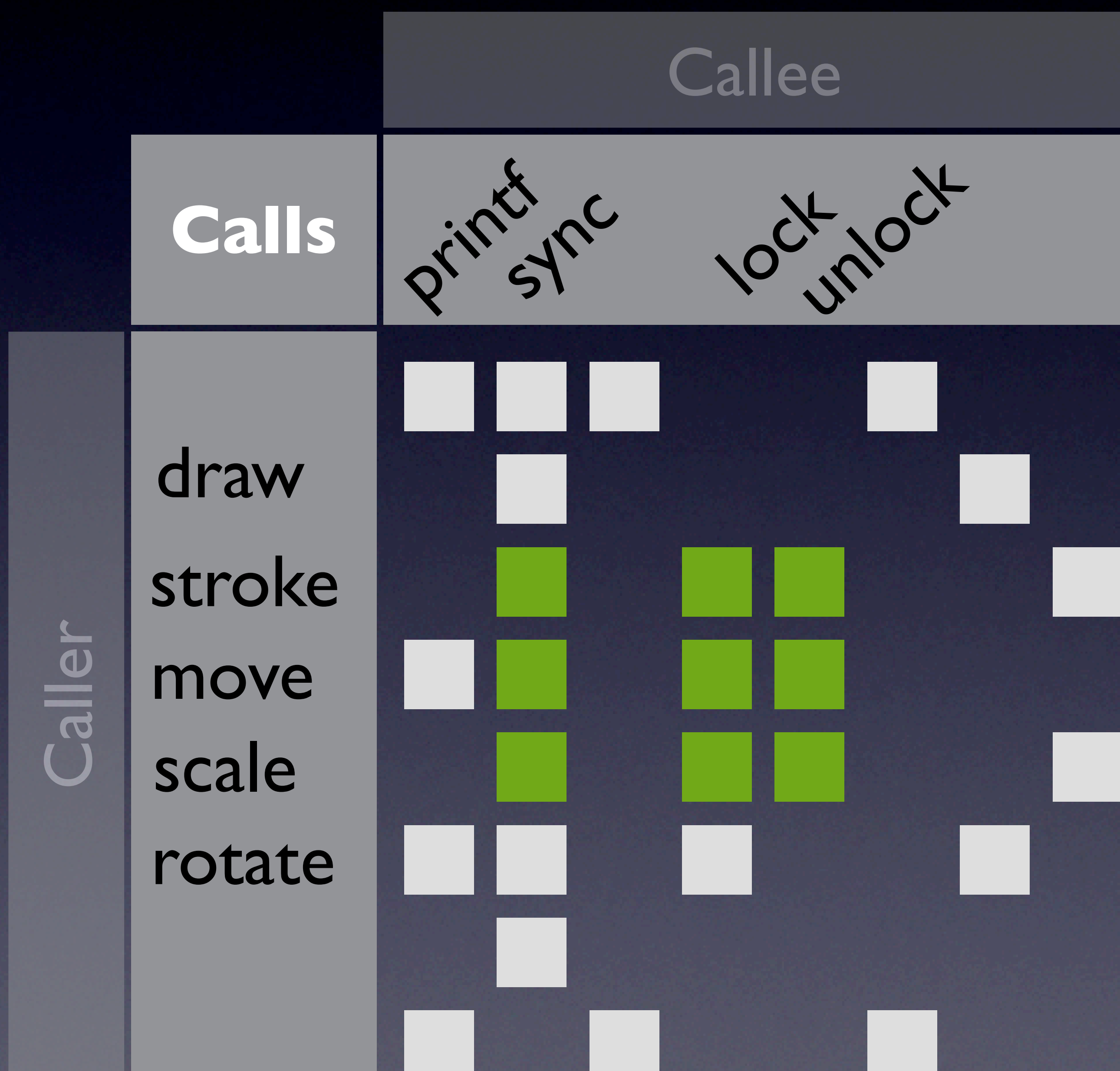


Blocks are patterns!

The call relation is easy to obtain by disassembling executable C code.

Slide provided by C. Lindig

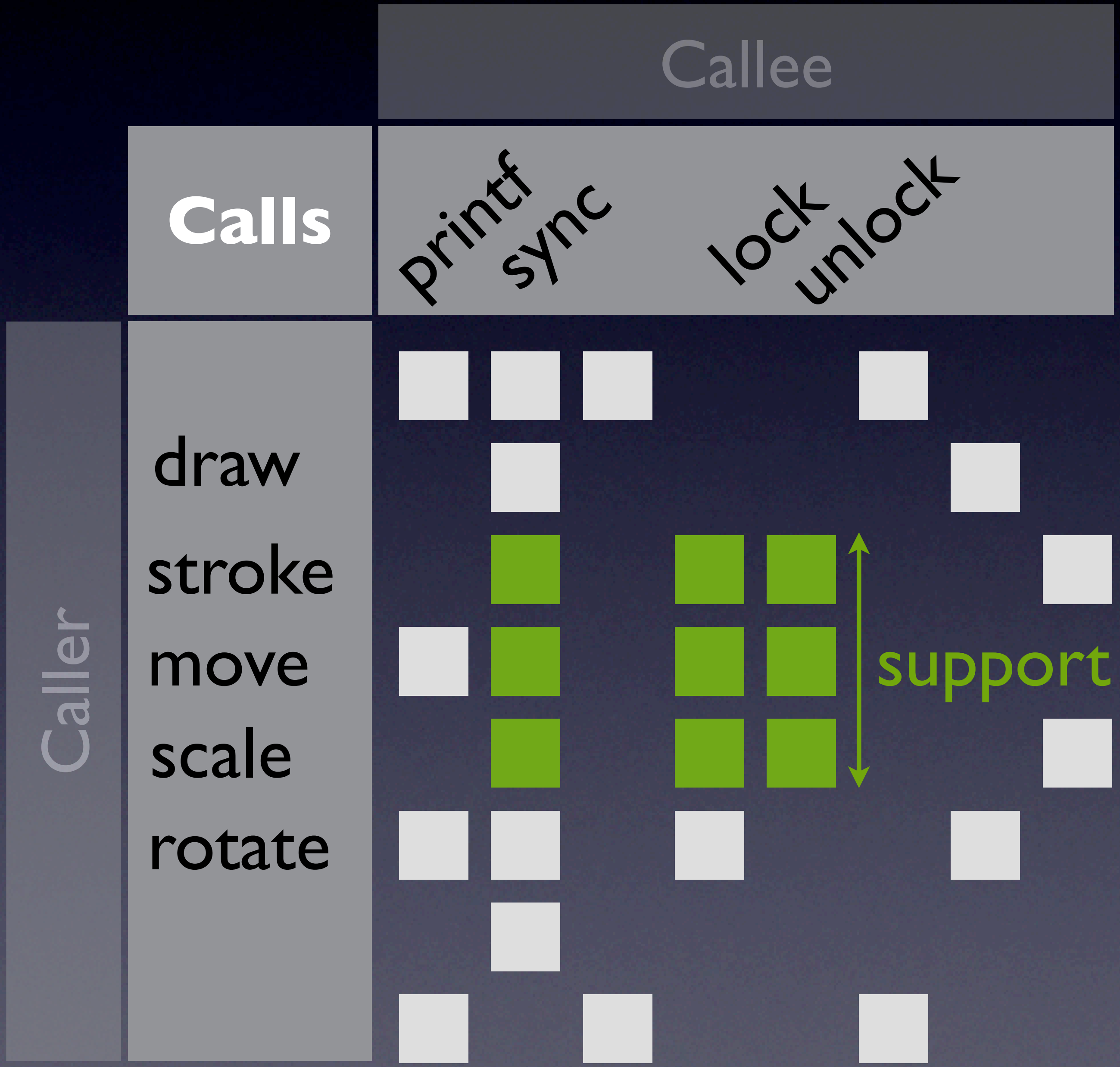
The Call relation



The call relation is easy to obtain by disassembling executable C code.

Slide provided by C. Lindig

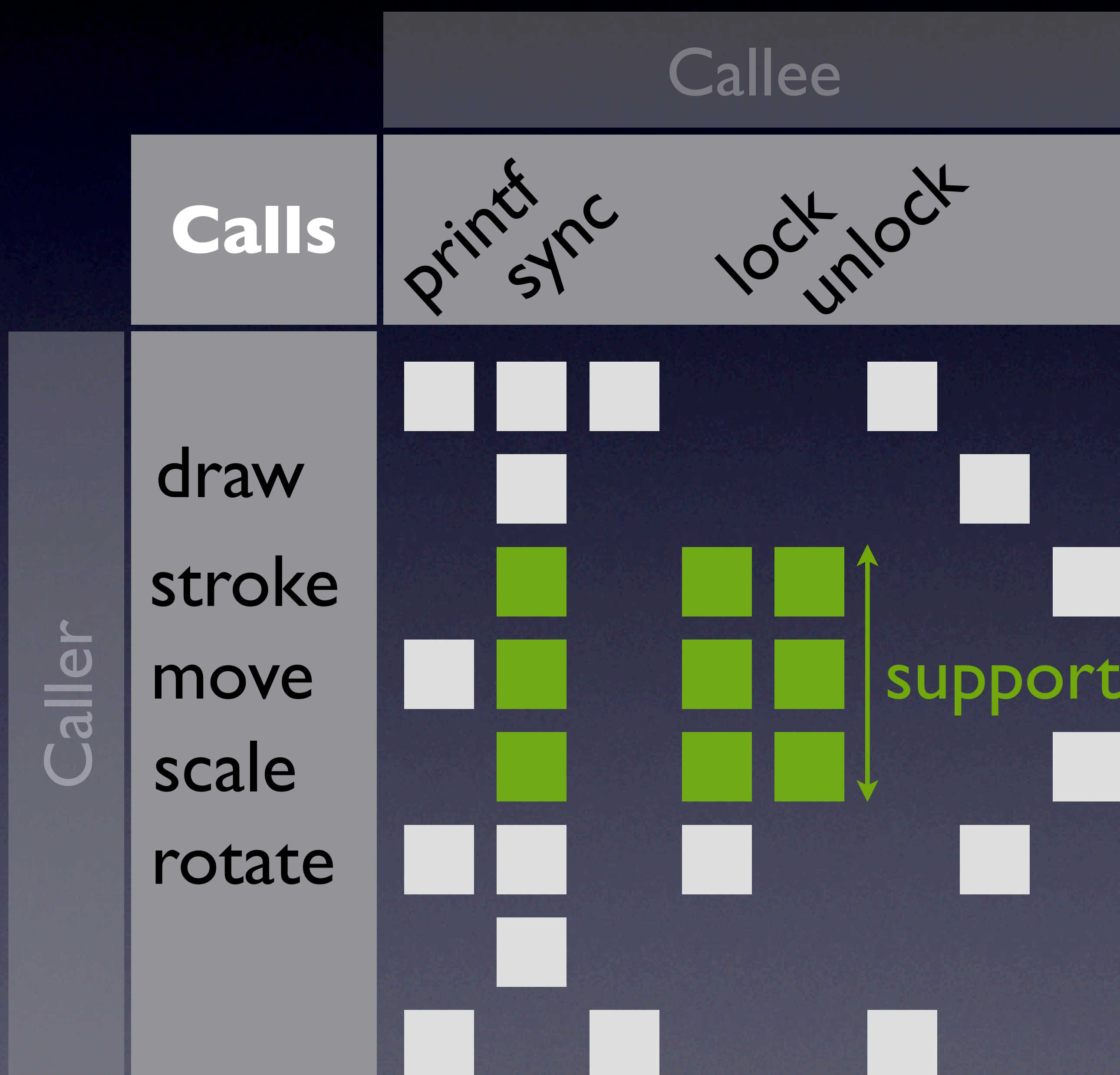
The Call relation



The call relation is easy to obtain by disassembling executable C code.

Slide provided by C. Lindig

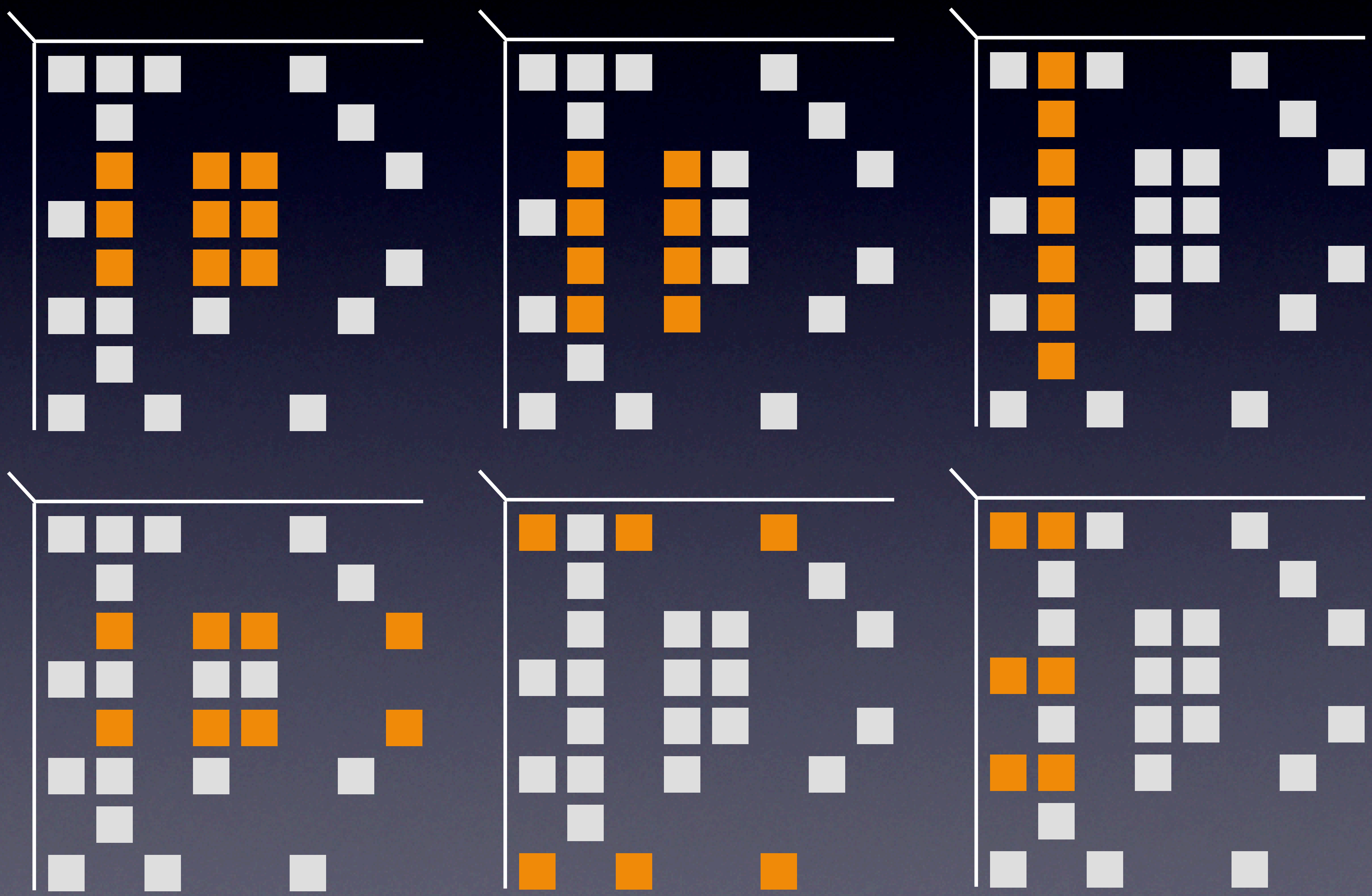
The Call relation



{sync, lock, unlock}
has support = 3
and size = 3

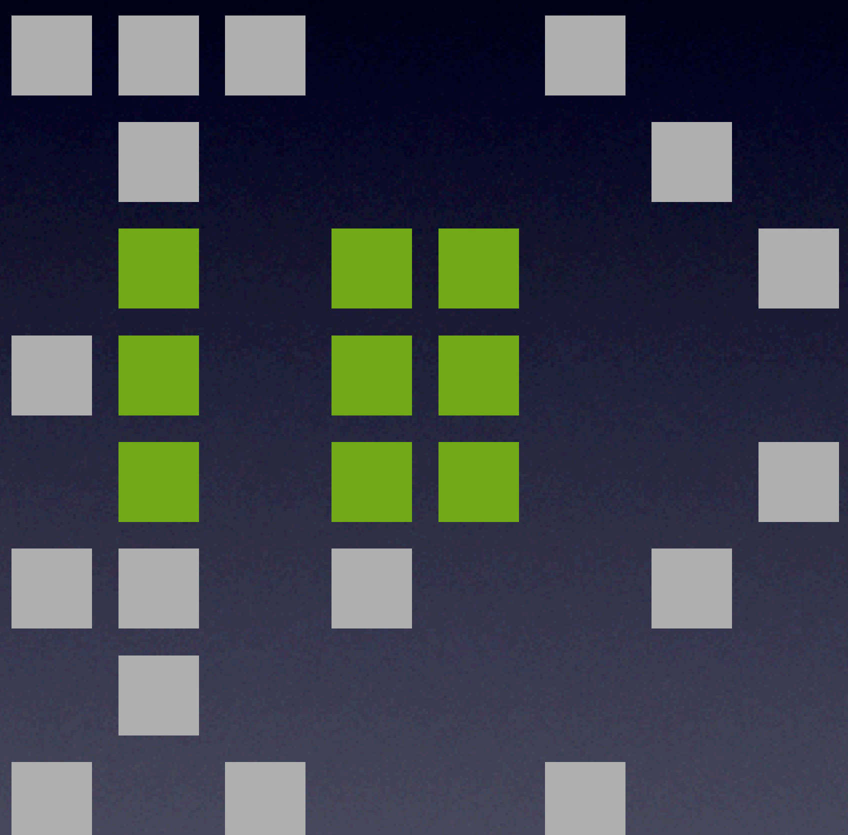
The call relation is easy to obtain by disassembling executable C code.

Many possible patterns



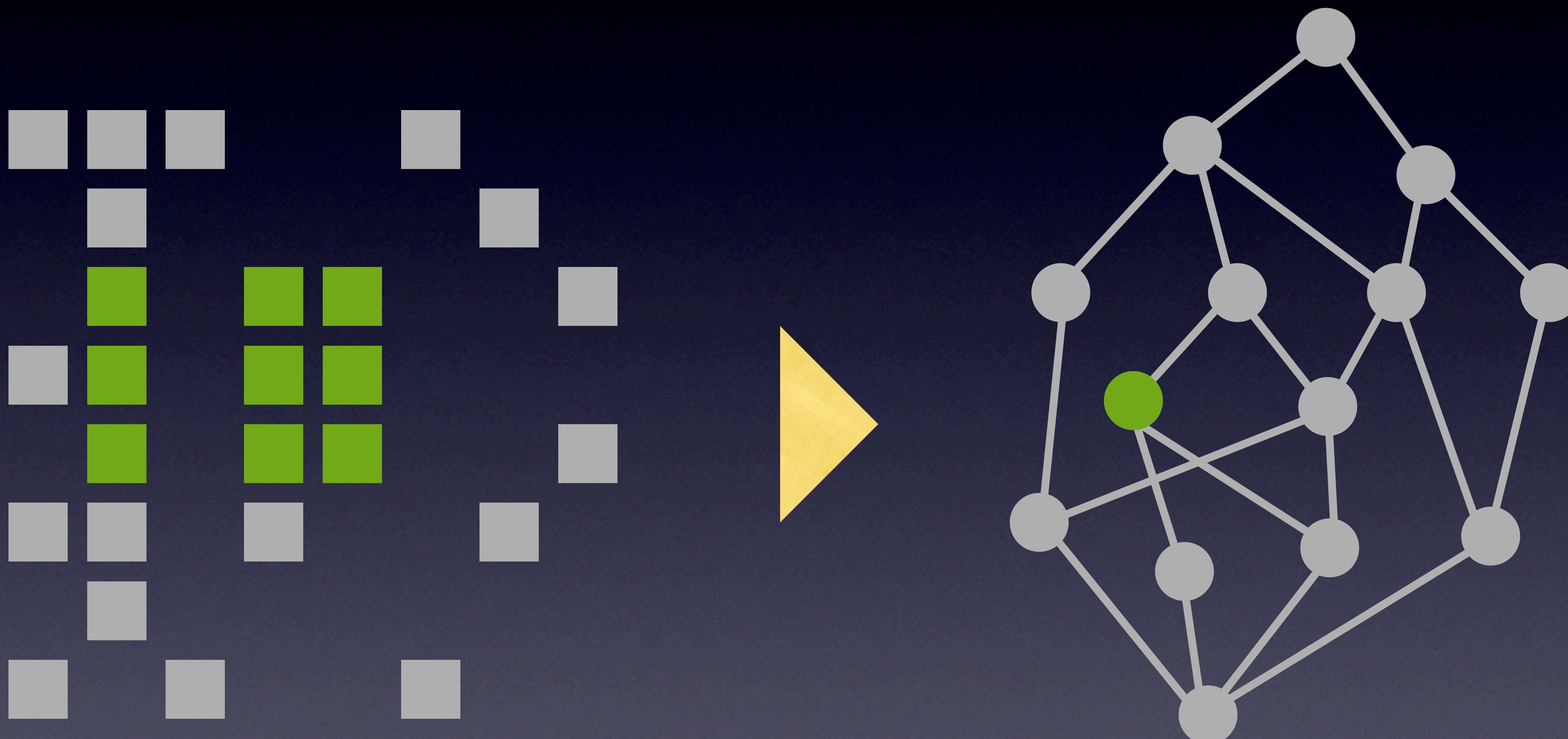
Slide provided by C. Lindig

Concept analysis



Slide provided by C. Lindig

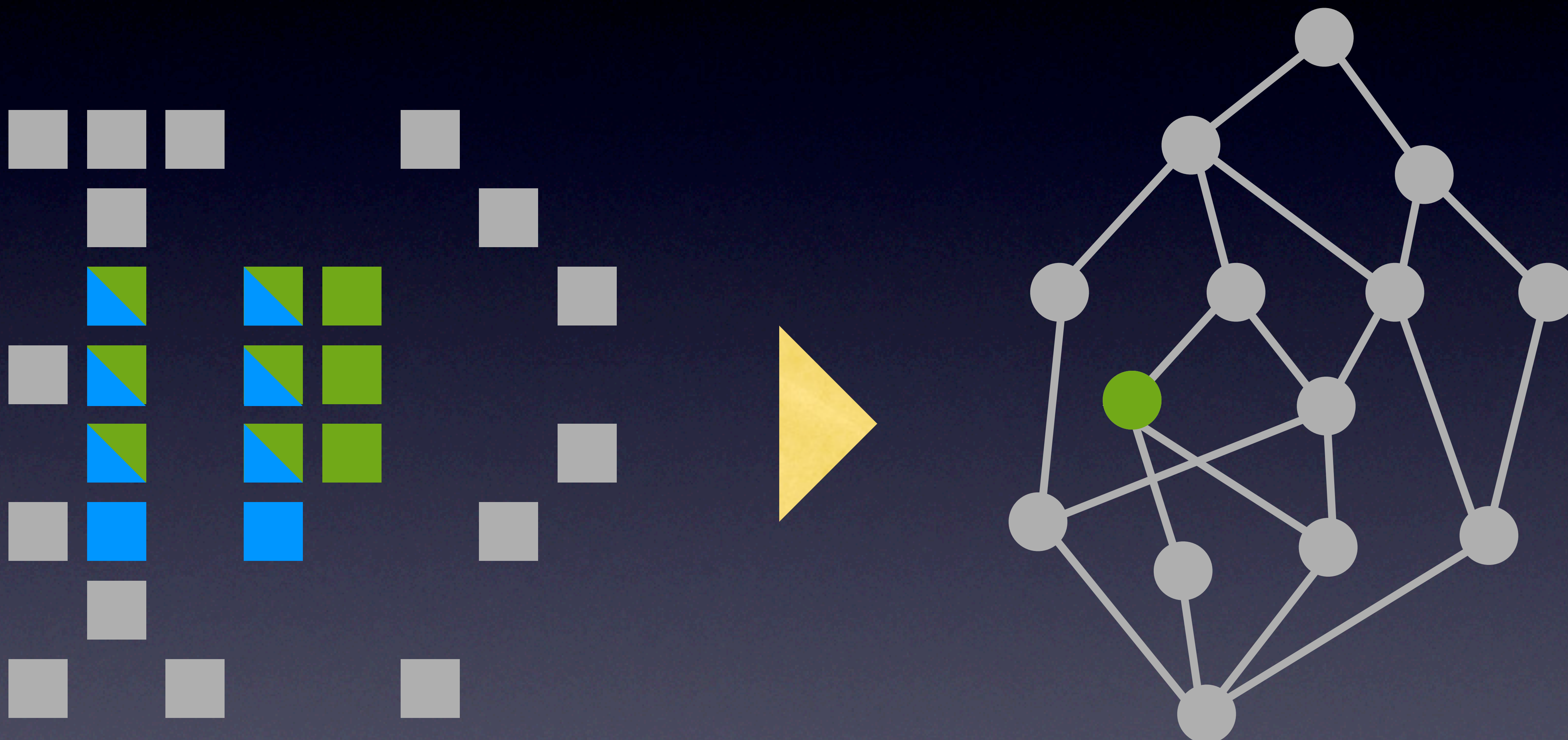
Concept analysis



Concept Analysis computes all blocks (patterns)

Slide provided by C. Lindig

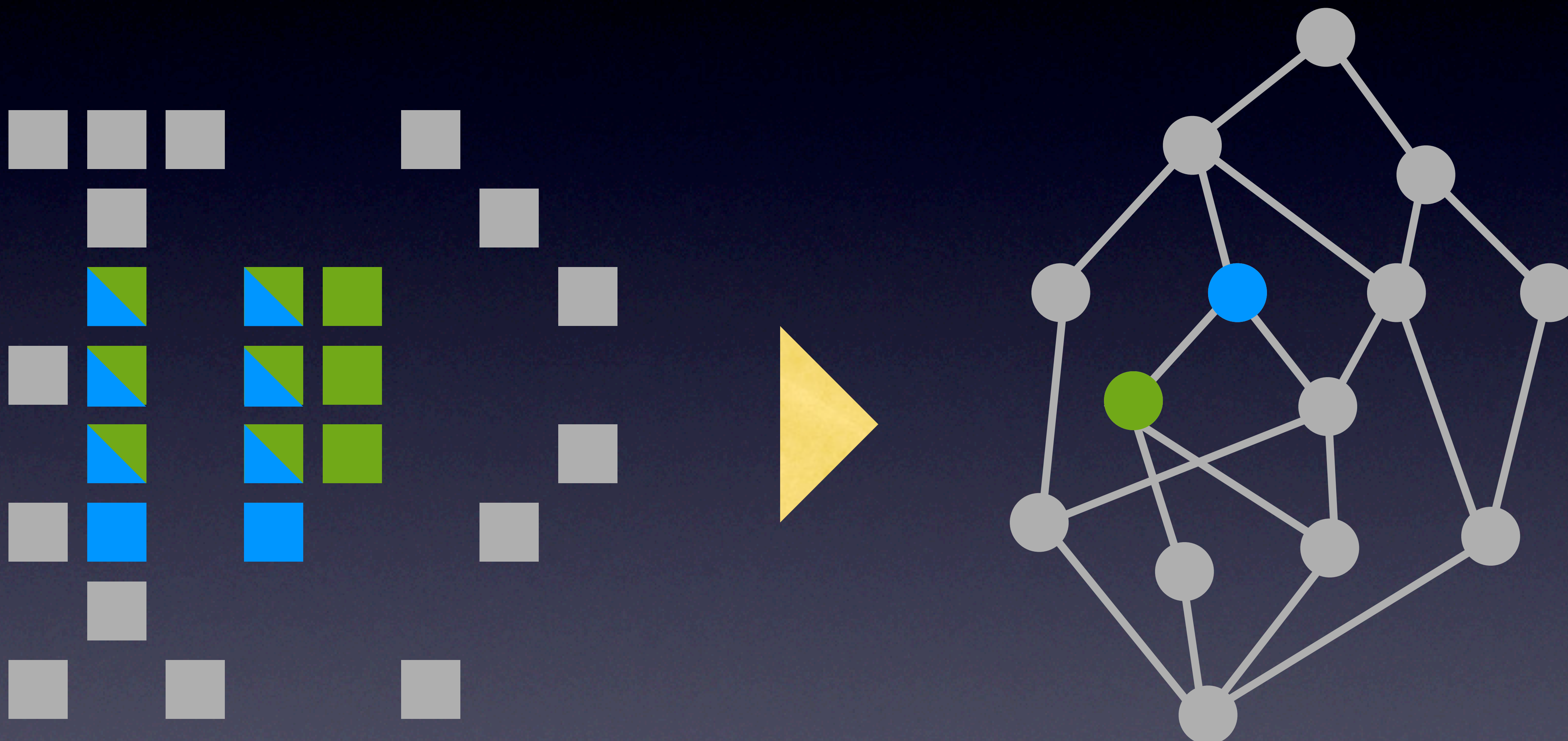
Concept analysis



Concept Analysis computes all blocks (patterns)

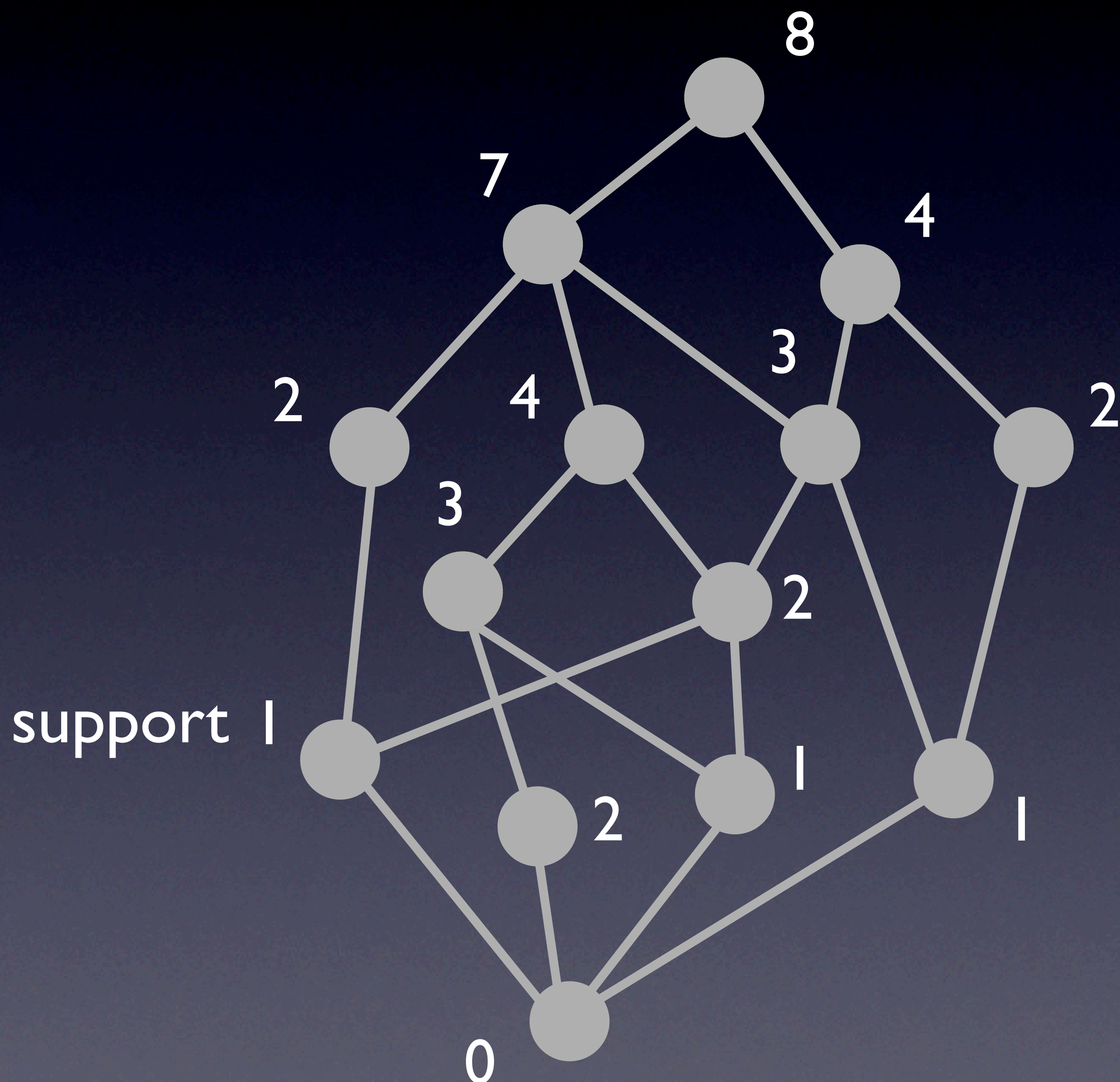
Slide provided by C. Lindig

Concept analysis

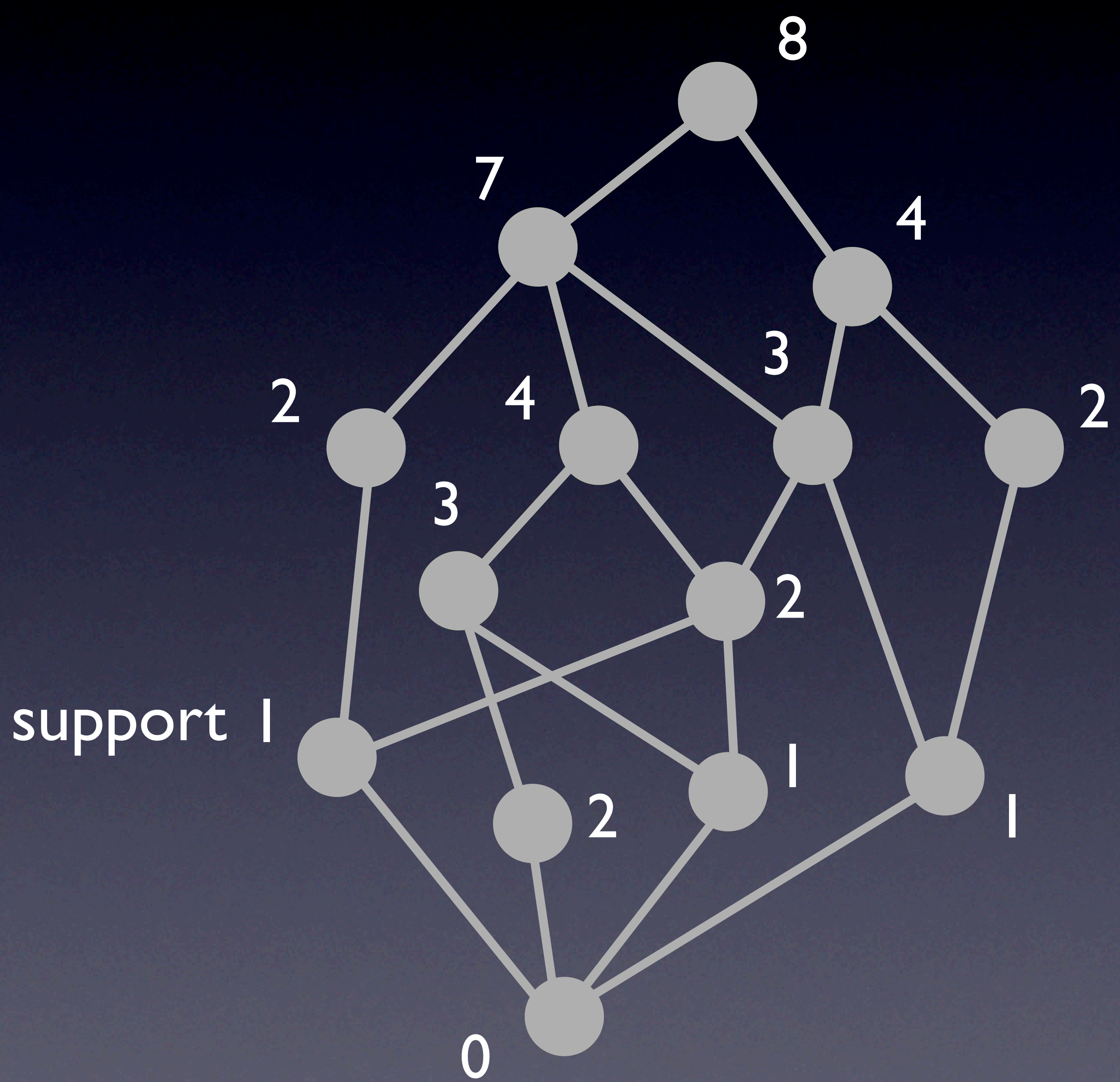


Concept Analysis computes all blocks (patterns)

Support vs. Size



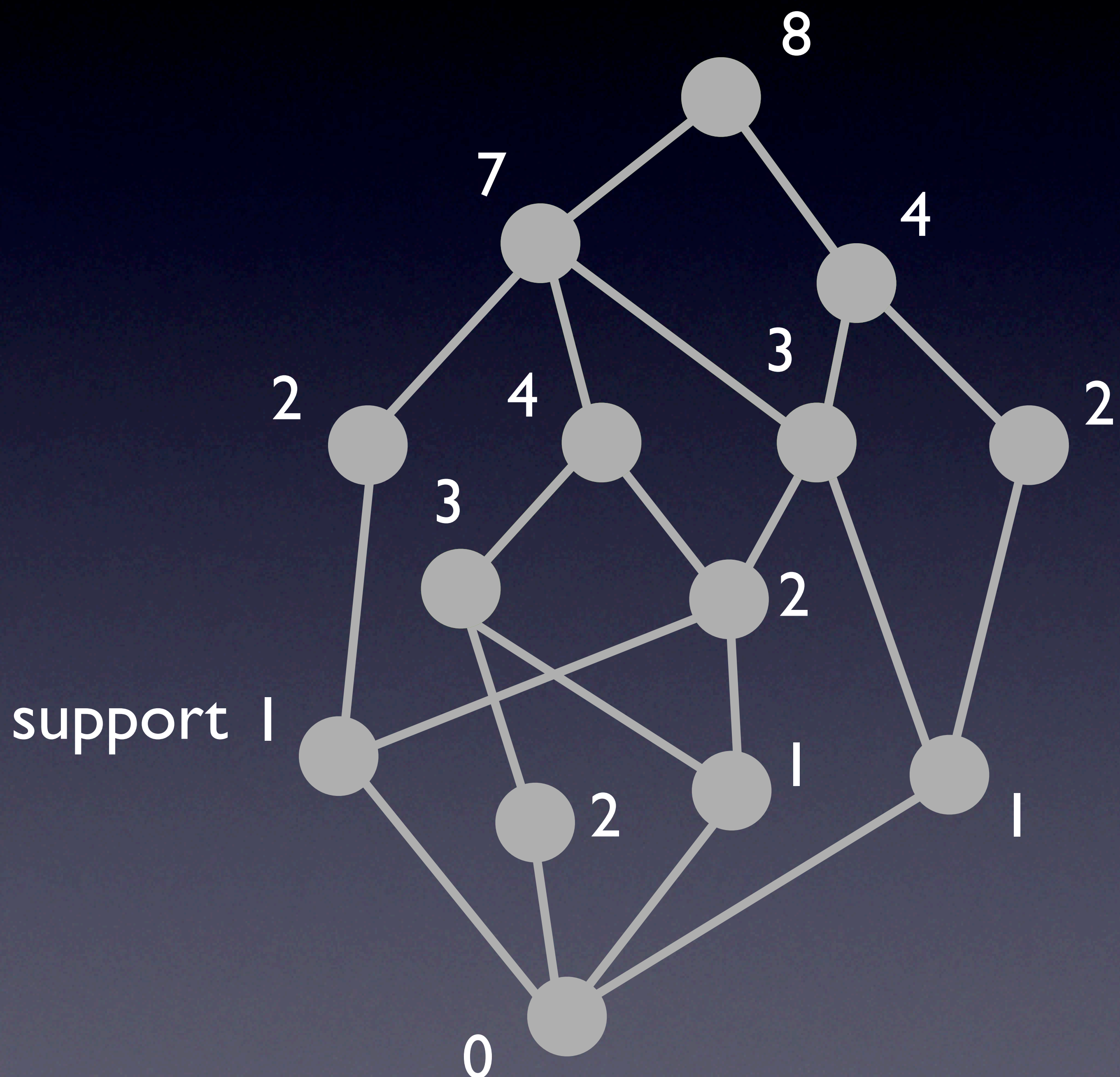
Support vs. Size



Support decreases monotonically from top to bottom.

Slide provided by C. Lindig

Support vs. Size

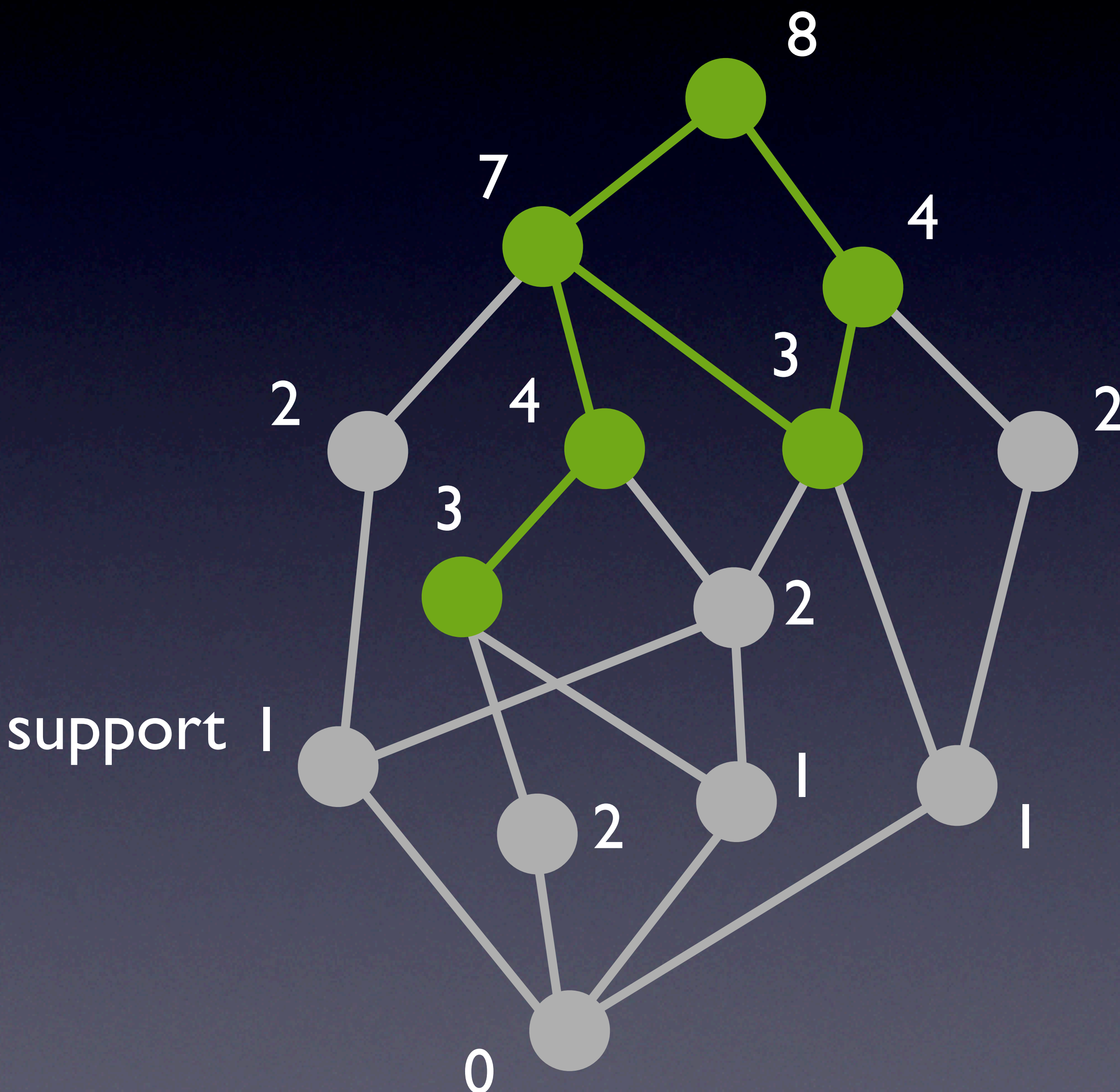


Support decreases monotonically from top to bottom.

Support can be used as a cut-off criterion (support ≥ 3).

Slide provided by C. Lindig

Support vs. Size

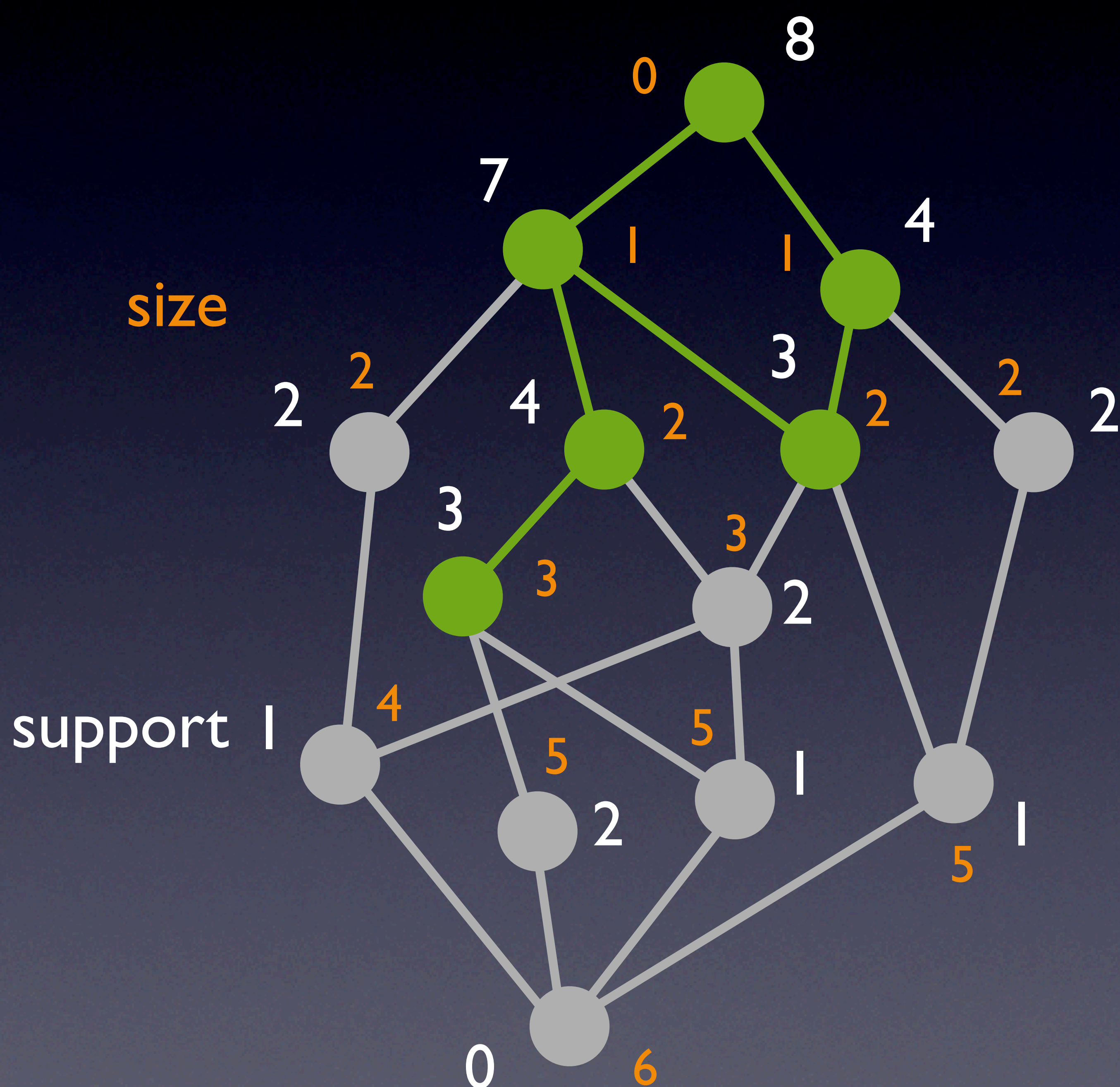


Support decreases monotonically from top to bottom.

Support can be used as a cut-off criterion (support ≥ 3).

Slide provided by C. Lindig

Support vs. Size



Support decreases monotonically from top to bottom.

Support can be used as a cut-off criterion (support ≥ 3).

Size increases monotonically from top to bottom.

Slide provided by C. Lindig

Example patterns

Subject	Pattern
Ruby CVS	va_end va_start
Linux 2.6 (s)	mutex_lock mutex_unlock
Linux 2.6 (xs)	kmem_cache_alloc kmem_cache_free
Python SVN	PyErr_SetString PyExc_TypeError
Ruby CVS	rb_fix2int rb_num2int

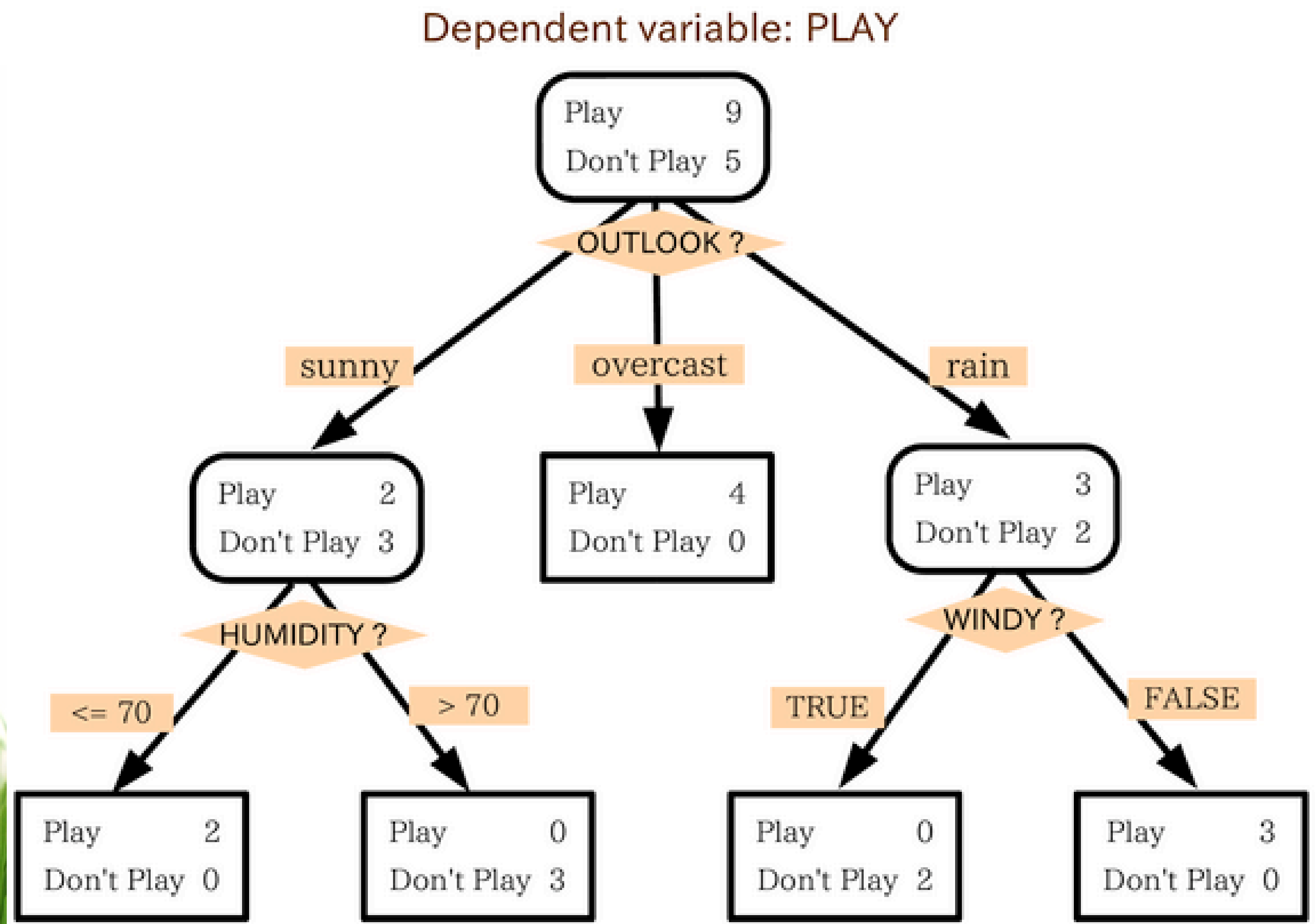
Decision trees

Decision trees

Play golf dataset

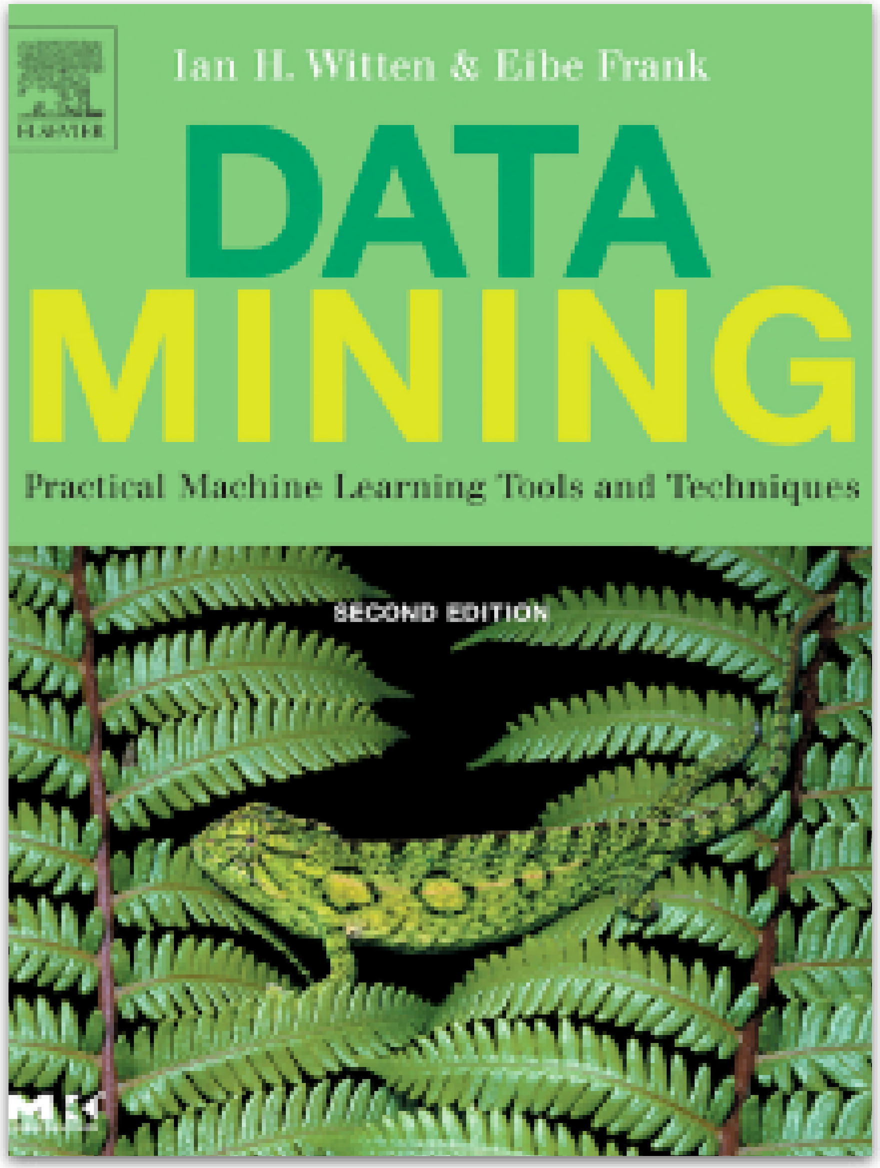
Independent variables				Dep. var
OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY
sunny	85	85	FALSE	Don't Play
sunny	80	90	TRUE	Don't Play
overcast	83	78	FALSE	Play
rain	70	96	FALSE	Play
rain	68	80	FALSE	Play
rain	65	70	TRUE	Don't Play
overcast	64	65	TRUE	Play
sunny	72	95	FALSE	Don't Play
sunny	69	70	FALSE	Play
rain	75	80	FALSE	Play
sunny	75	70	TRUE	Play
overcast	72	90	TRUE	Play
overcast	81	75	FALSE	Play
rain	71	80	TRUE	Don't Play

Decision trees



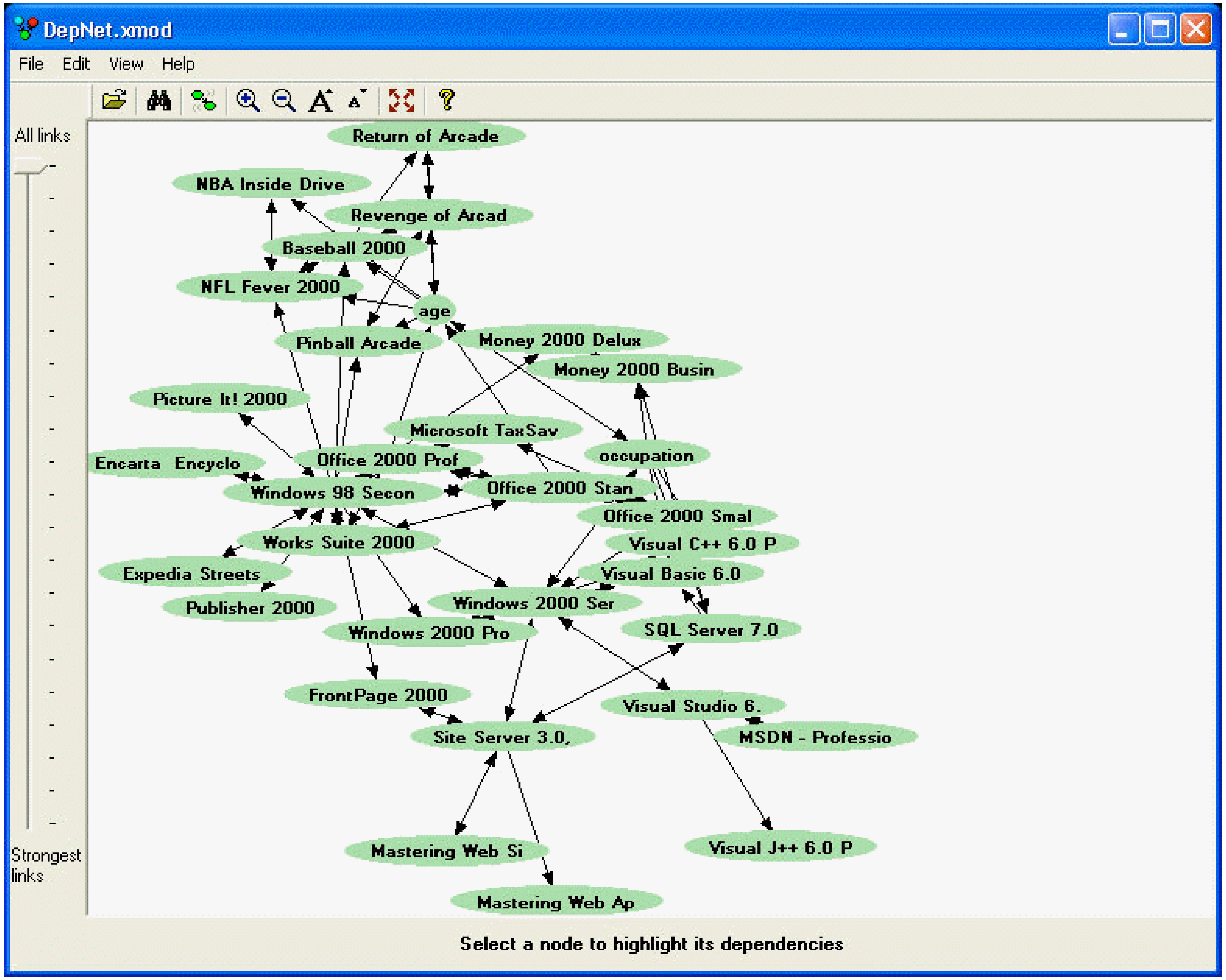
Weka

WEKA
The University
of Waikato



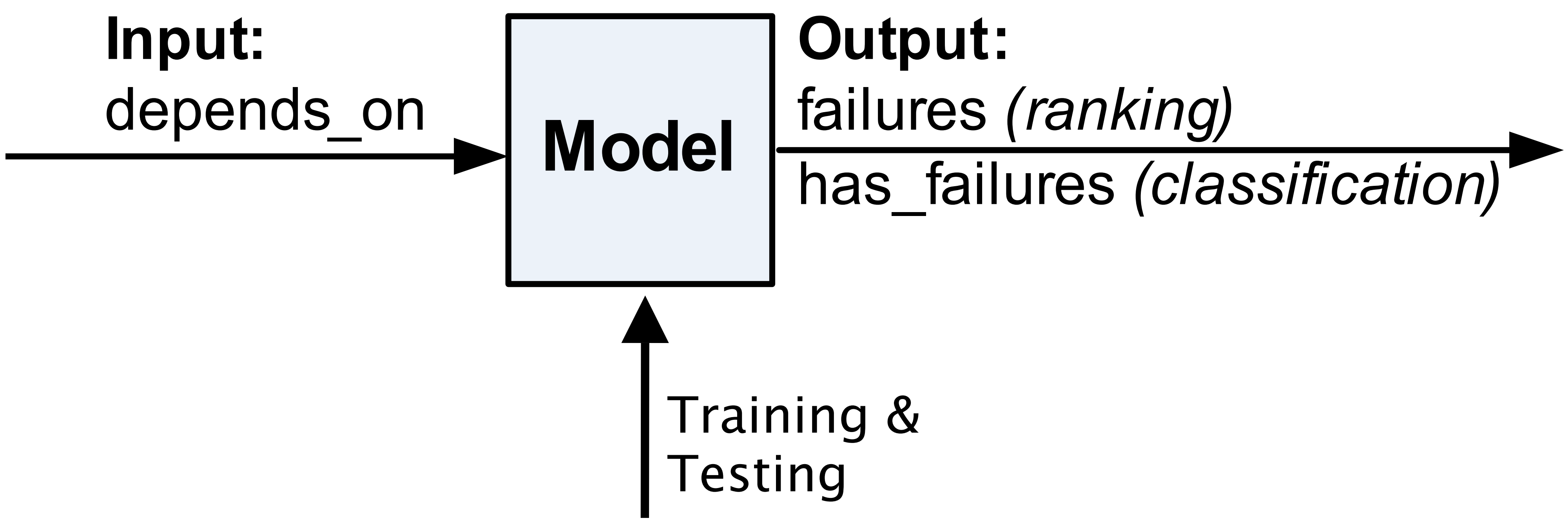
http://www.cs.waikato.ac.nz/ml/weka/

WinMine Toolkit



http://research.microsoft.com/~dmax/winmine/tooldoc.htm

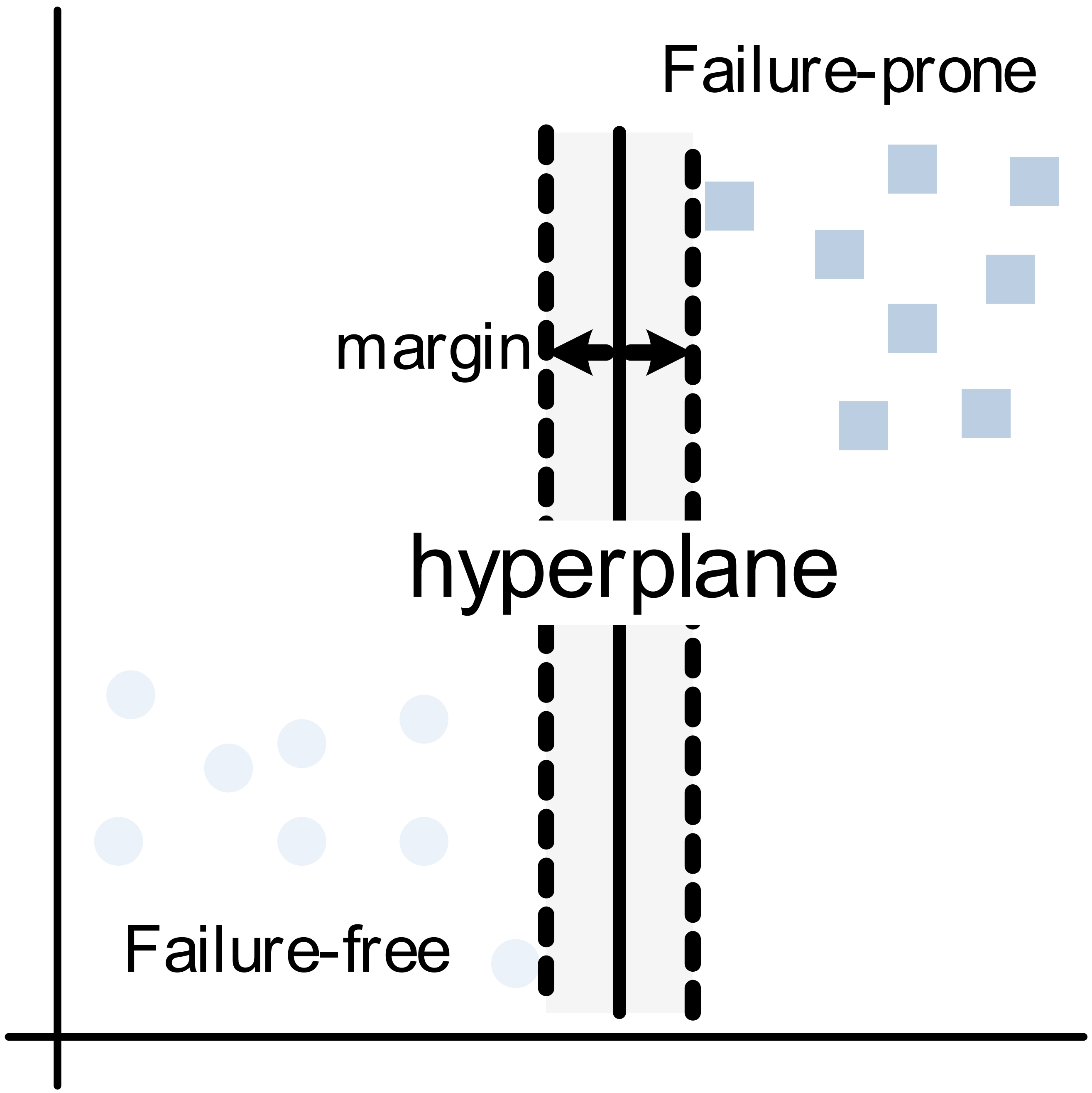
Classification with SVM



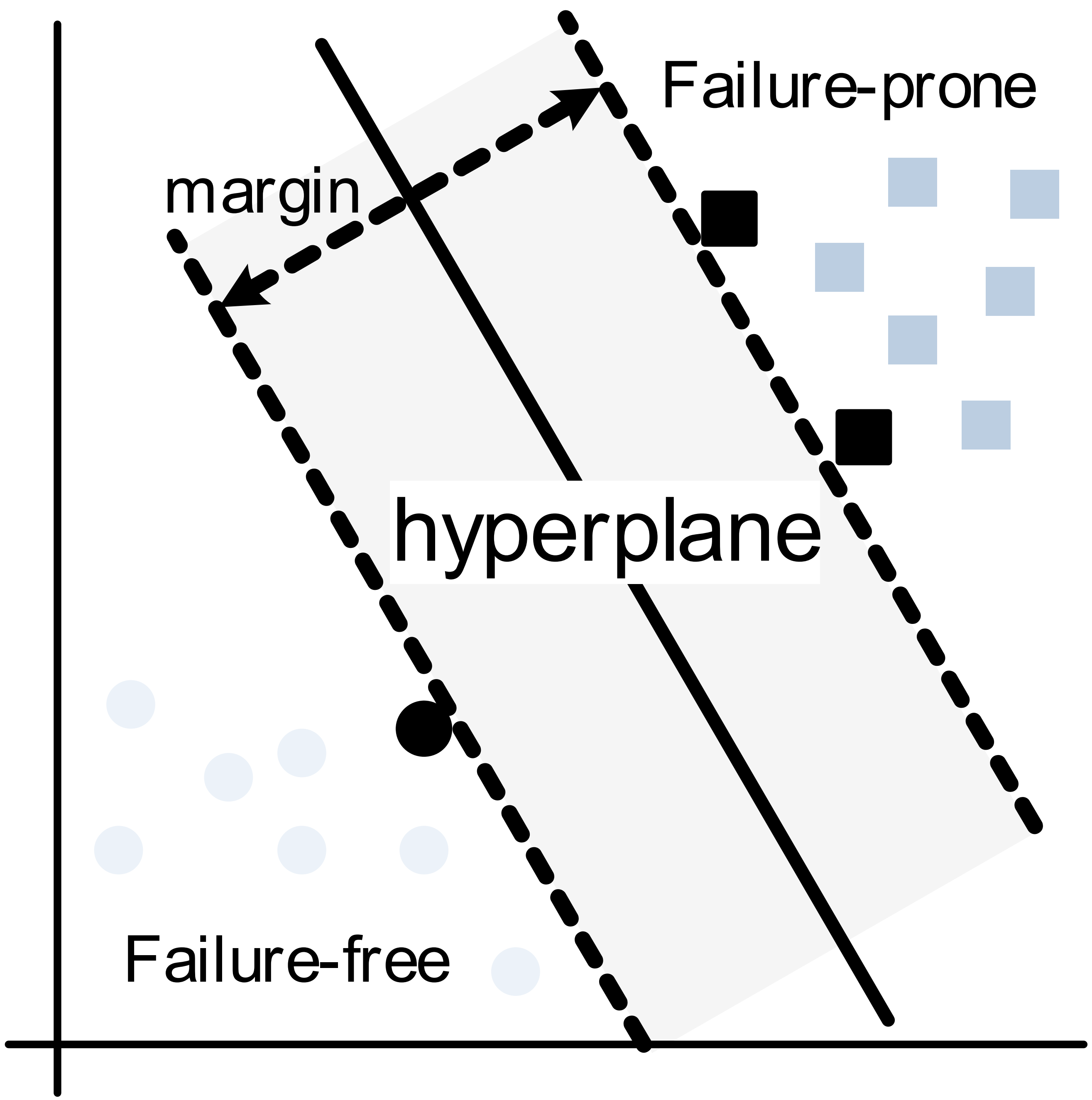
Dataset:

binary=i.dll	depends_on=h.dll m.dll	failures=0
binary=m.dll	depends_on=a.exe d.dll	failures=0
binary=h.dll	depends_on=e.exe j.exe	failures=0
binary=d.dll	depends_on=i.dll e.exe	failures=1
binary=l.dll	depends_on=g.exe b.dll	failures=0
binary=b.dll	depends_on=m.dll i.dl k.dlll	failures=0
binary=g.exe	depends_on=m.dll	failures=0

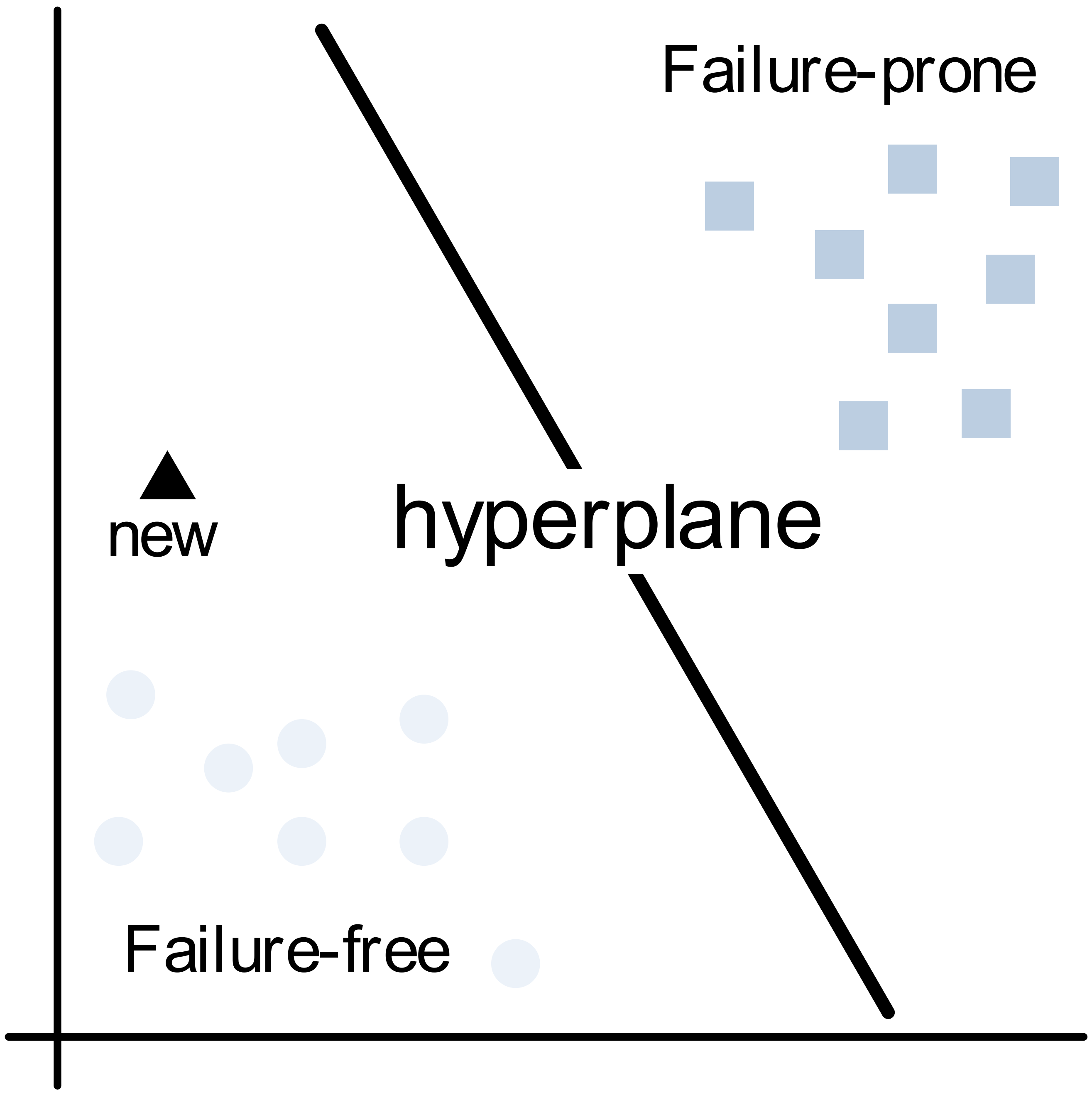
The black box: SVMs



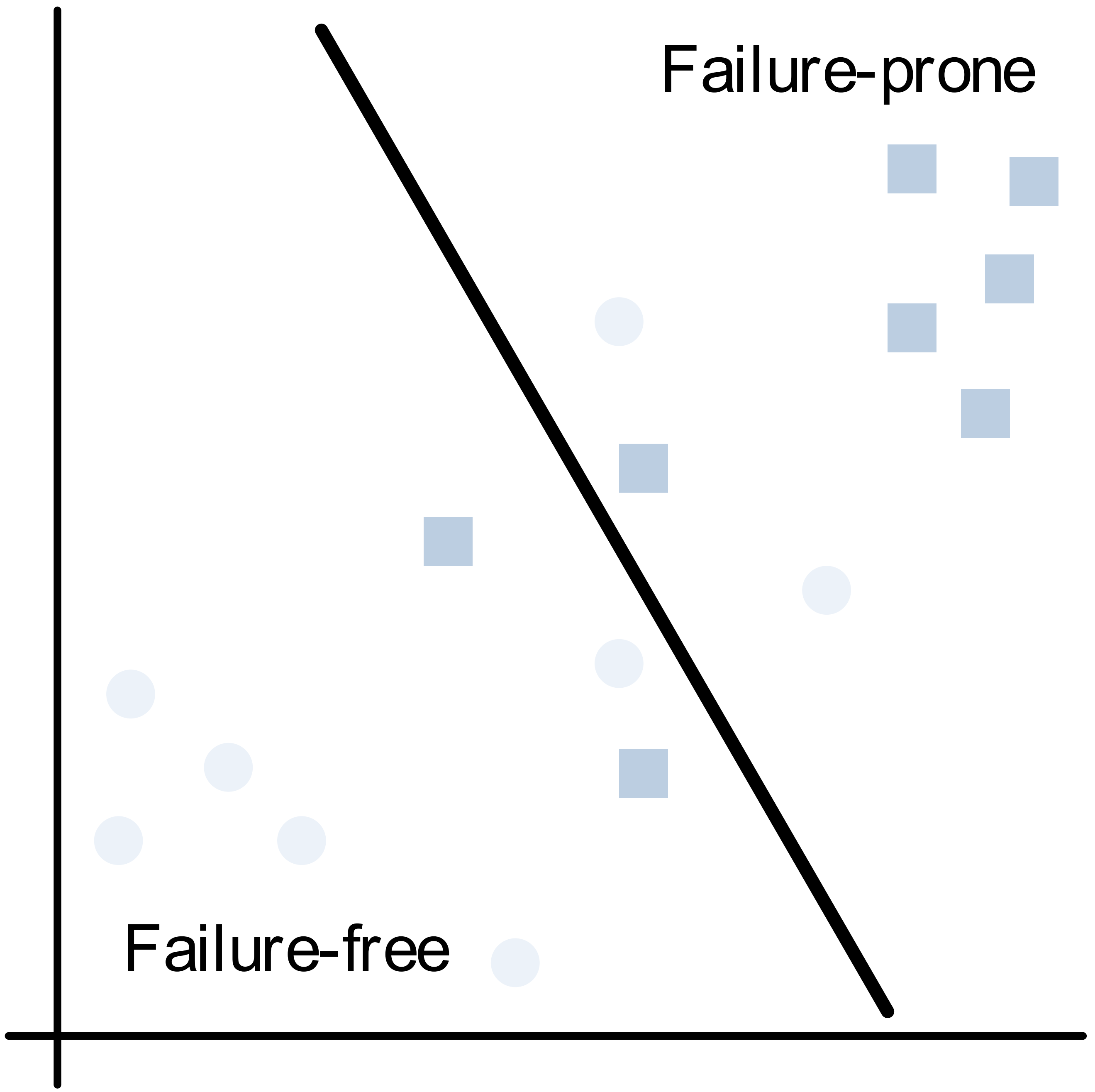
SVM: Maximal margins



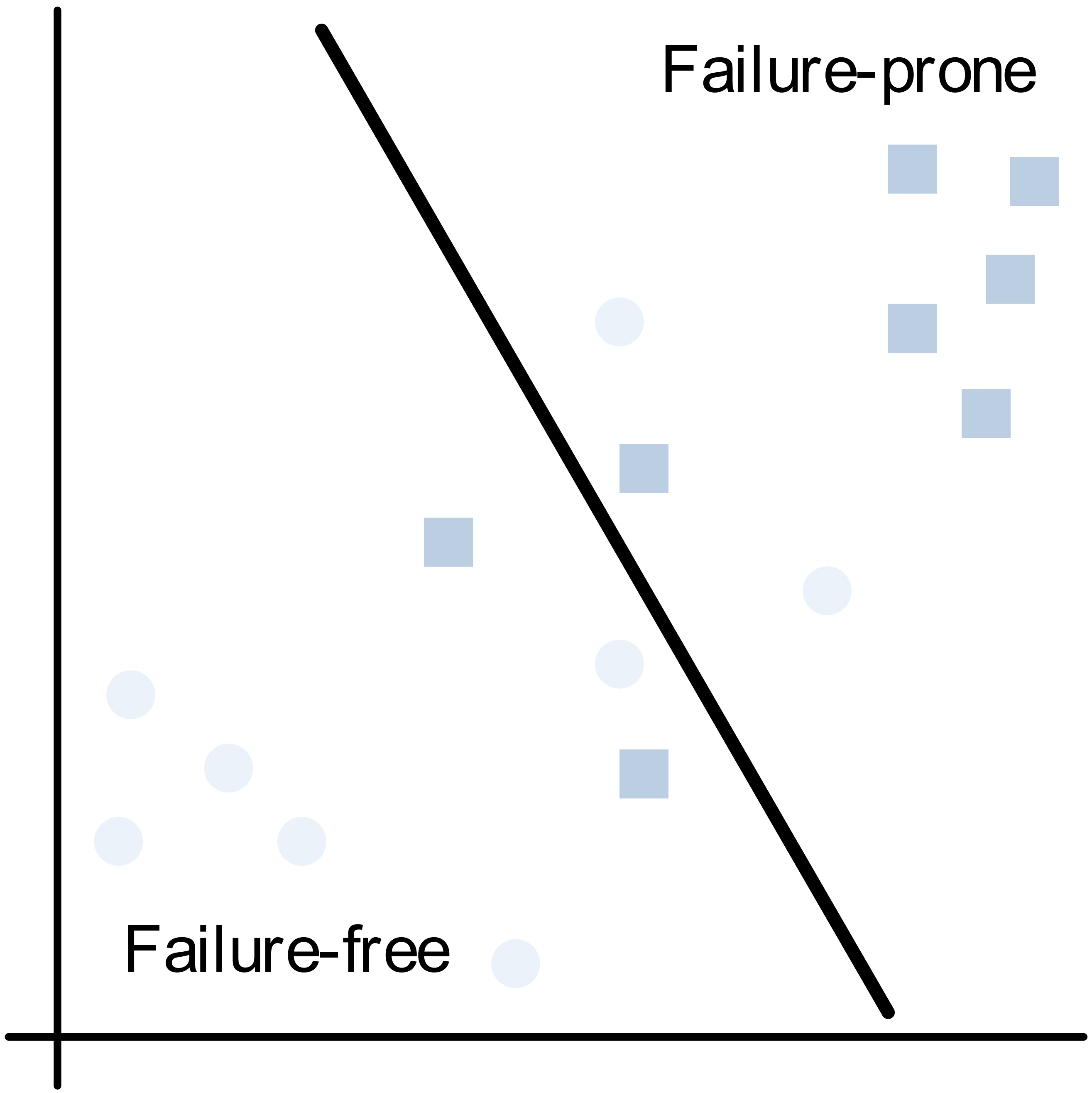
SVMs: Predictions



SVMs: Non-linear data



SVMs: Non-linear data

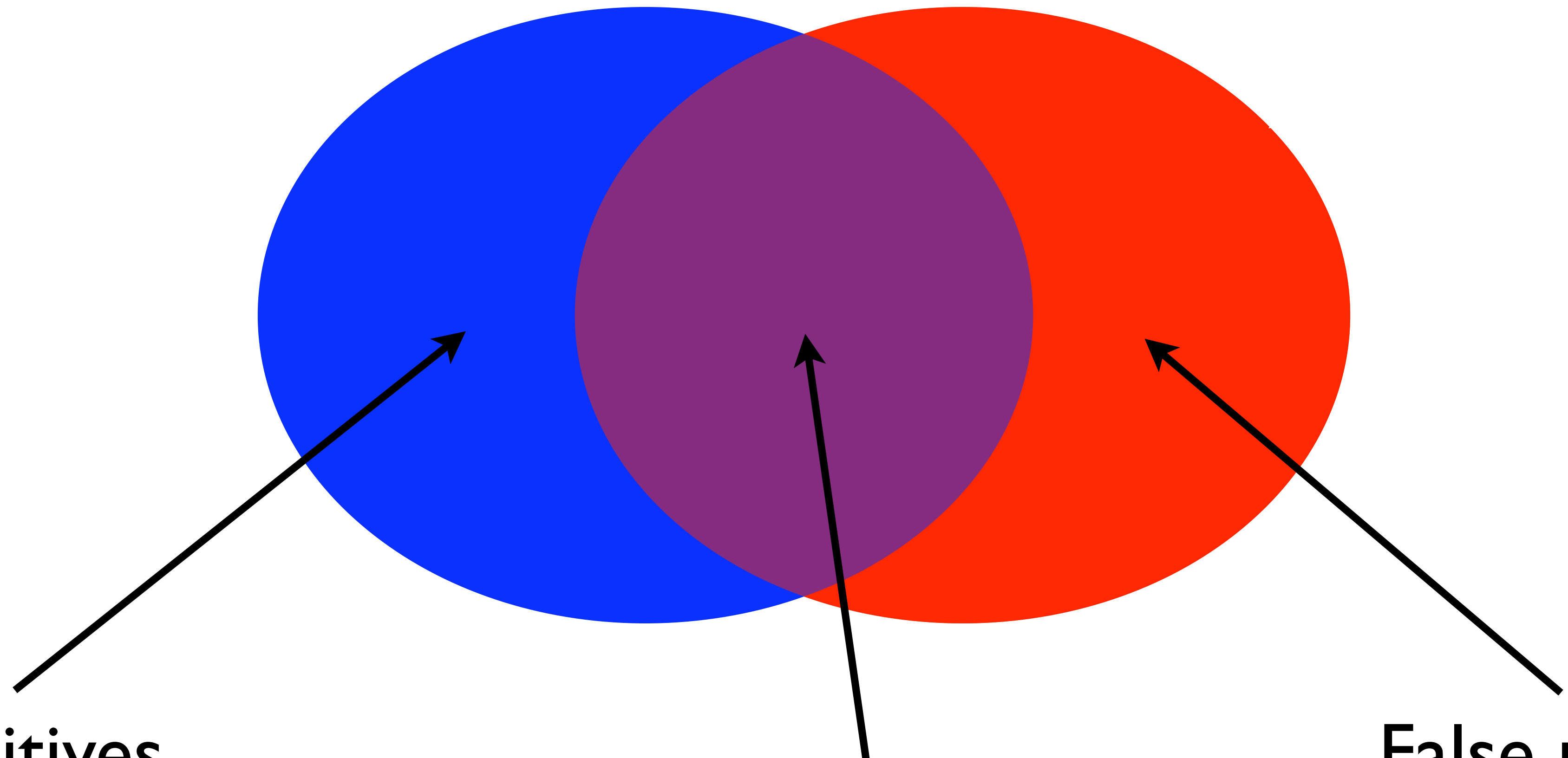


Kernel trick!

Precision and Recall

What a tool finds

What it should find



False positives

Correct predictions

False negatives

High precision = returned elements are relevant

High recall = relevant elements are returned

Precision and Recall

		Actually has property: yes	Actually has property: no	
Predicted to have property	yes	True Positive (TP)	False Positive (FP)	Precision
	no	False Negative (FN)	True Negative (TN)	
		Recall		Accuracy

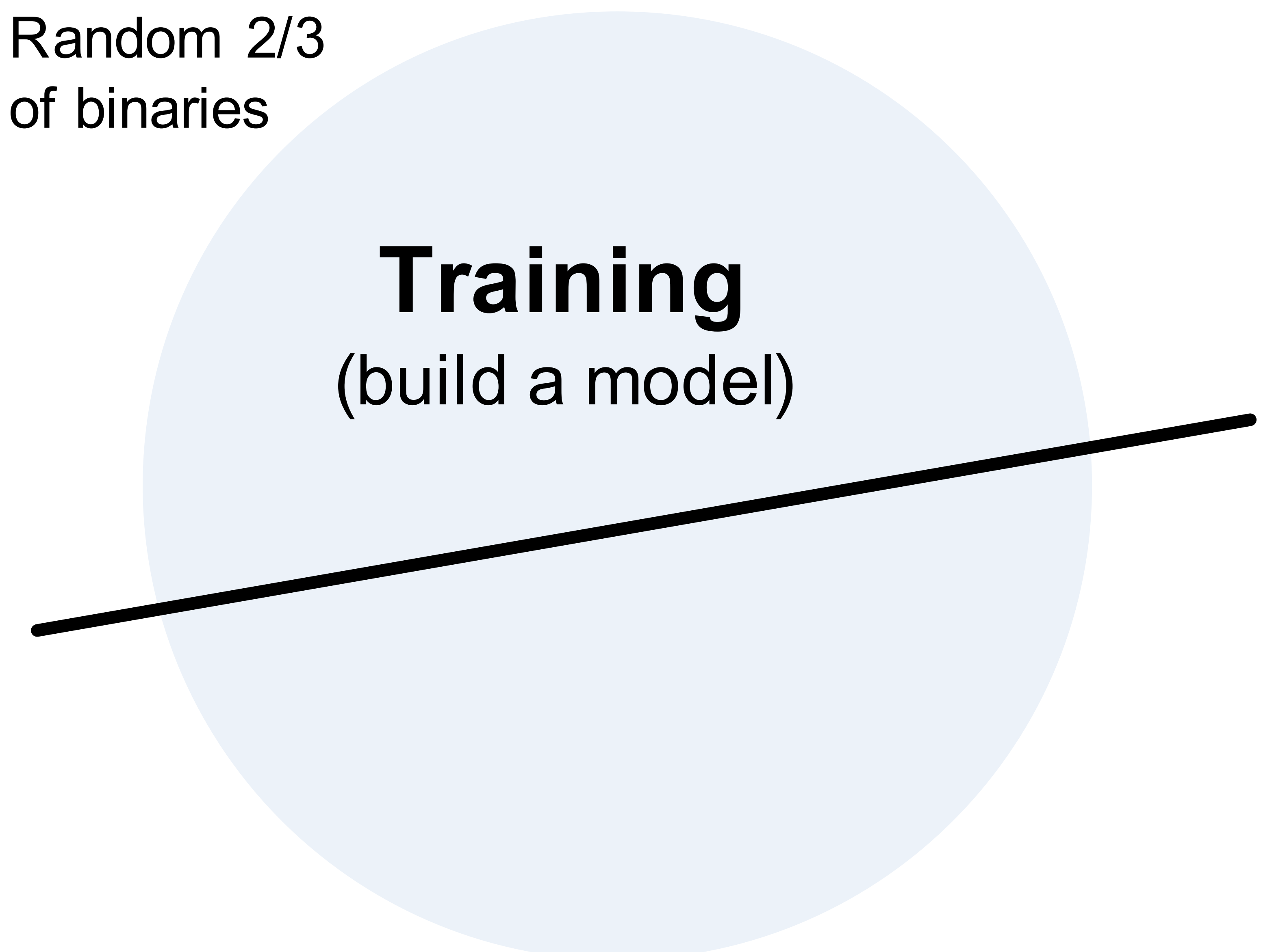
$$\text{Recall} = TP / (TP + FN)$$

$$\text{Precision} = TP / (TP + FP)$$

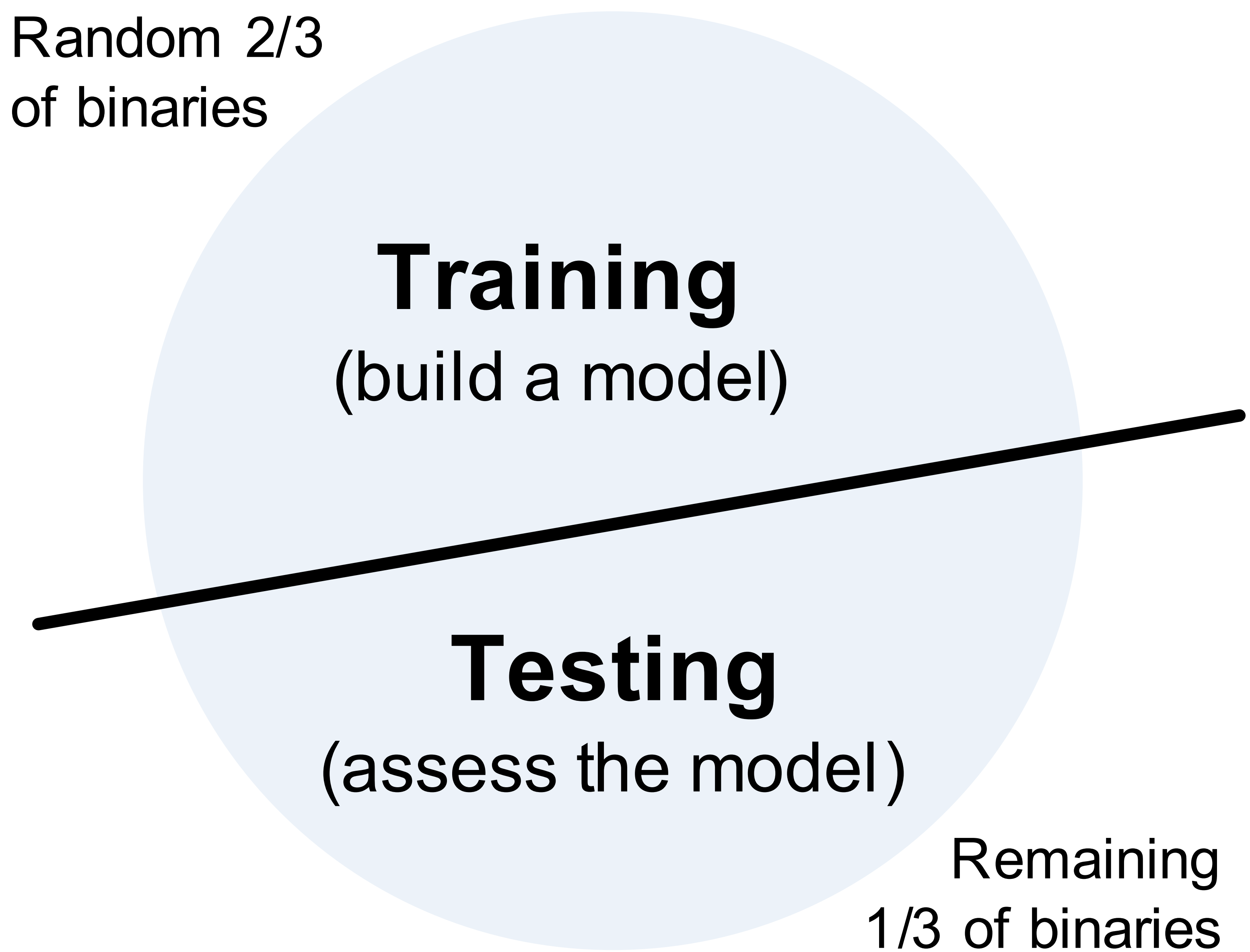
$$\text{Accuracy} = (TP + TN) / (TP + FP + FN + TN)$$

Random splits

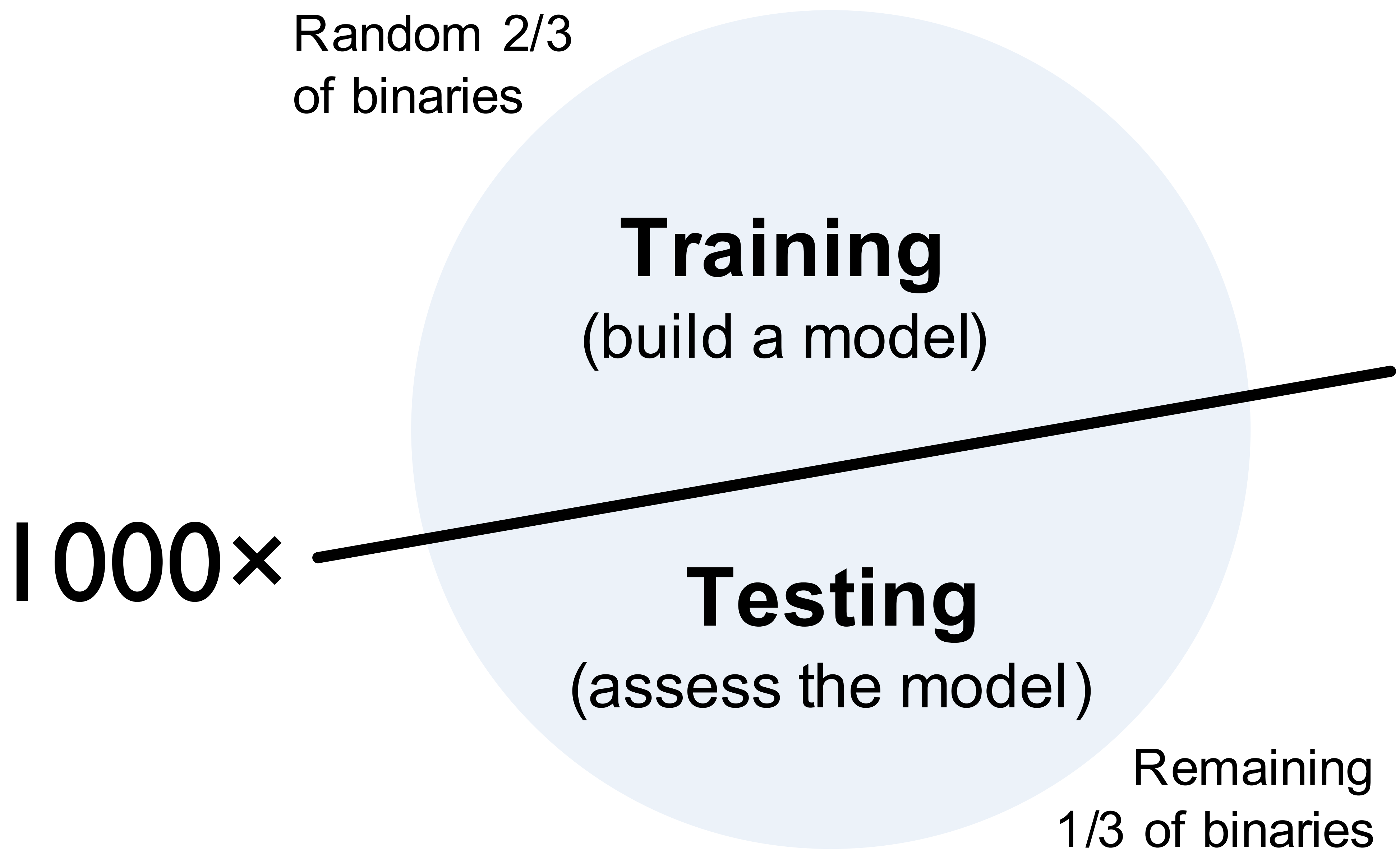
Random splits



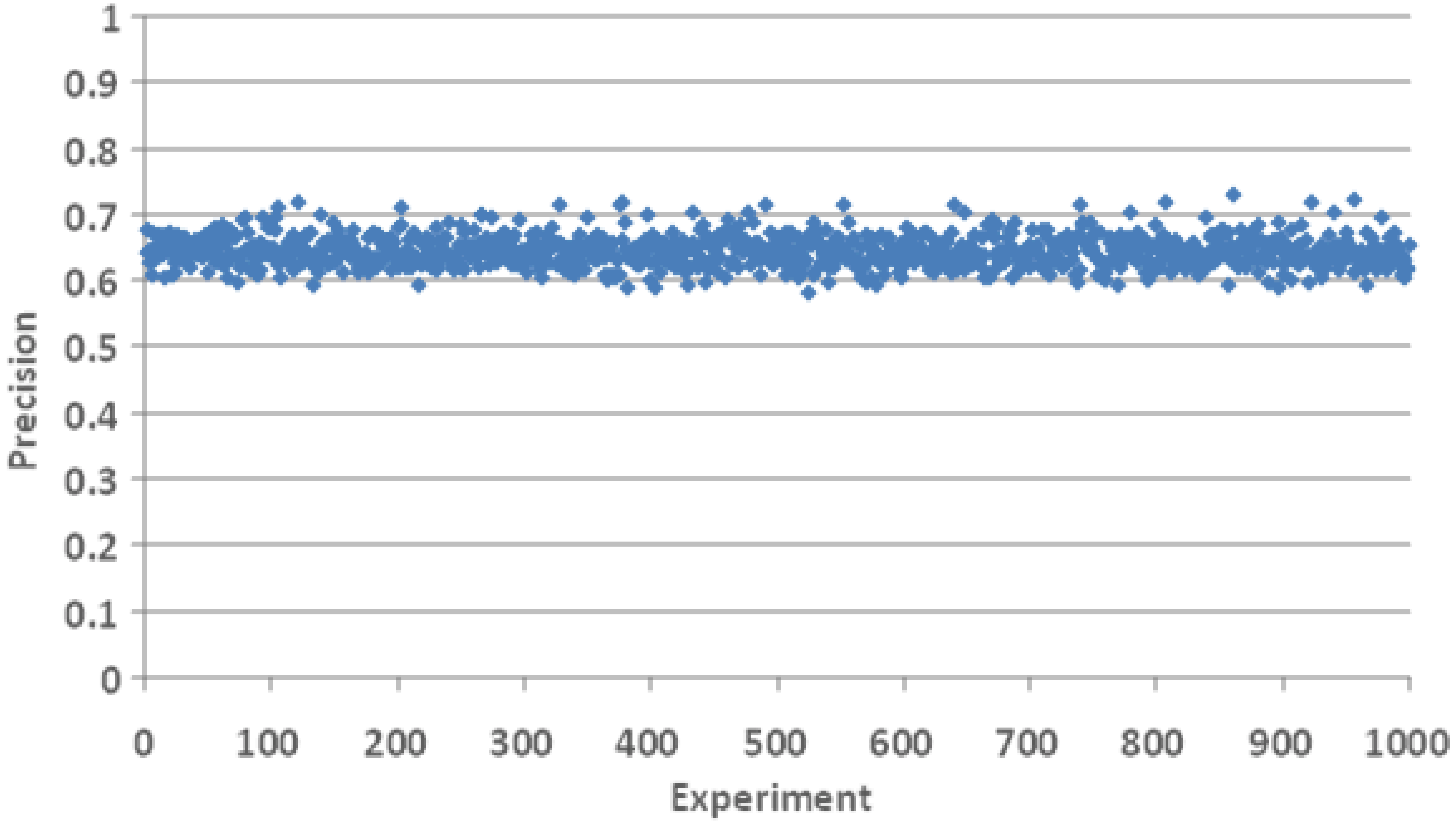
Random splits



Random splits

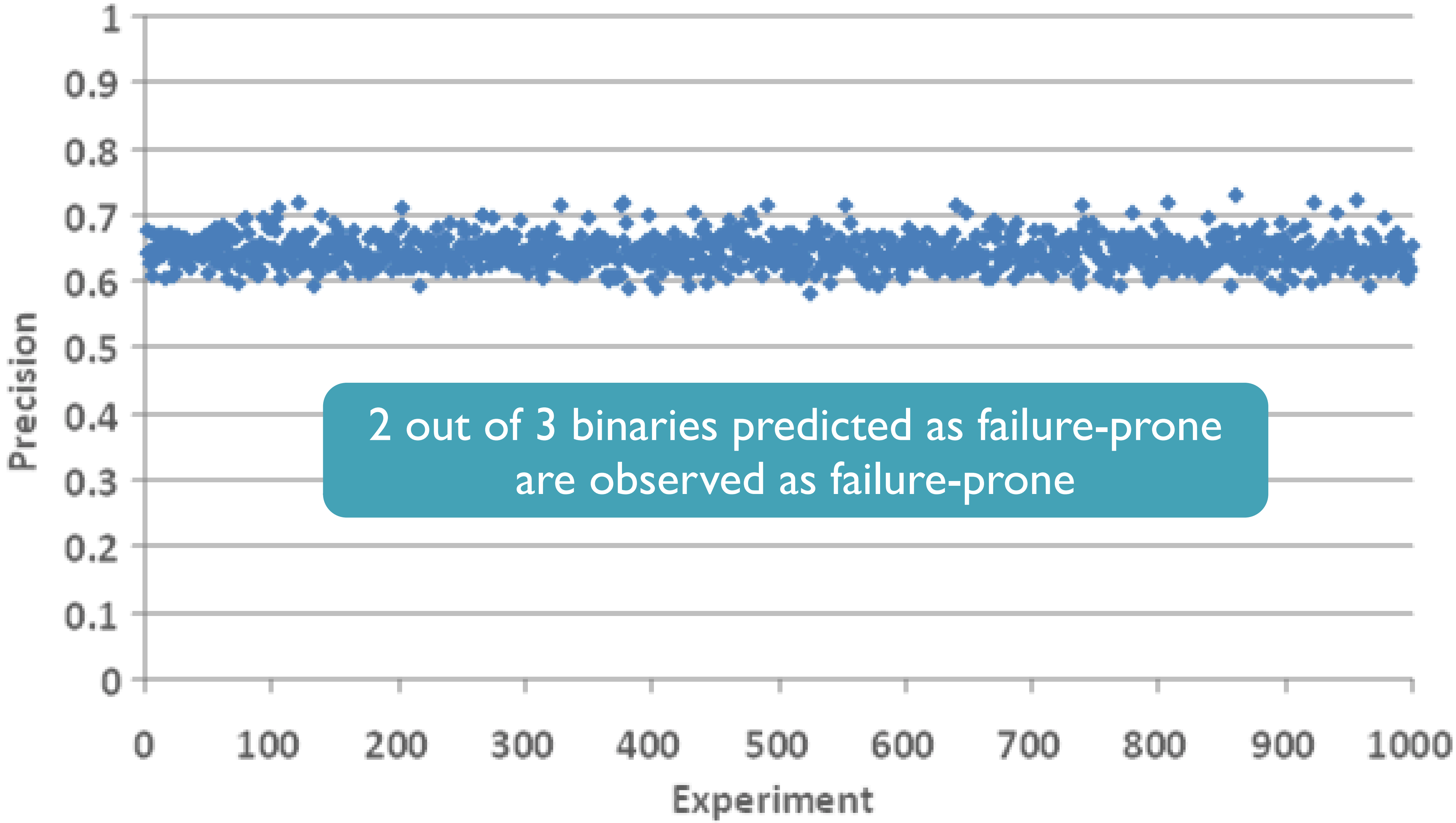


Precision



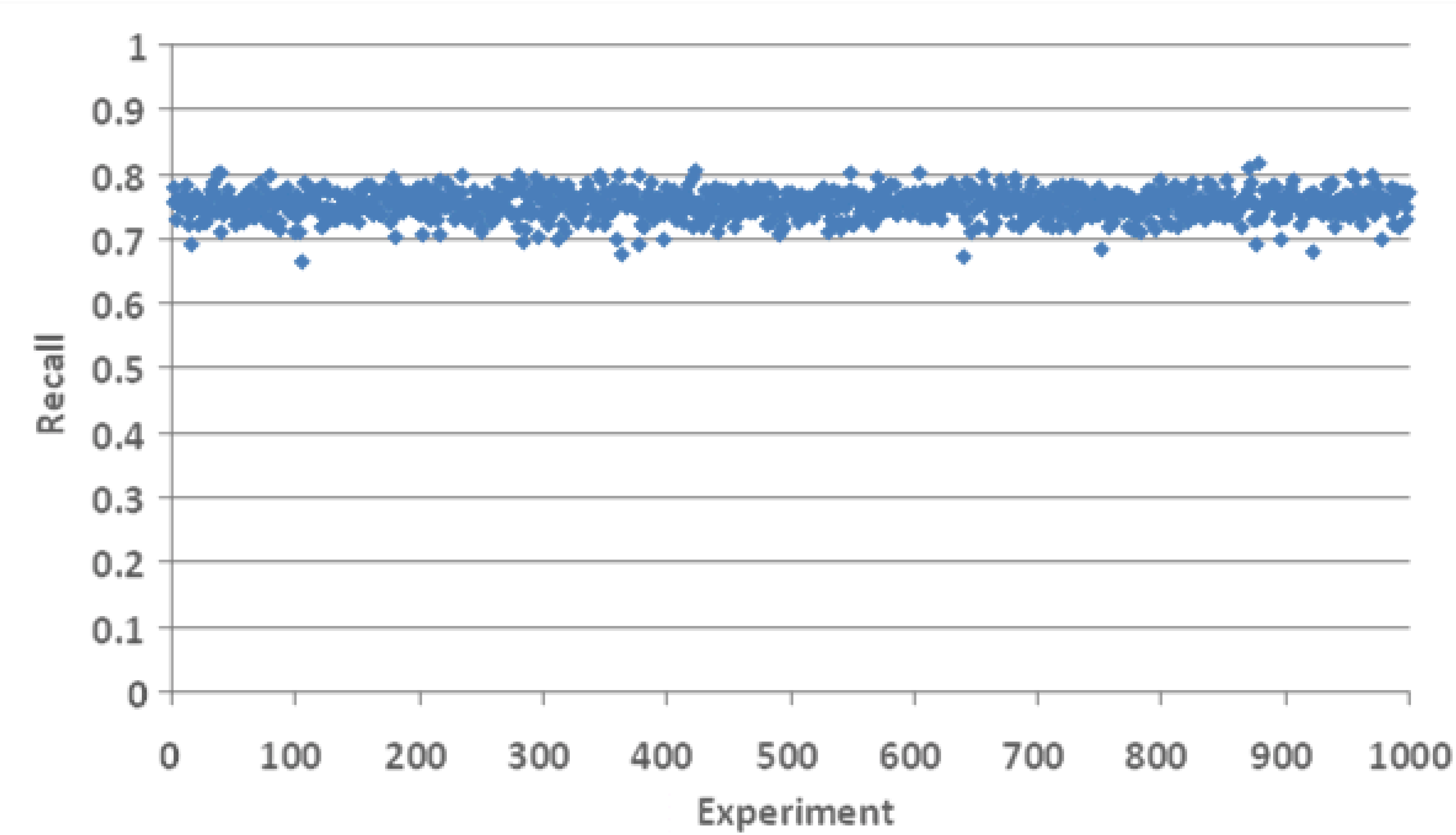
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Precision
0
100
200
300
400
500
600
700
800
900
1000
Experiment

Precision



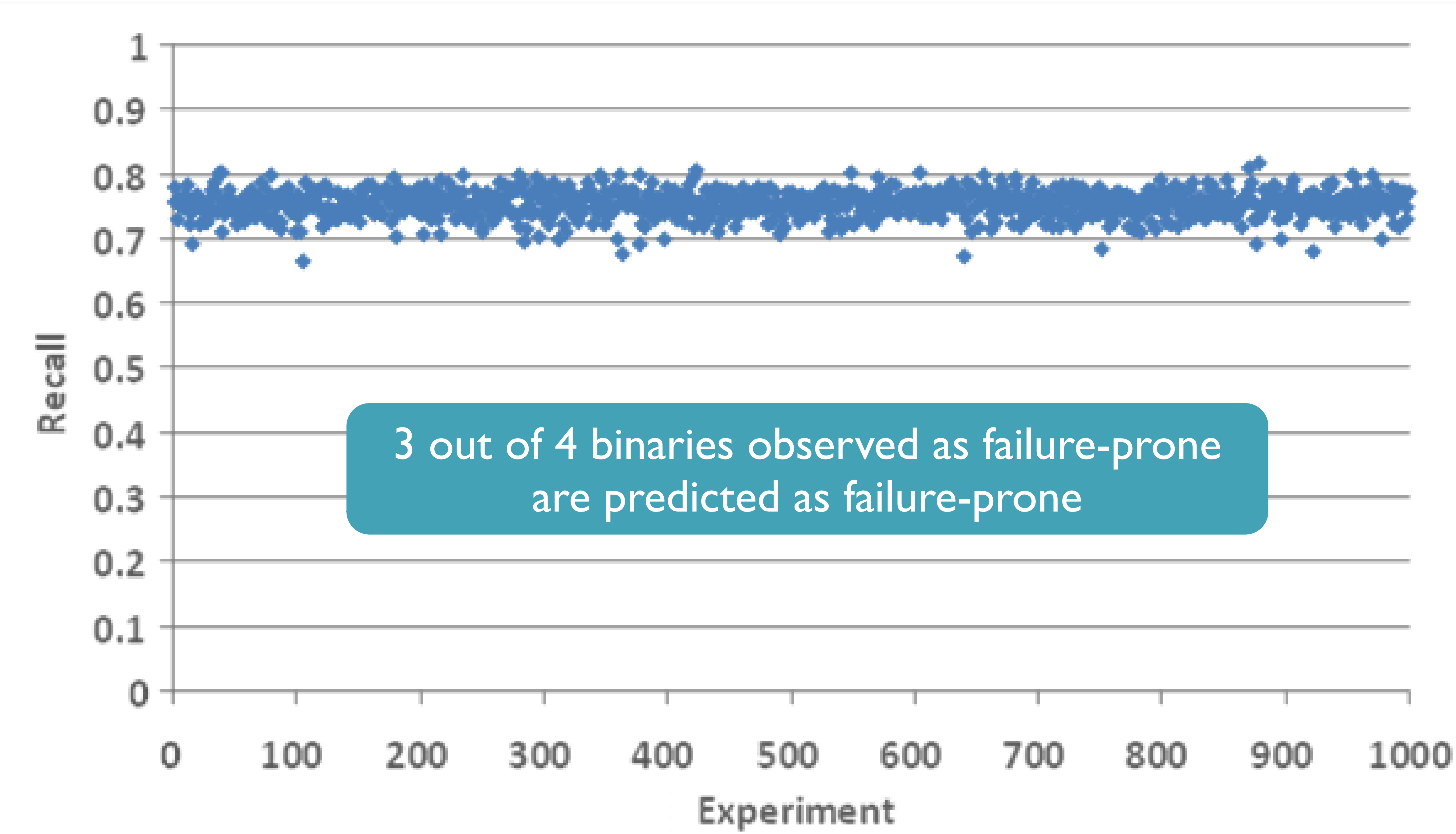
Precision
Experiment
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
200
300
400
500
600
700
800
900
1000
2 out of 3 binaries predicted as failure-prone
are observed as failure-prone

Recall



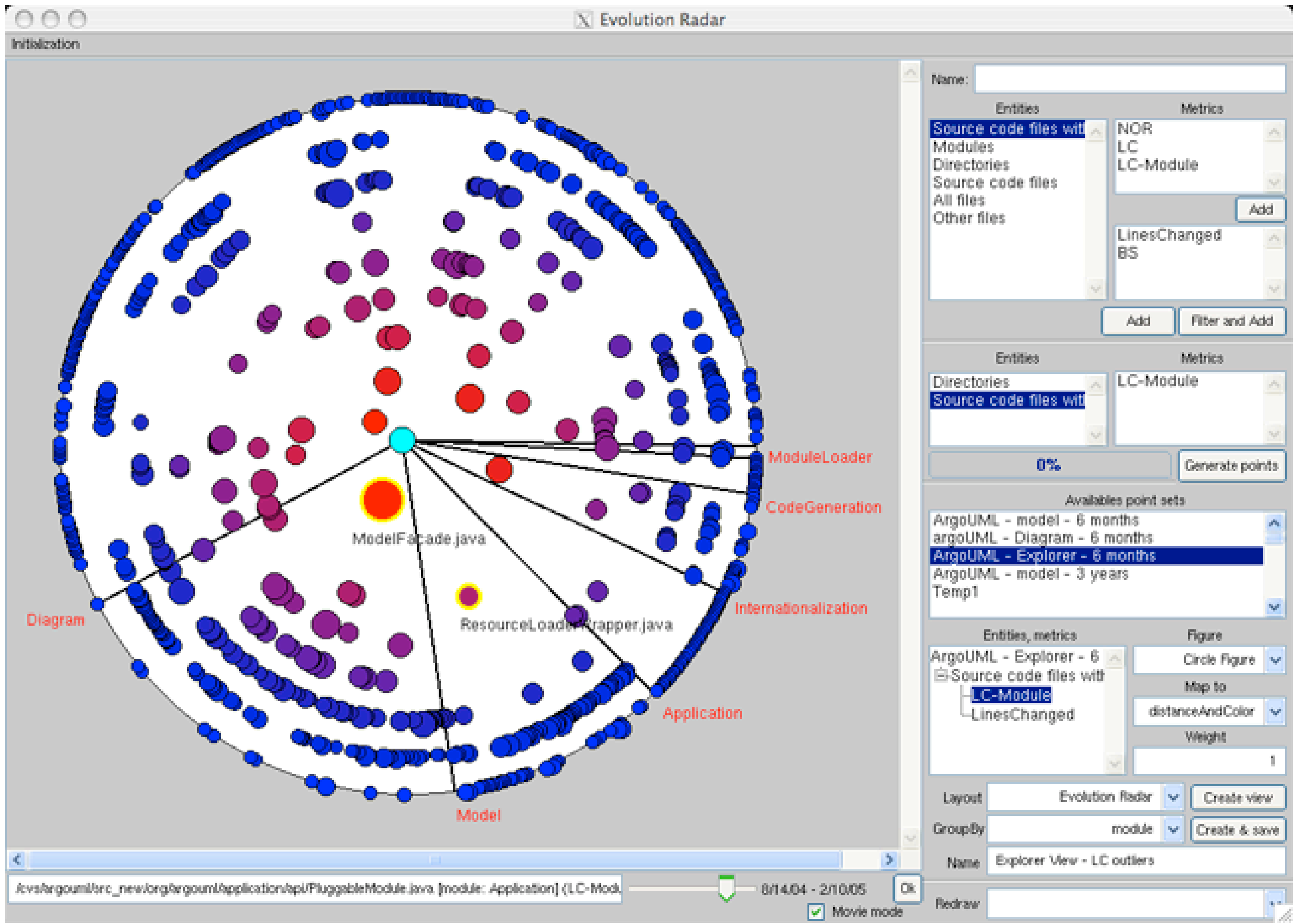
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Recall
0
100
200
300
400
500
600
700
800
900
1000
Experiment

Recall



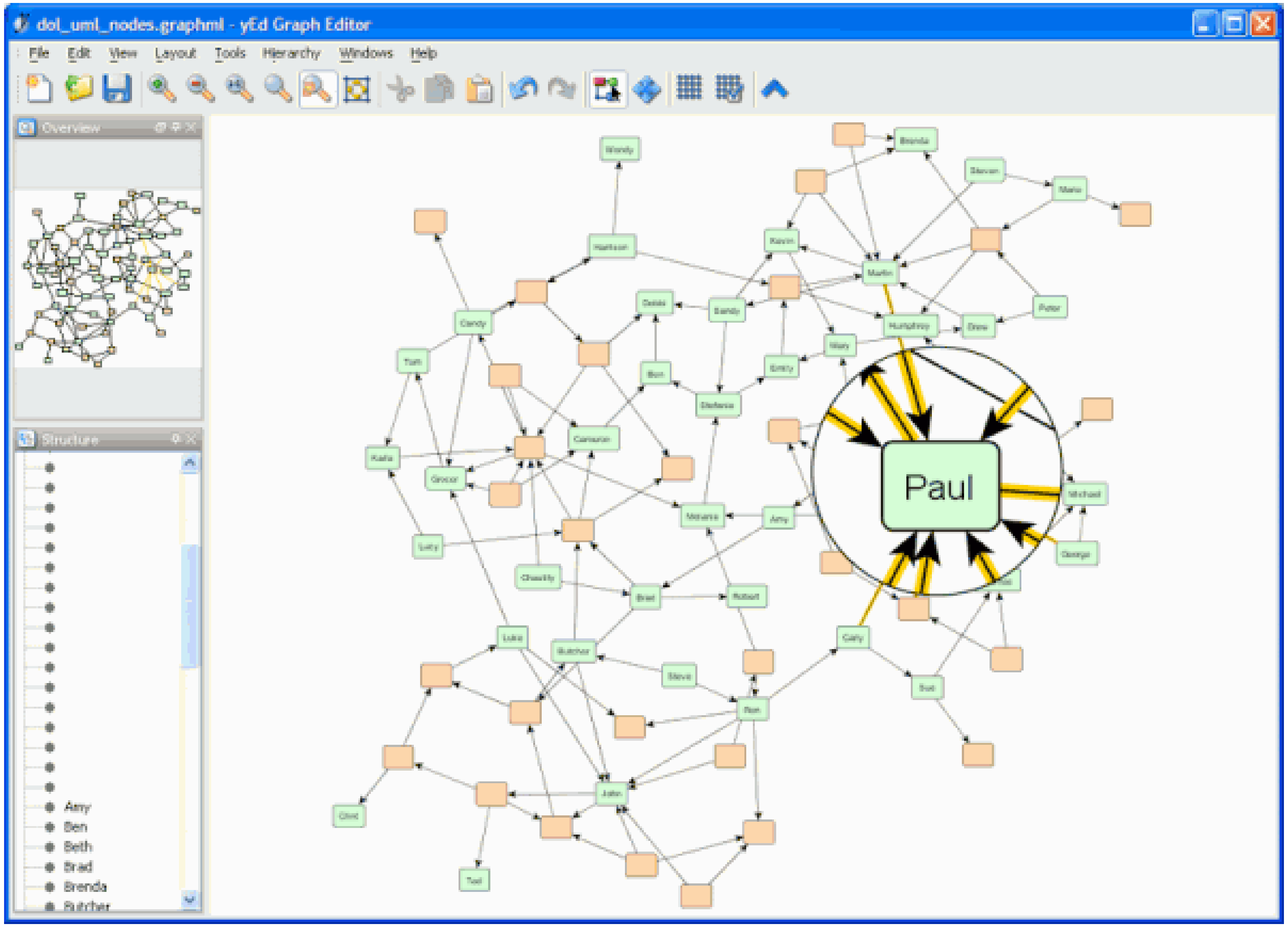
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Recall
0
100
200
300
400
500
600
700
800
900
1000
Experiment
3 out of 4 binaries observed as failure-prone
are predicted as failure-prone

Visualization



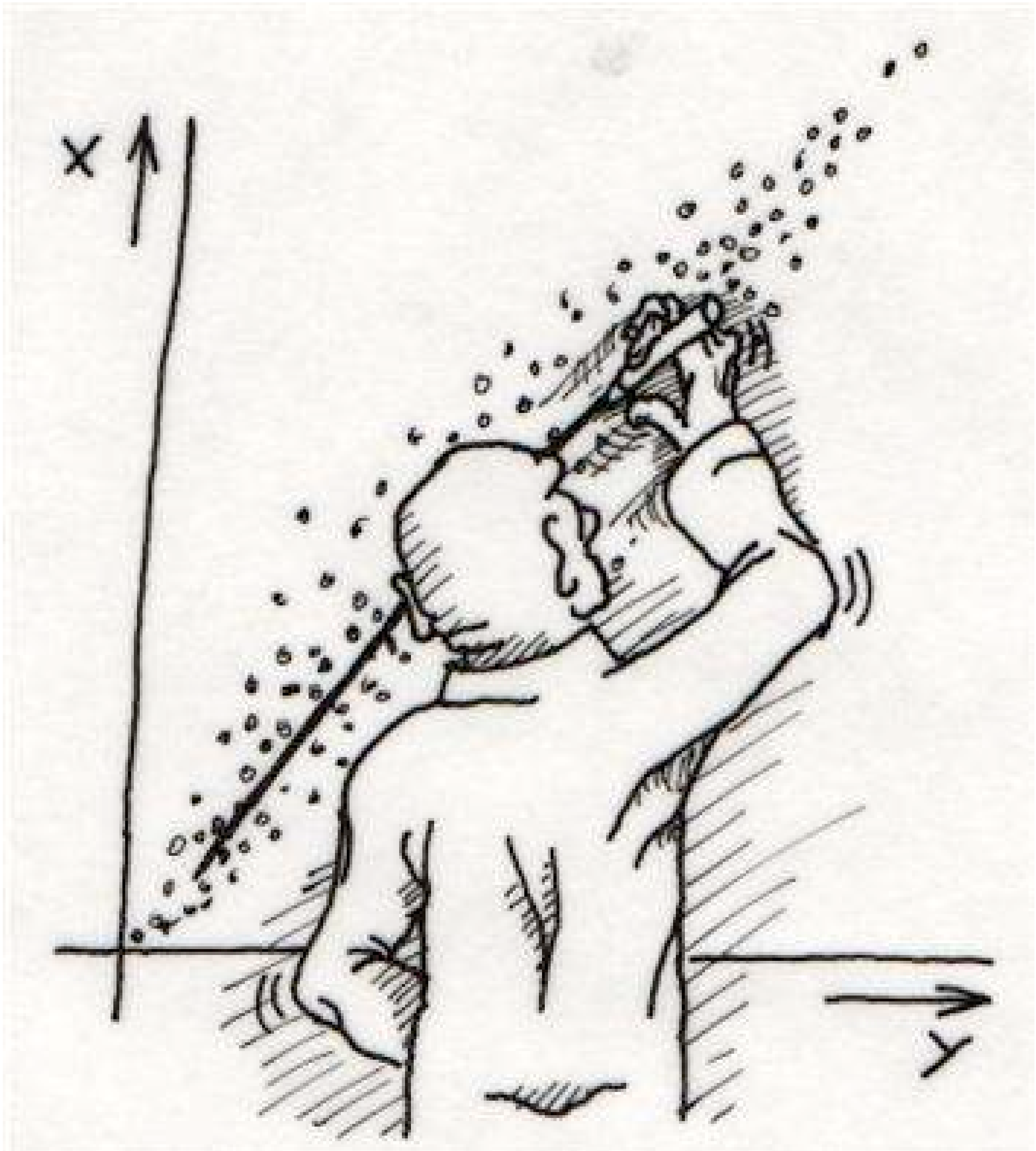
http://www.inf.unisi.ch/phd/dambros/tools/

yEd - Java Graph Editor



http://www.yworks.com/en/products_yed_about.htm

Regression Analysis



k-Nearest Neighbour

What describes the two?

k-Nearest Neighbour

What describes the two?

taste

colour

size

smell

shape

texture

k-Nearest Neighbour

So what is this one?

k-Nearest Neighbour

So what is this one?

k-Nearest Neighbour

So what is this one?

- taste
- colour
- size
- smell
- shape
- texture

k-Nearest Neighbour

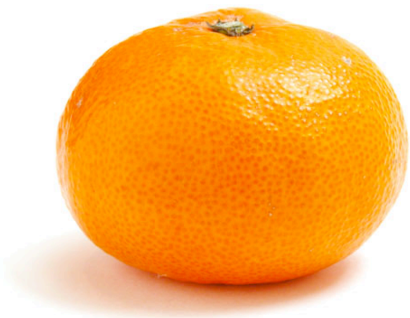
So what is this one?

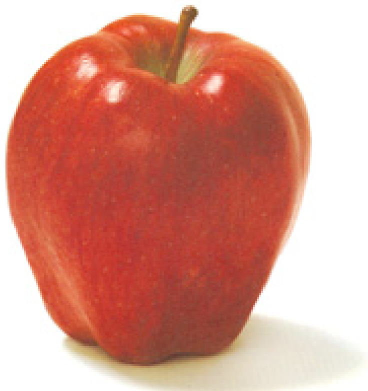
k-Nearest Neighbour

Similar problems have similar solutions...

- The car mechanic can infer that if the ignition doesn’t turn on, there might be a problem with the spark plug.
- Court case A is similar to my case B. Can I reuse it’s solution to win mine?
- Code Reuse ;-)

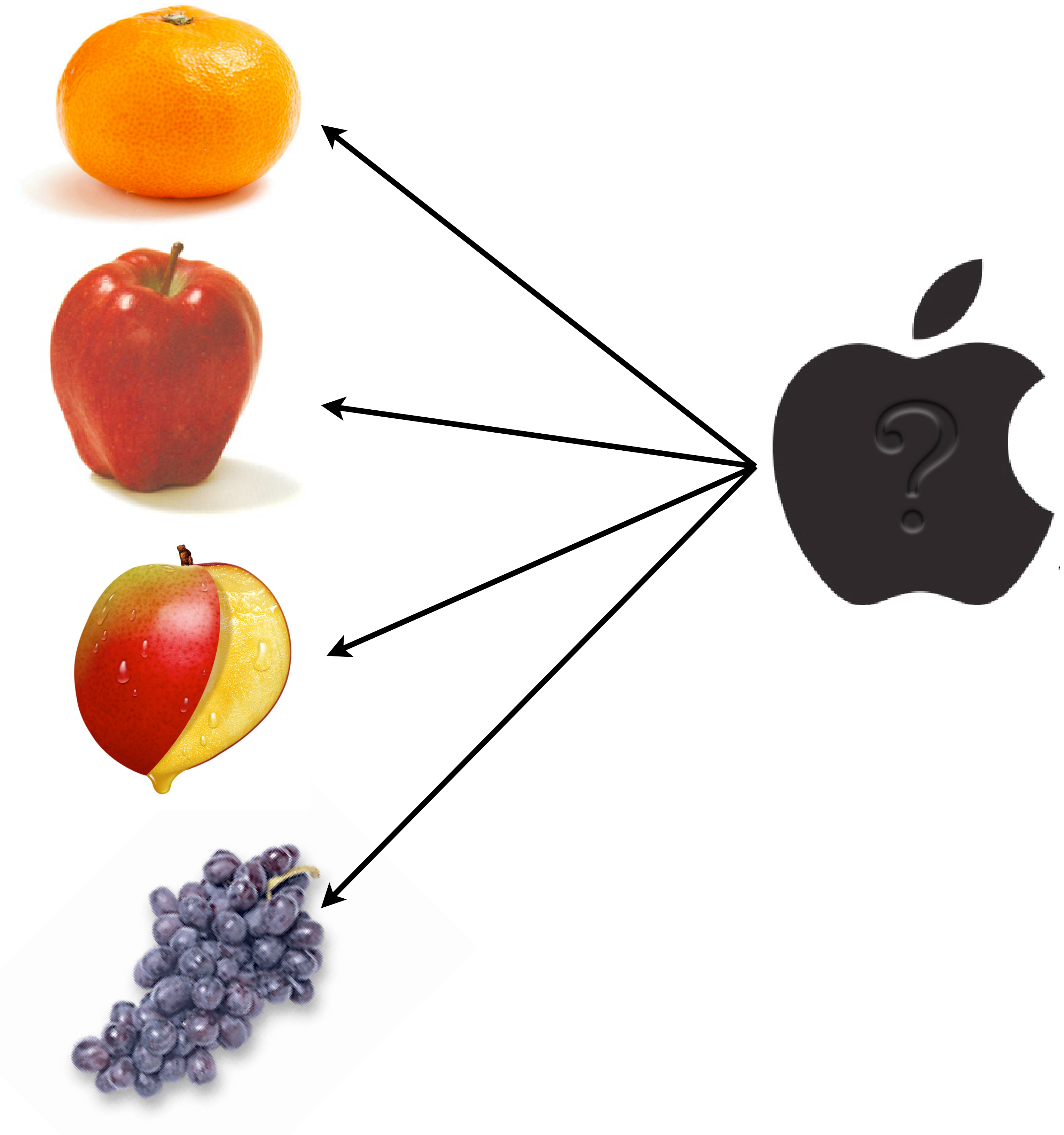
k-Nearest Neighbour



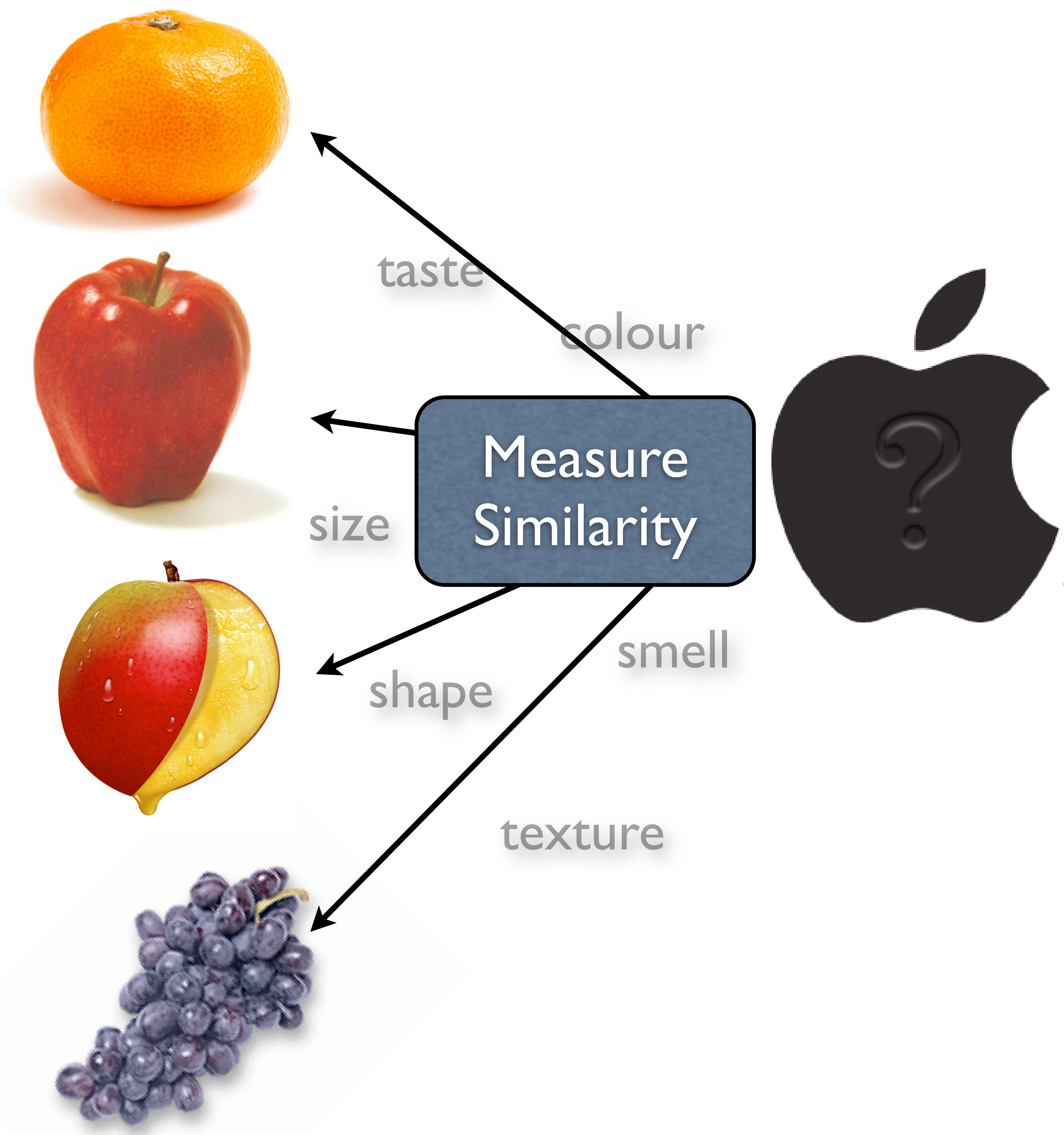


k-Nearest Neighbour

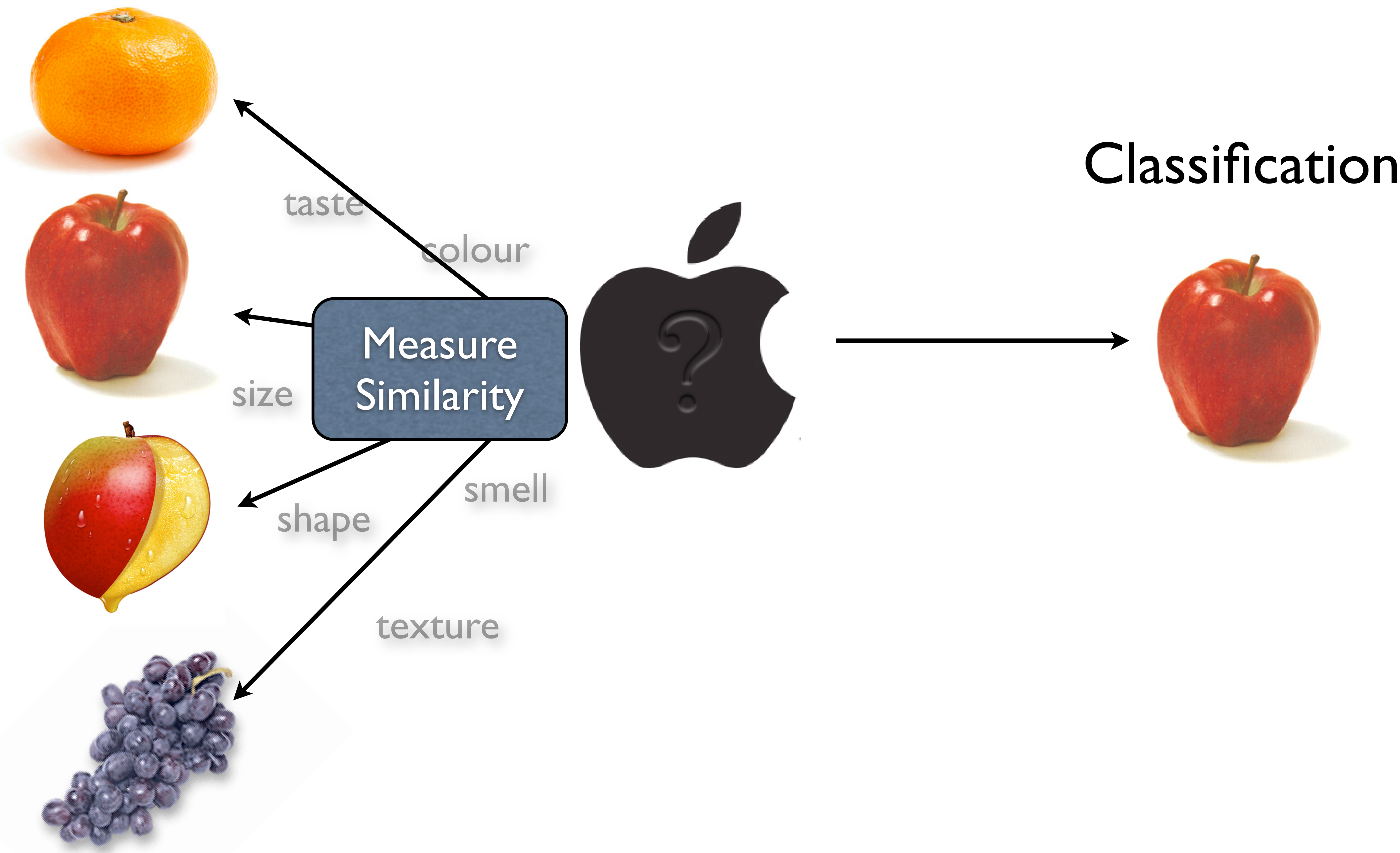
k-Nearest Neighbour



k-Nearest Neighbour



k-Nearest Neighbour



k-Nearest Neighbour

What about prediction problems?

Recall the Desharnais data set from the Exercises...

k-Nearest Neighbour

What about prediction problems?

Recall the Desharnais data set from the Exercises...

Project A

Project B

Project C

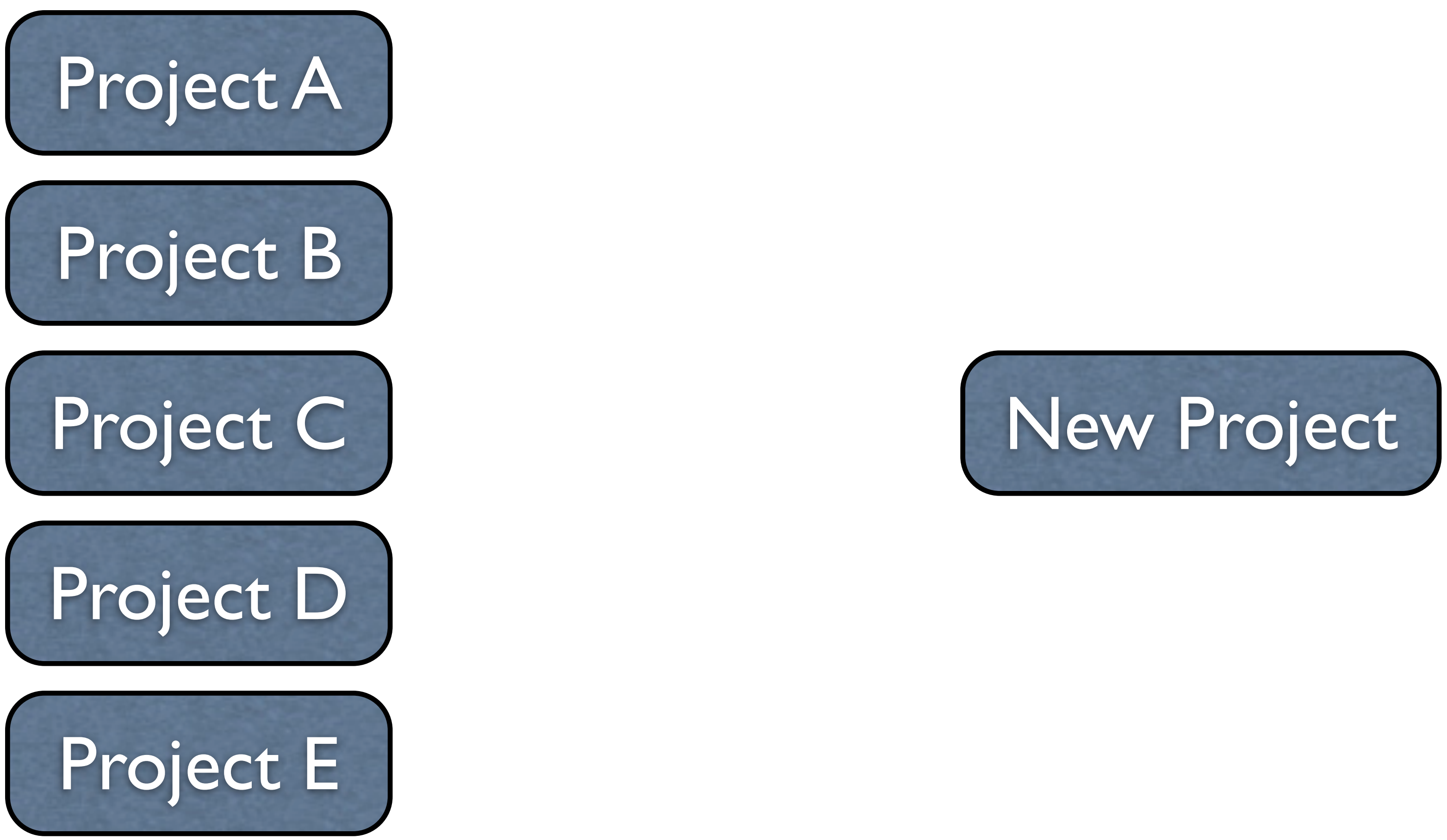
Project D

Project E

k-Nearest Neighbour

What about prediction problems?

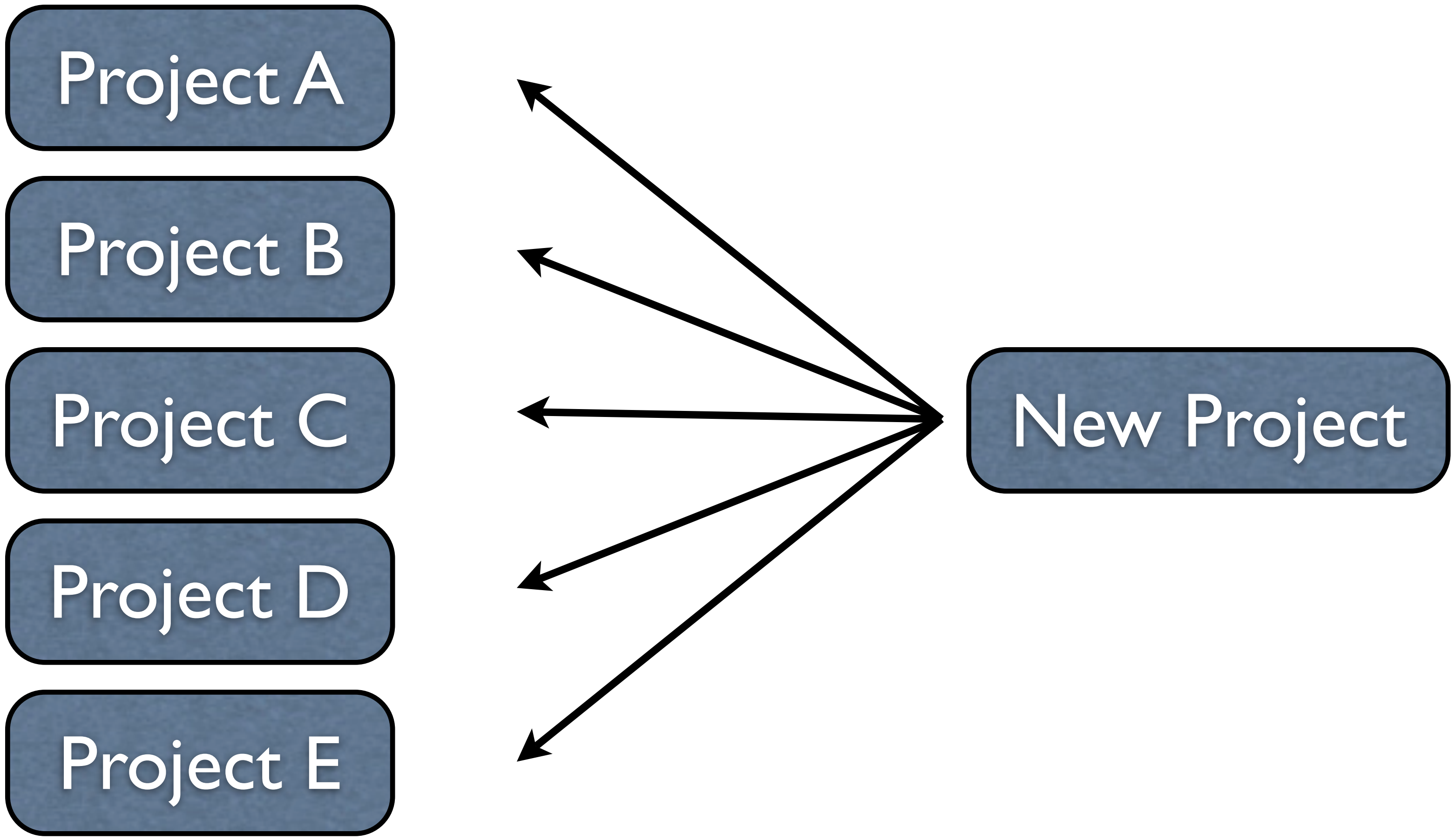
Recall the Desharnais data set from the Exercises...



k-Nearest Neighbour

What about prediction problems?

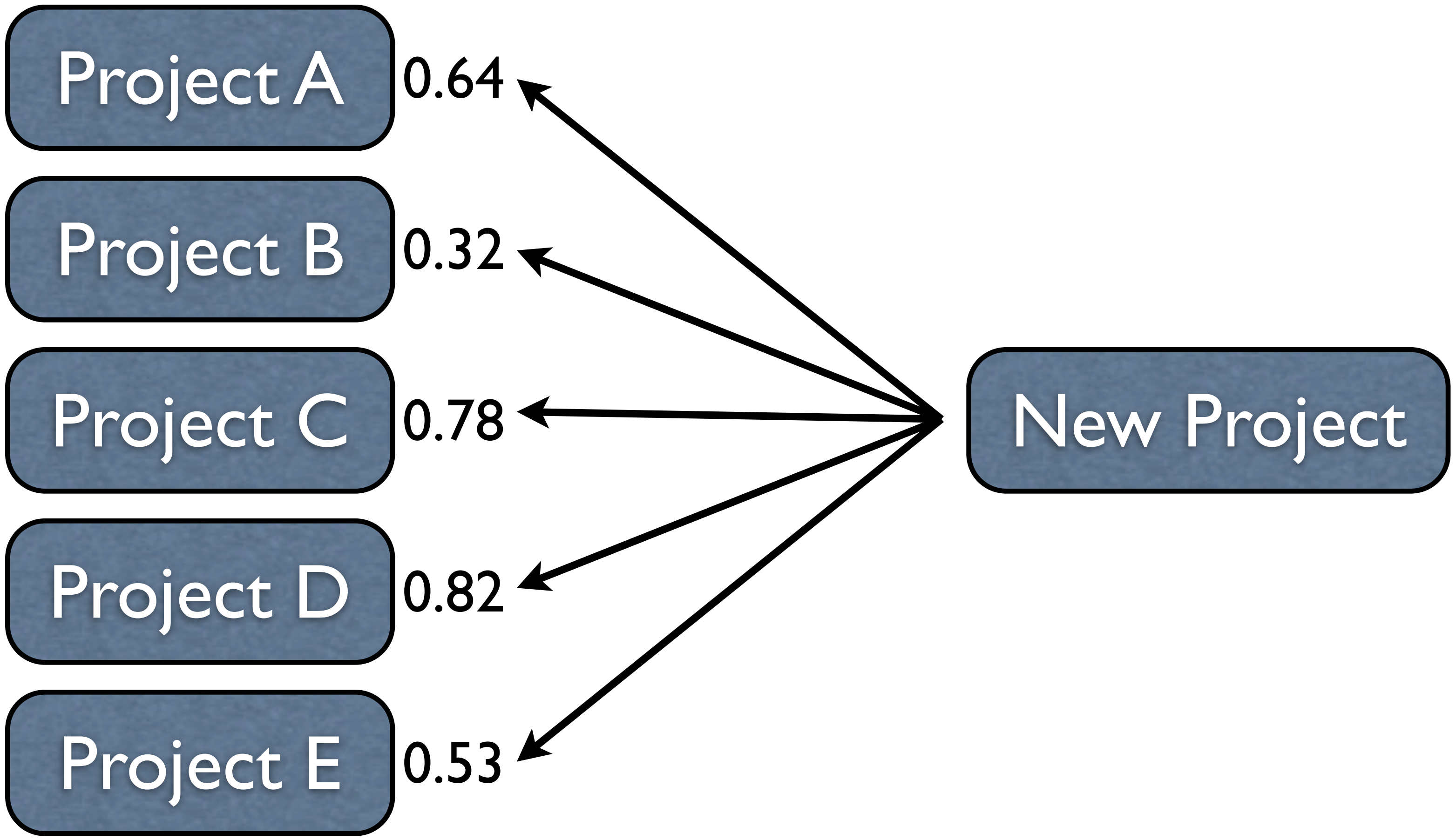
Recall the Desharnais data set from the Exercises...



k-Nearest Neighbour

What about prediction problems?

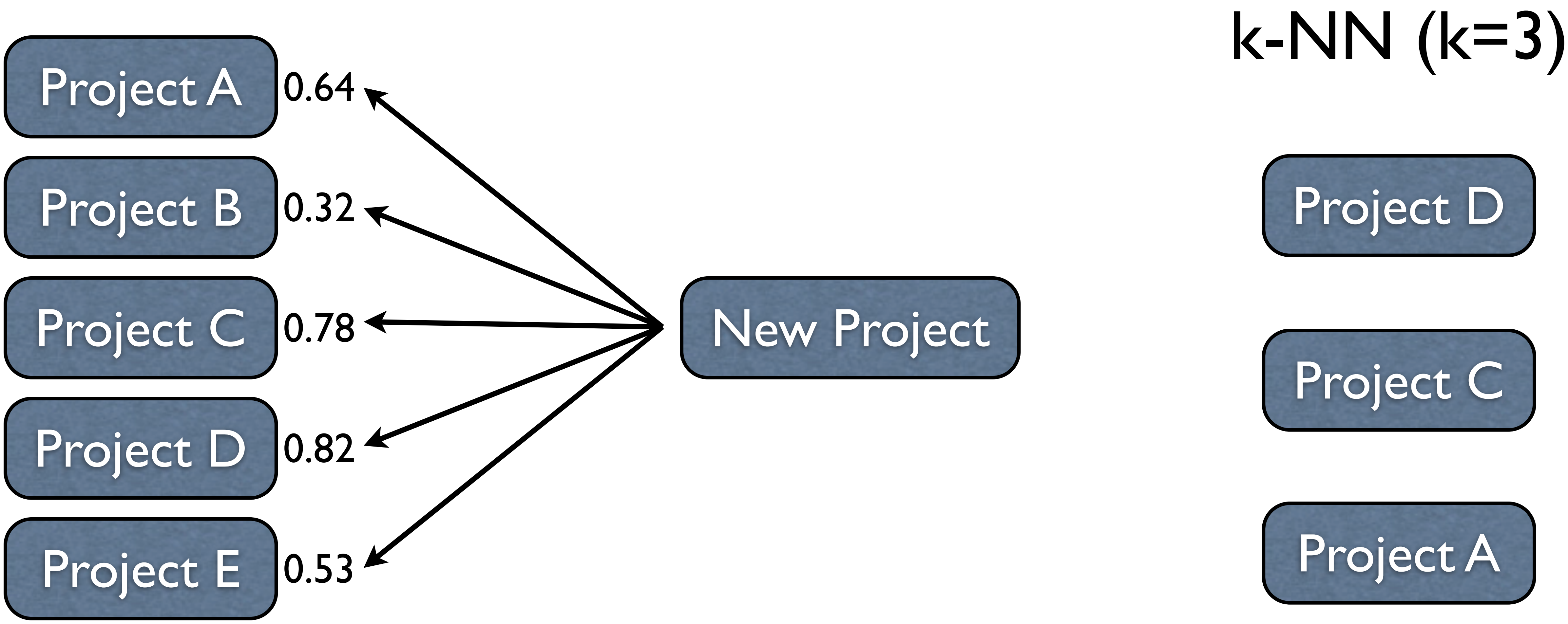
Recall the Desharnais data set from the Exercises...



k-Nearest Neighbour

What about prediction problems?

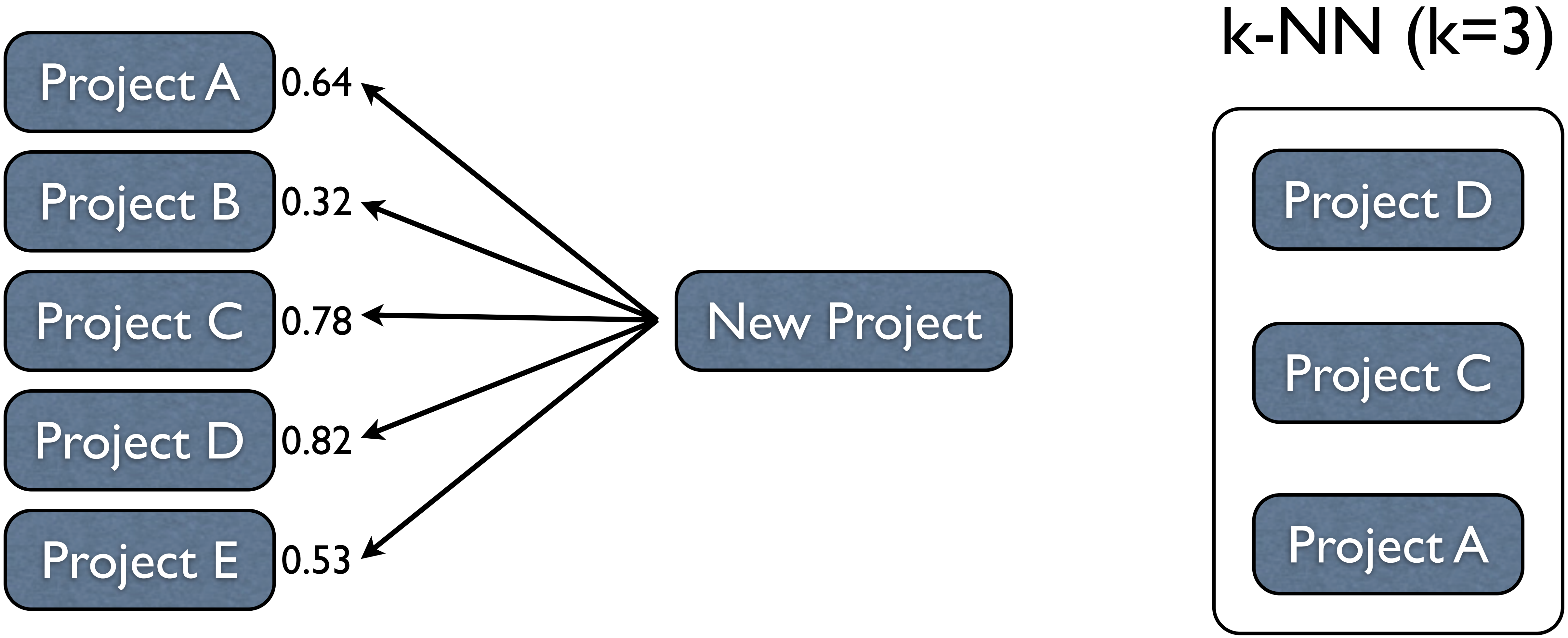
Recall the Desharnais data set from the Exercises...



k-Nearest Neighbour

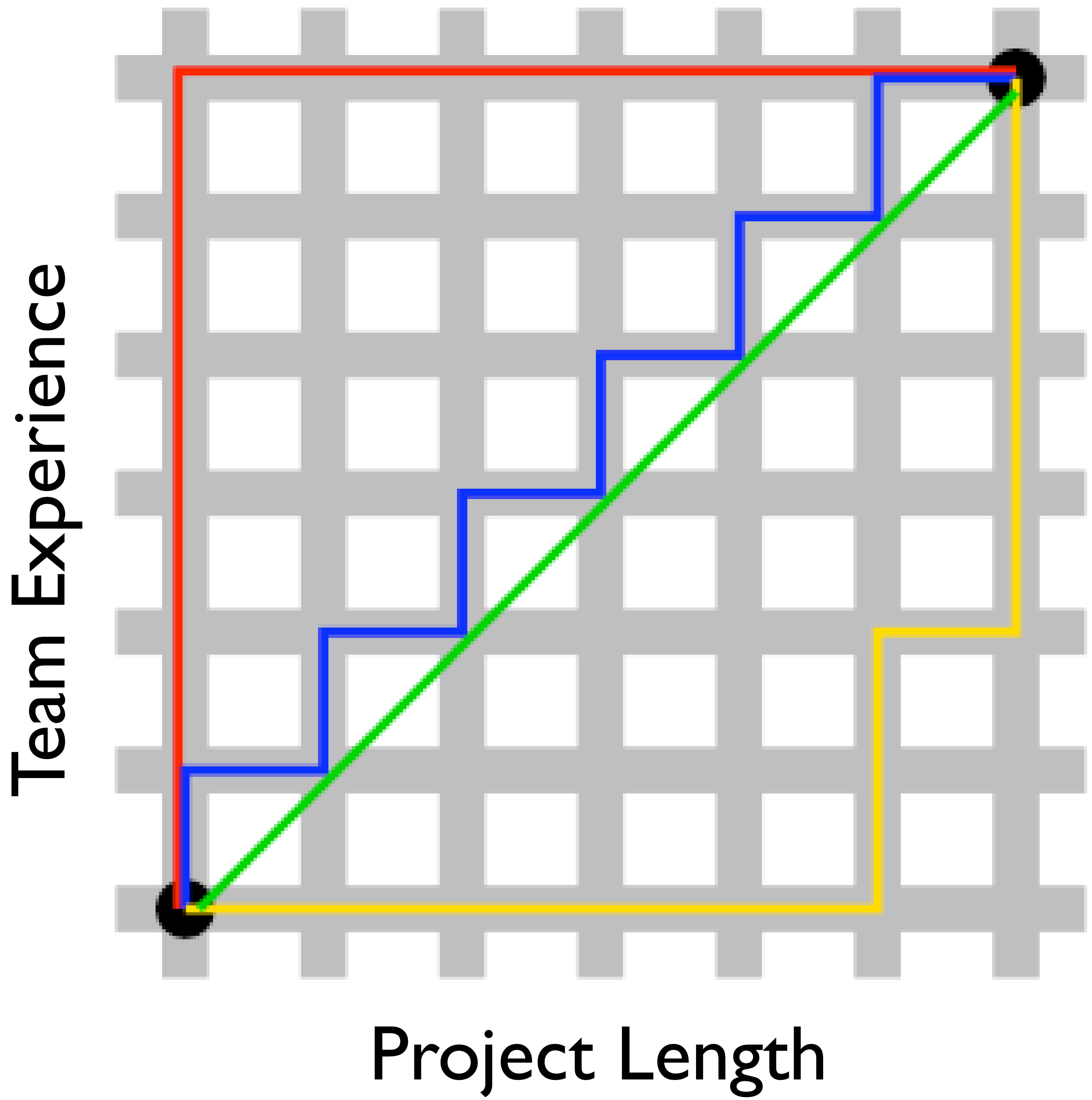
What about prediction problems?

Recall the Desharnais data set from the Exercises...



Average the solution!

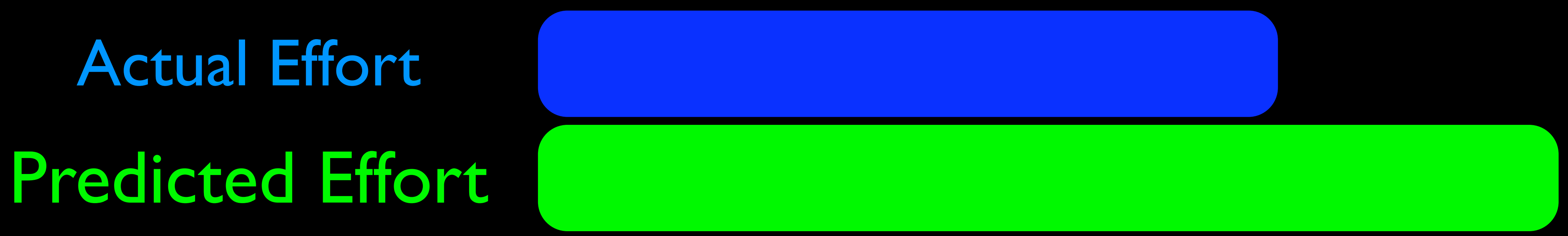
k-Nearest Neighbour



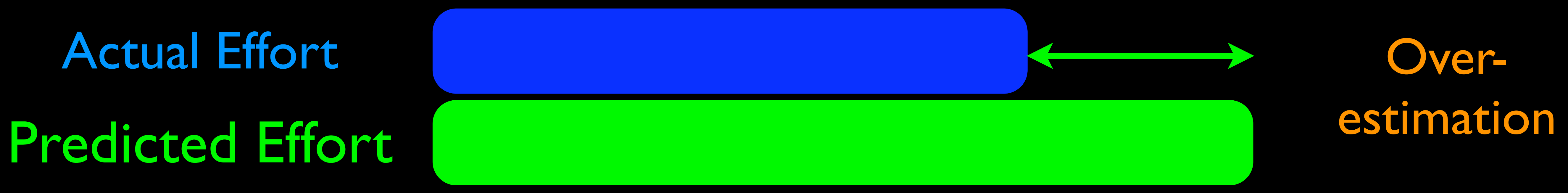
Evaluation of Results

Evaluation of Results

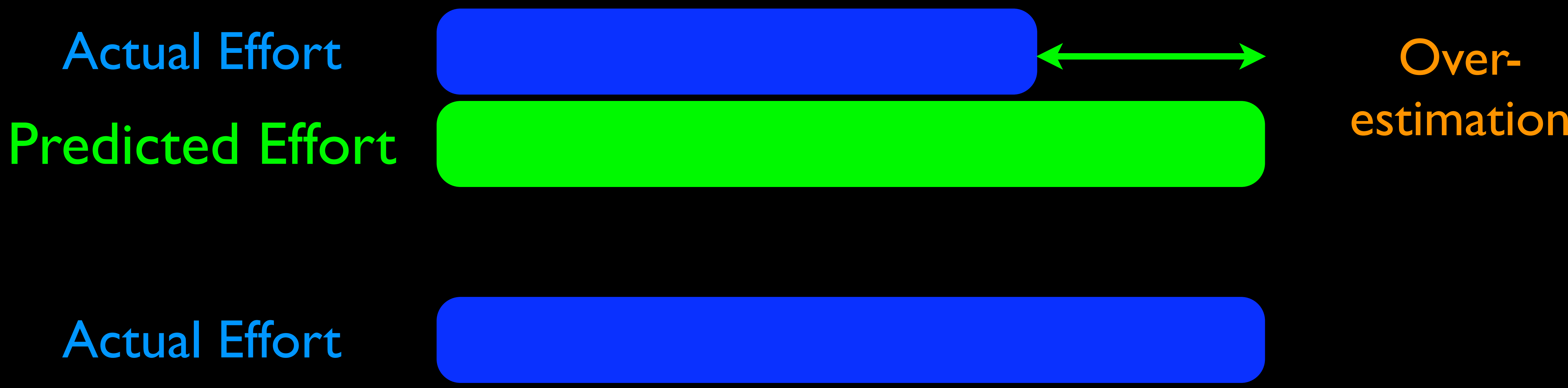
Evaluation of Results



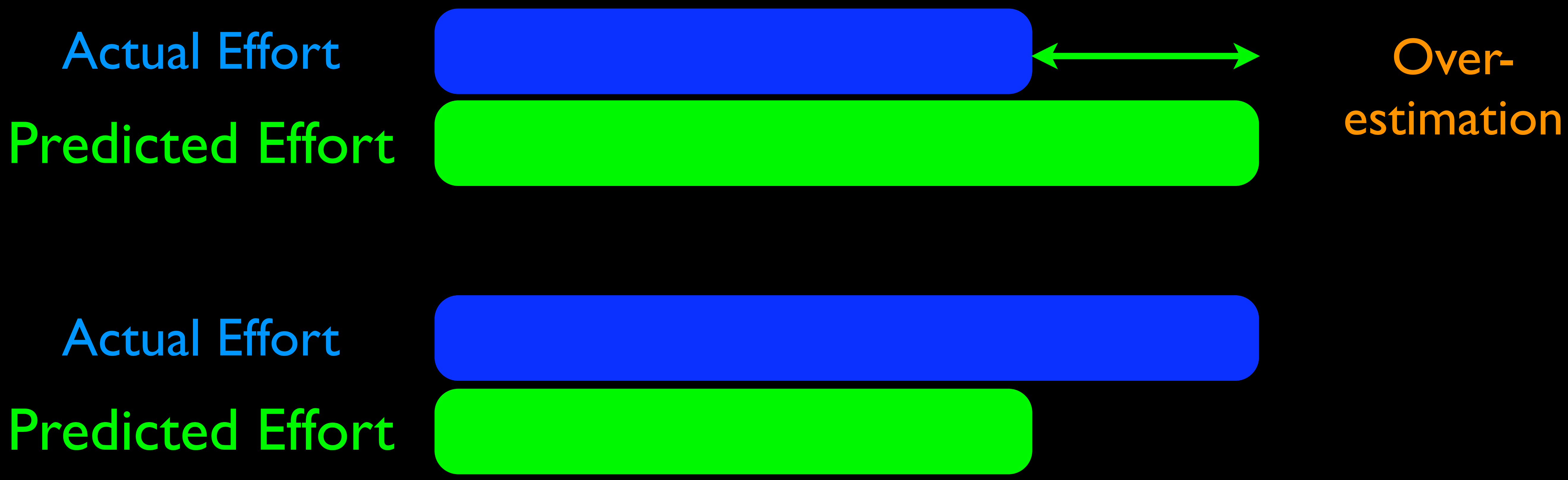
Evaluation of Results



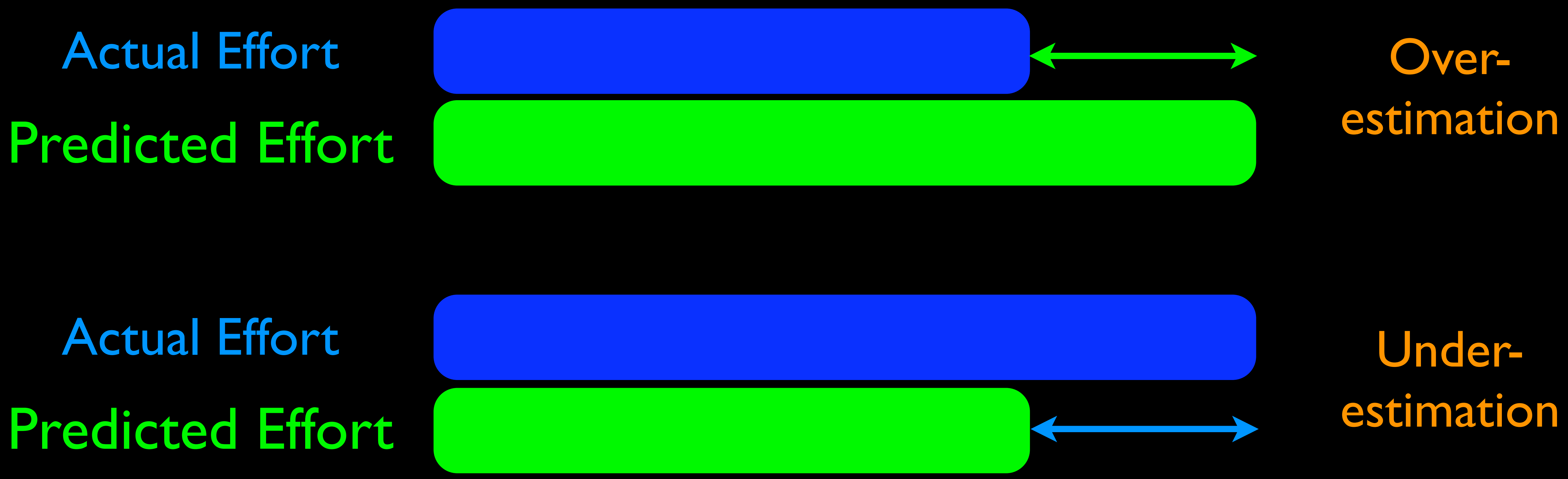
Evaluation of Results



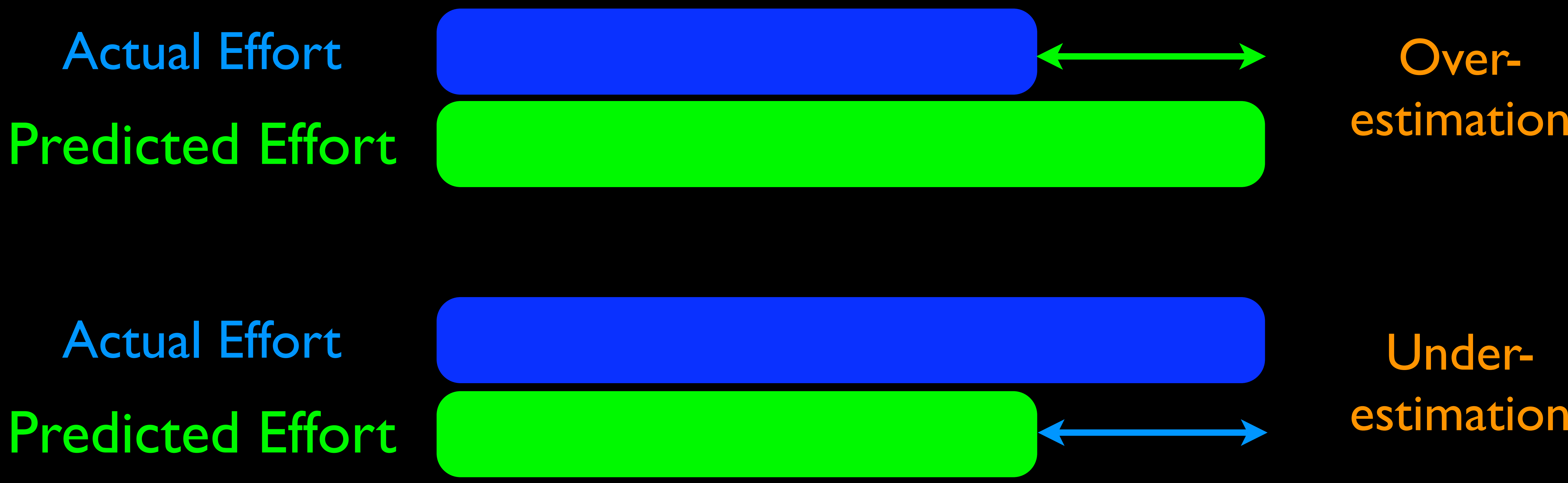
Evaluation of Results



Evaluation of Results

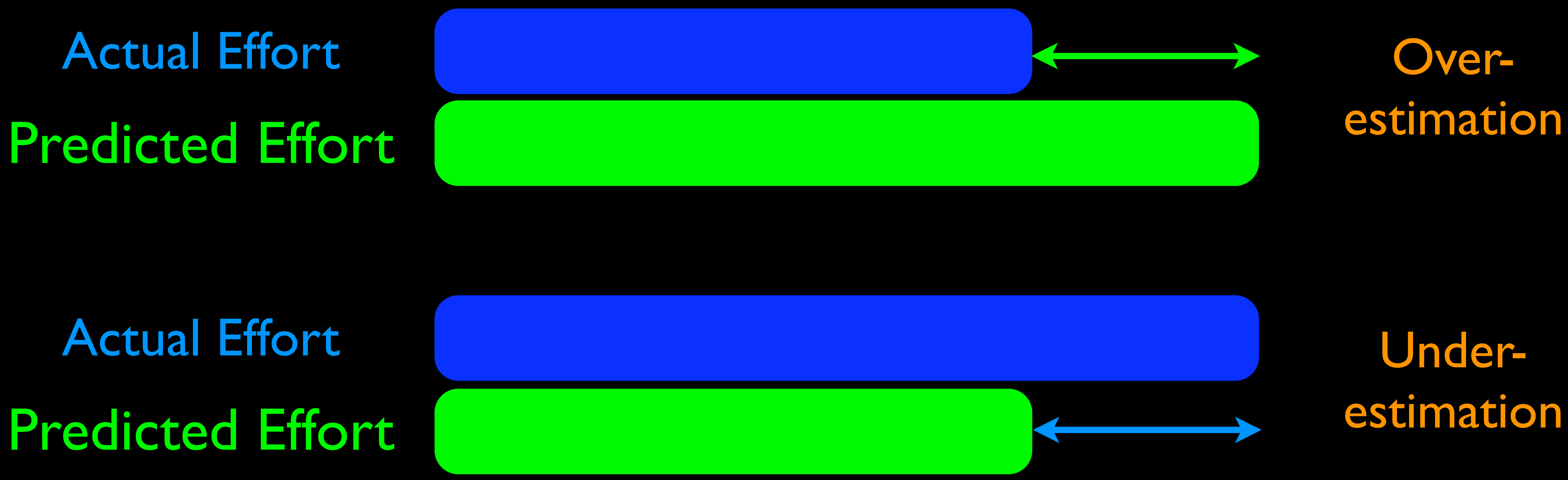


Evaluation of Results

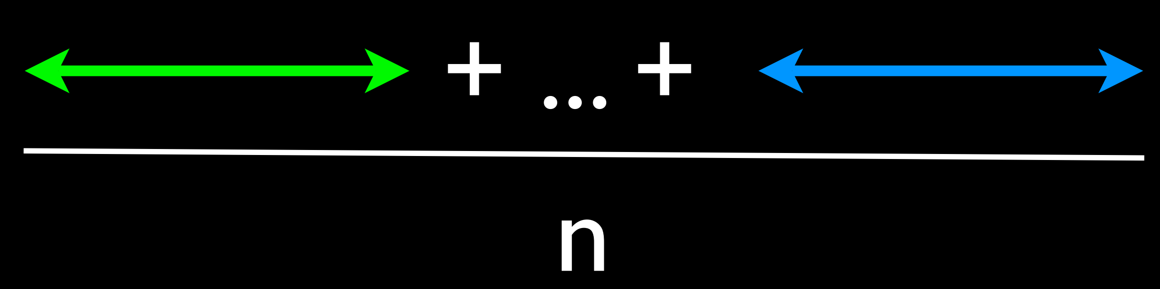


Average of
Absolute Residuals

Evaluation of Results



Average of
Absolute Residuals



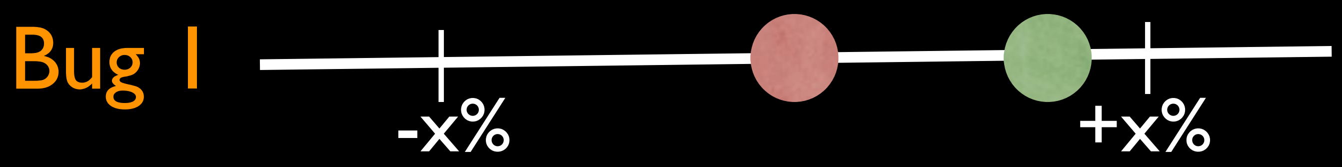
(n = number of predictions made)

Evaluation of Results

Pred(x)
% of predictions that lie within x% of Actual Effort

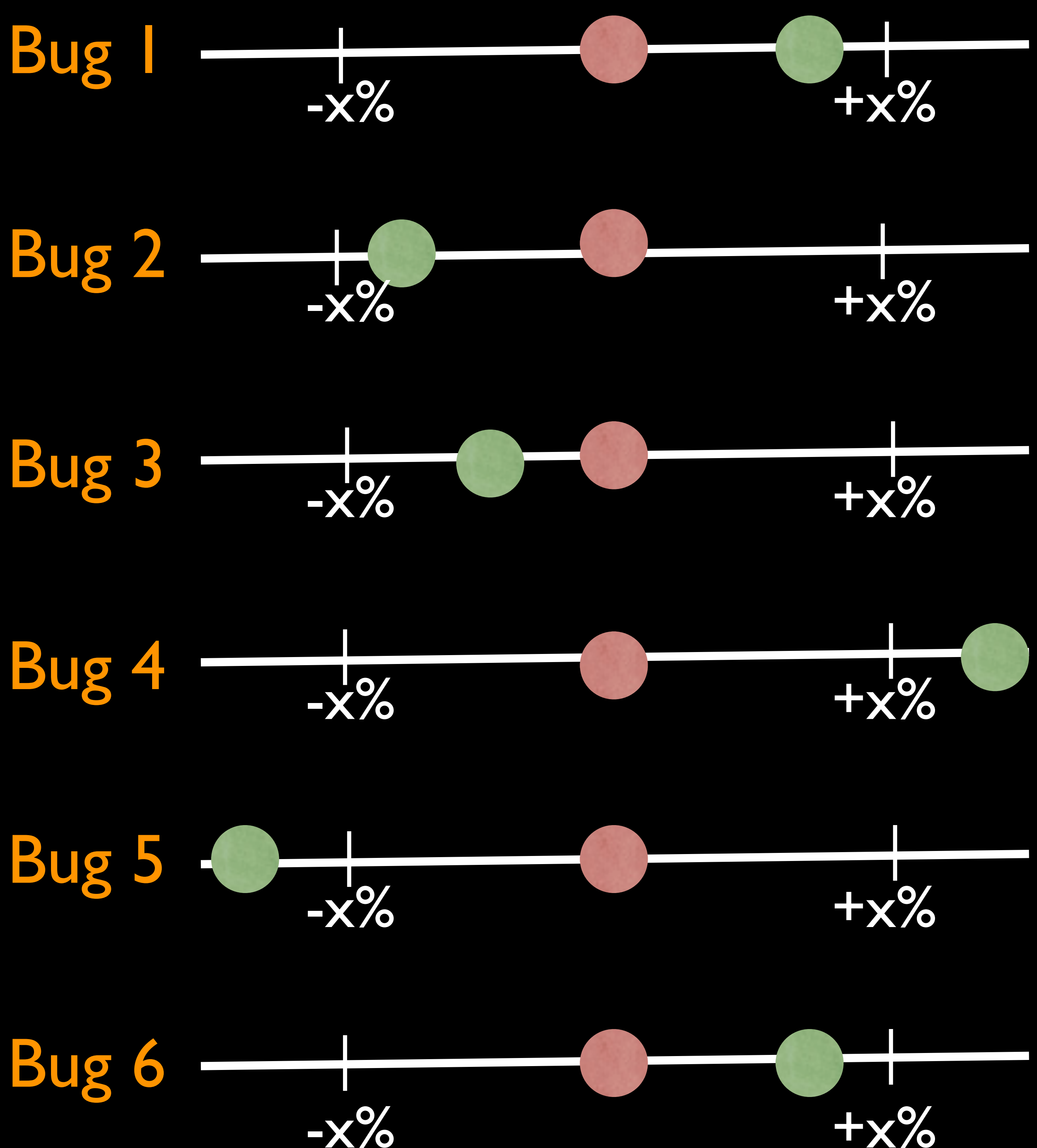
Evaluation of Results

Pred(x)
% of predictions that lie within x% of Actual Effort



Evaluation of Results

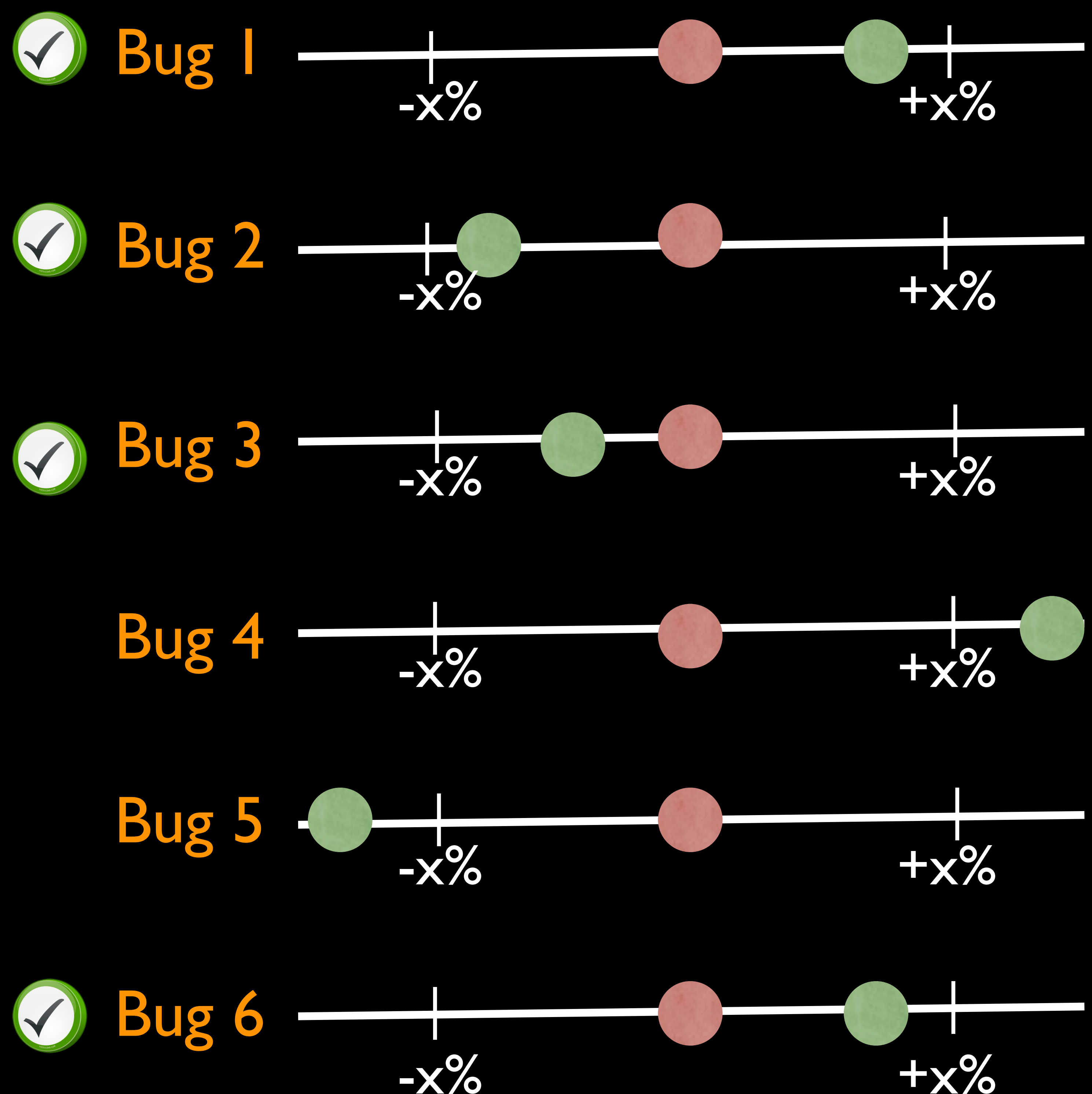
Pred(x)
% of predictions
that lie within x%
of Actual Effort



Evaluation of Results

Pred(x)

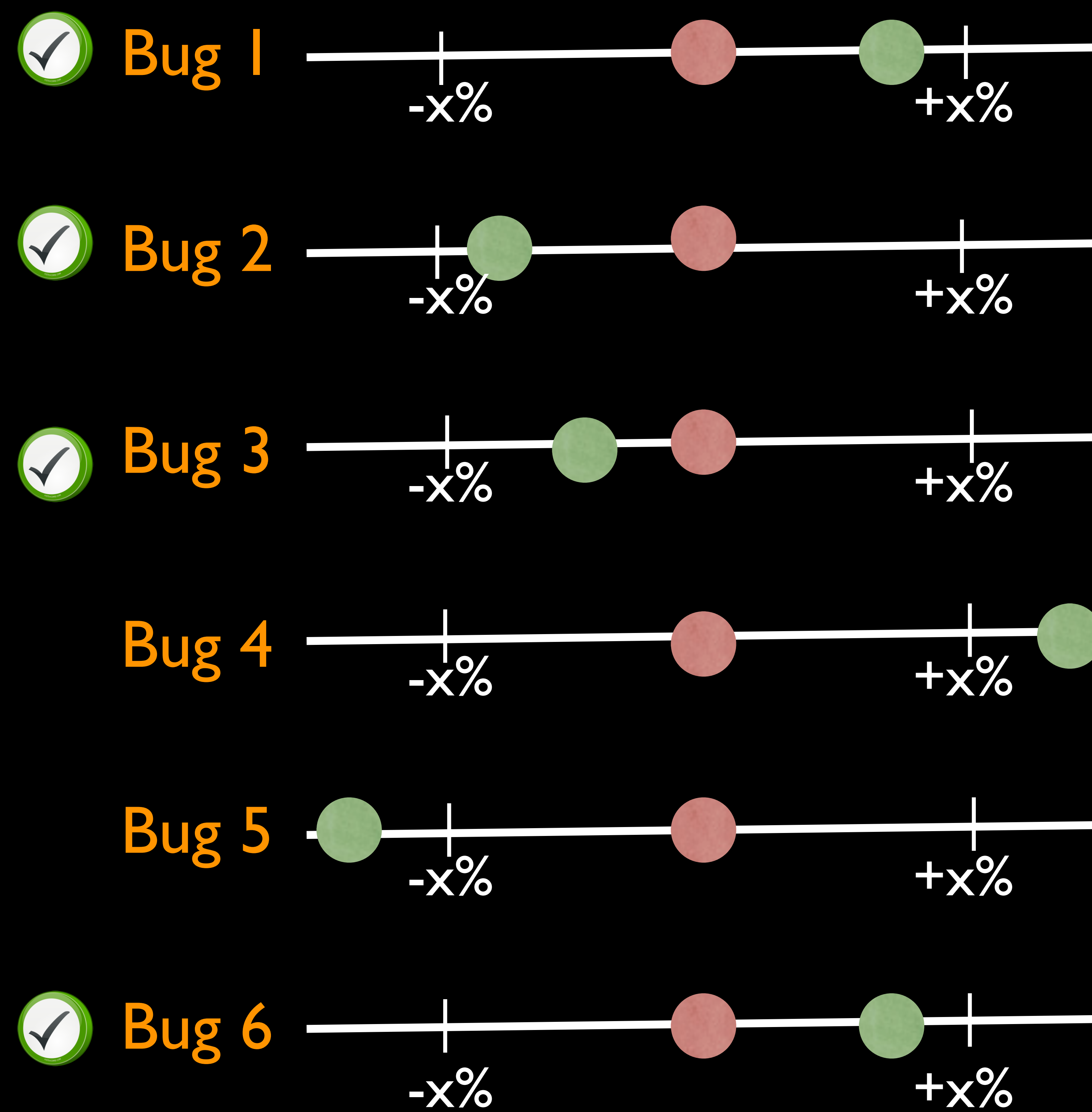
% of predictions that lie within x% of Actual Effort



Evaluation of Results

Pred(x)

% of predictions that lie within x% of Actual Effort



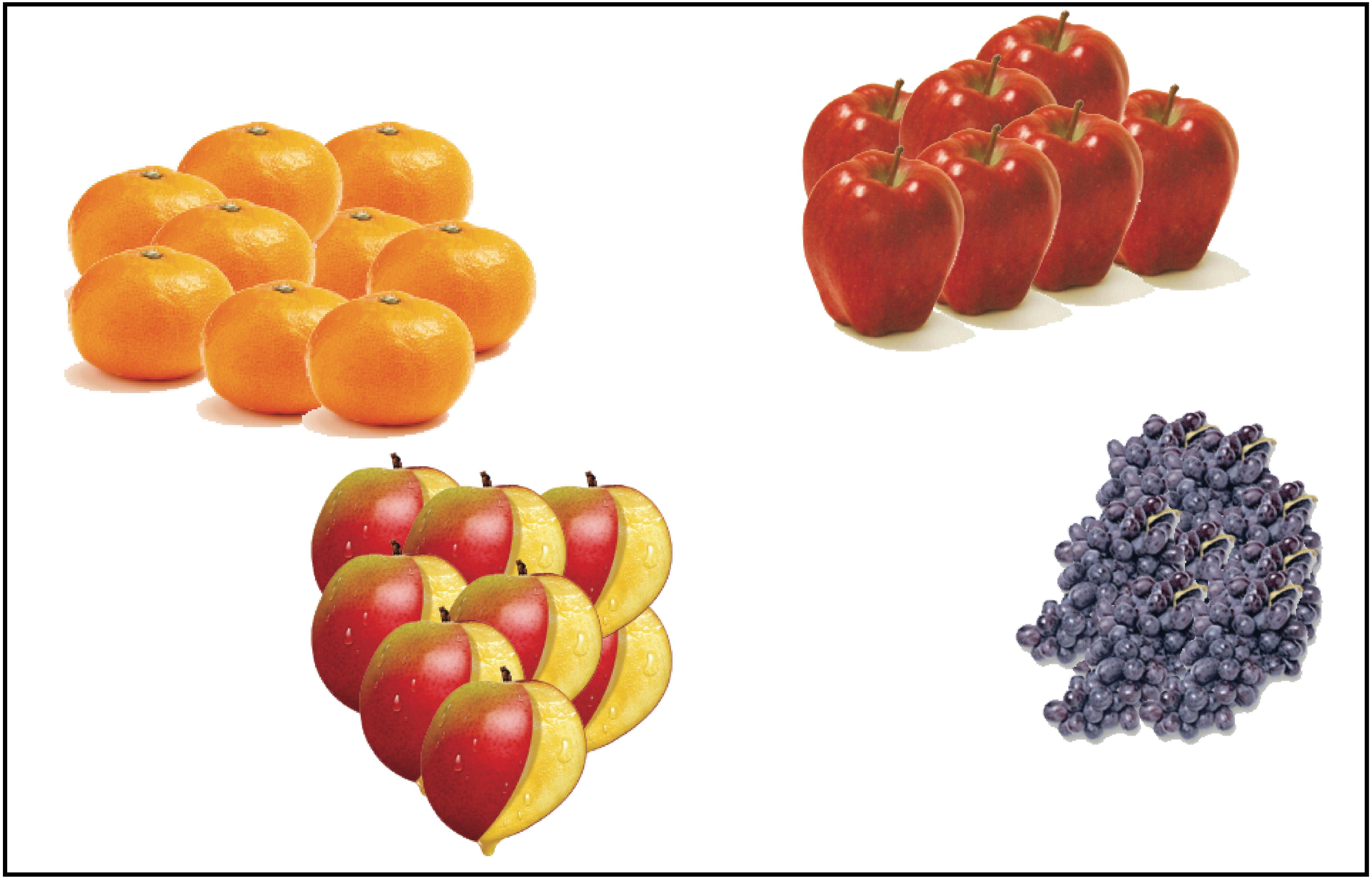
4 of 6 predictions made lie within x% of Actual Effort

Pred(x) = 66.67%

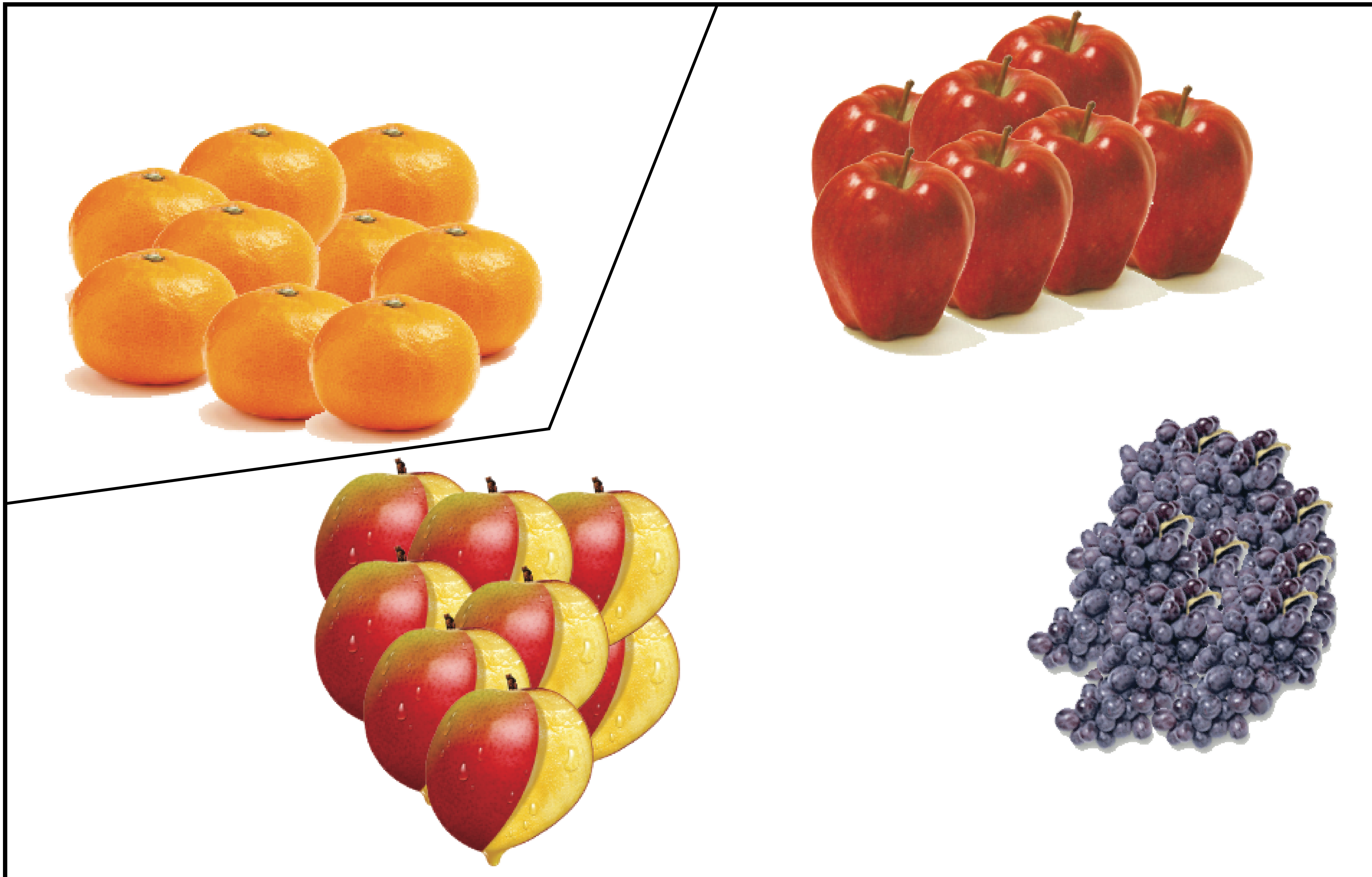
Clustering

Clustering

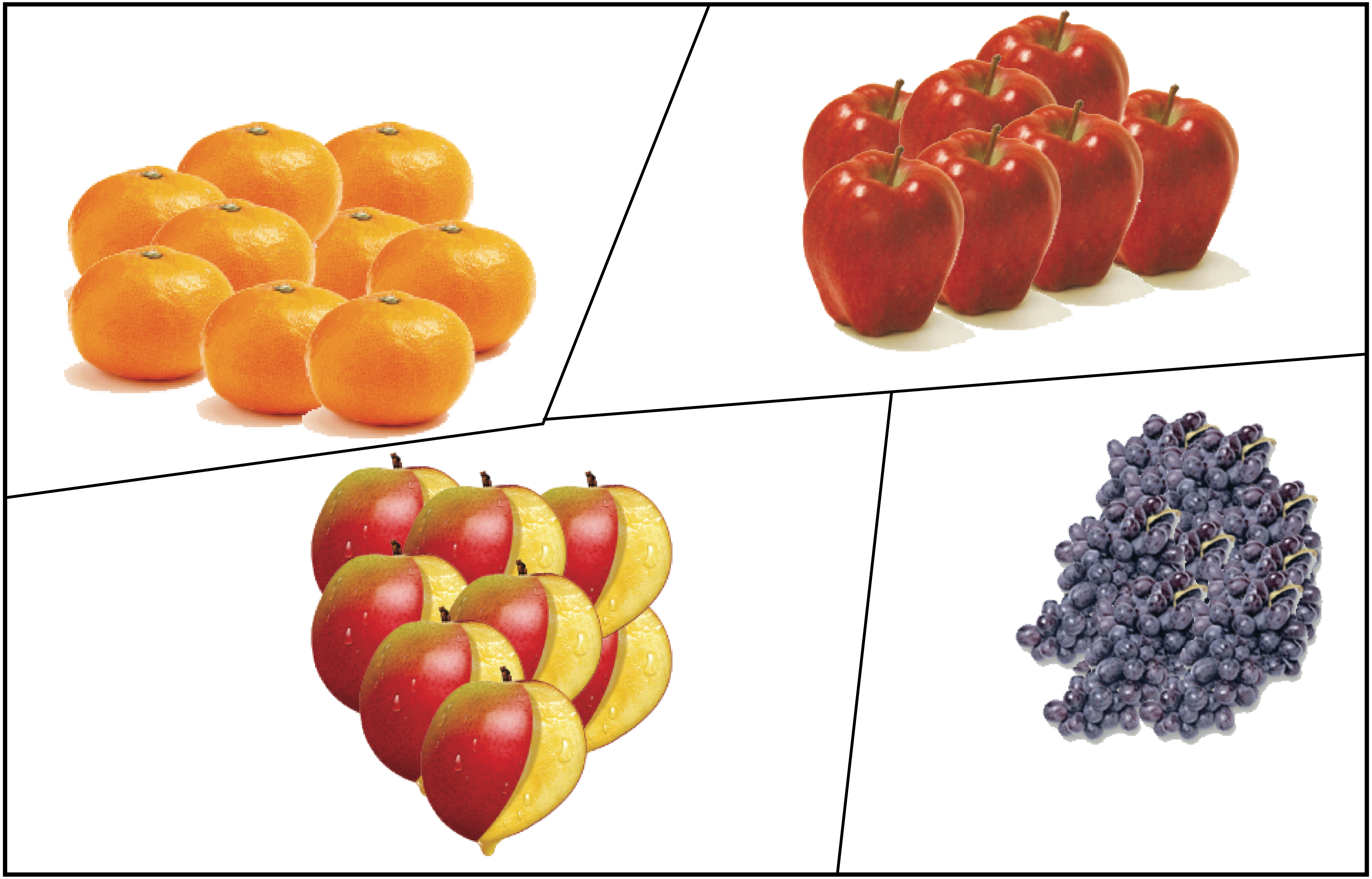
Clustering



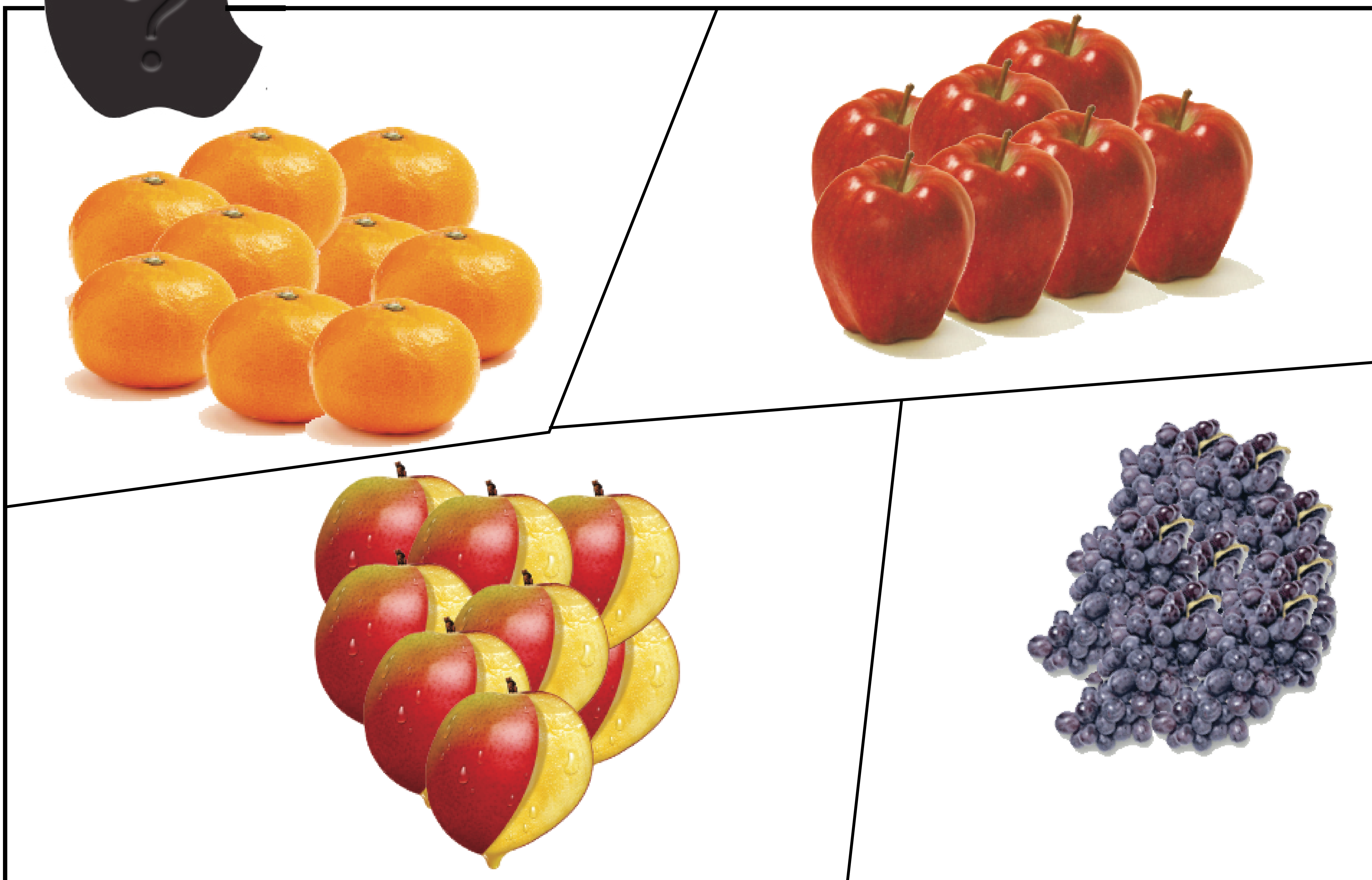
Clustering



Clustering



Clustering



Bayesian Filtering

Bayesian Filtering

Bayesian Filtering

Bayesian Filtering

Bayesian Filtering

Congratulations! You won the lottery...

$23 million in your custody

Nigeria

Bayesian Filtering

Congratulations! You won the lottery...

Congratulations! You won the competition...

$23 million in your custody

Nigeria