Finding local lessons in software engineering

Tim Menzies, WVU, USA, tim@menzies.us

GrammaTech, Dec'09

Sound bites

- An observation:
  - Surprisingly few general SE results.
- A requirement:
  - Need simple methods for finding local lessons.
- Take home lesson:
  - Finding useful local lessons is remarkably simple
  - E.g. using “W” or “NOVA”

Roadmap

- Motivation: generality in SE
- A little primer: DM for SE
- “W”: finding contrast sets
- “W”: case studies
- “W”: drawbacks
- “NOVA”: a better “W”
- Conclusions

Roadmap

- Motivation: generality in SE
- A little primer: DM for SE
- “W”: finding contrast sets
- “W”: case studies
- “W”: drawbacks
- “NOVA”: a better “W”
- Conclusions

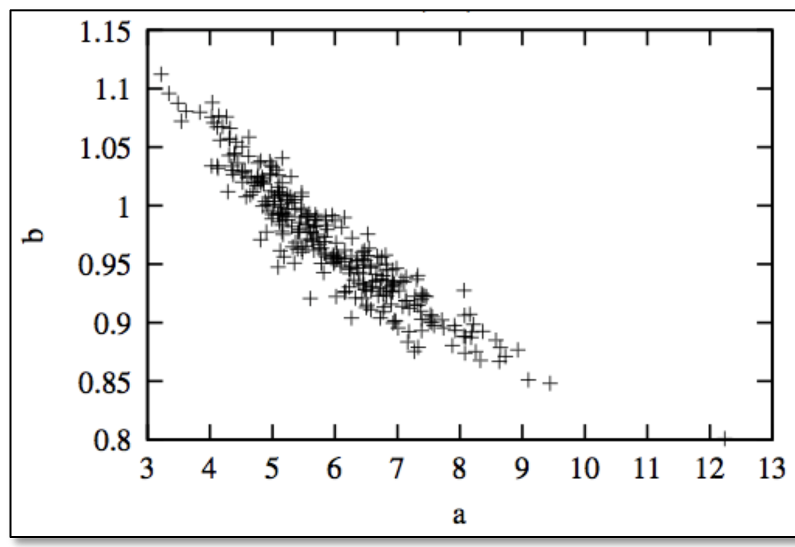
PROMISE

The 2010 International Conference on
Predictive Models in Software Engineering

Have we lived up to our PROMISE?

Few general results

- PROMISE 2005 … 2009 : 64 presentations
- 48 papers
  - tried a new analysis on old data
  - Or reported a new method that worked once for one project.
- 4 papers
  - argued against model generality
- 9 papers
  - questioned validity of prior results
- E.g. Menzies et al. Promise 2006
  - 100 times
    - Select 90% of the training data
    - Find<a,b> in effort = x.a.LOC ^{b}



Have we lived up to our PROMISE?

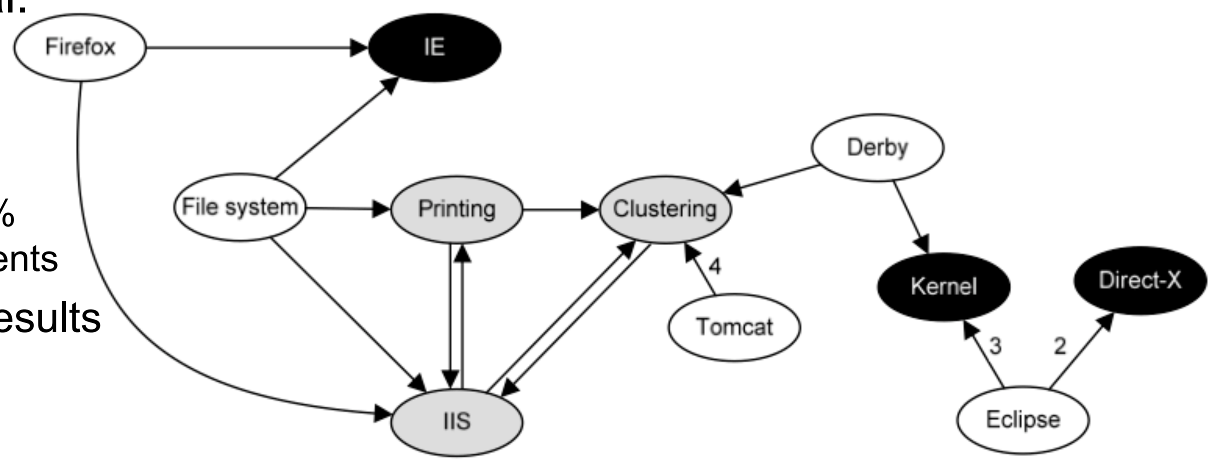
Only 11% of papers proposed general models

- E.g. Ostrand, Weyuker, Bell ‘08, ‘09
  - Same functional form
  - Predicts defects for generations of AT&T software
- E.g. Turhan, Menzies, Bener ’08, ‘09
  - 10 projects
    - Learn on 9
    - Apply to the 10th
  - Defect models learned from NASA projects work for Turkish whitegoods software
    - Caveat: need to filter irrelevant training examples

Less Promising Results

Lessons learned are very localized

- FSE'09: Zimmerman et al.
  - Defect models not generalizable
    - Learn "there", apply "here" only works in 4% of their 600+ experiments
  - Opposite to Turhan'09 results
    - ?add relevancy filter



- ASE'09: Green, Menzies et al.
  - AI search for better software project options
  - Conclusions highly dependent on local business value proposition

- And others
  - TSE'06: Menzies, Greenwald
  - Menzies et al. in ISSE 2007
  - Zannier et al ICSE'06

Overall

The gods are (a little) angry

- Fenton at PROMISE' 07
  - "... much of the current software metrics research is inherently irrelevant to the industrial mix ..."
  - "... any software metrics program that depends on some extensive metrics collection is doomed to failure ..."
- Budgen & Kitchenham:
  - "Is Evidence Based Software Engineering mature enough for Practice & Policy? "
  - Need for better reporting: more reviews.
  - Empirical SE results too immature for making policy.
- Basili : still far to go
  - But we should celebrate the progress made over the last 30 years.
  - And we are turning the corner

Experience Factories

Methods to find local lessons

- Basili'09 (pers. comm.):
  - "All my papers have the same form.
  - "For the project being studied, we find that changing X improved Y."

- Translation (mine):
  - Even if we can't find general models (which seem to be quite rare)….
  - … we can still research general methods for finding local lessons learned

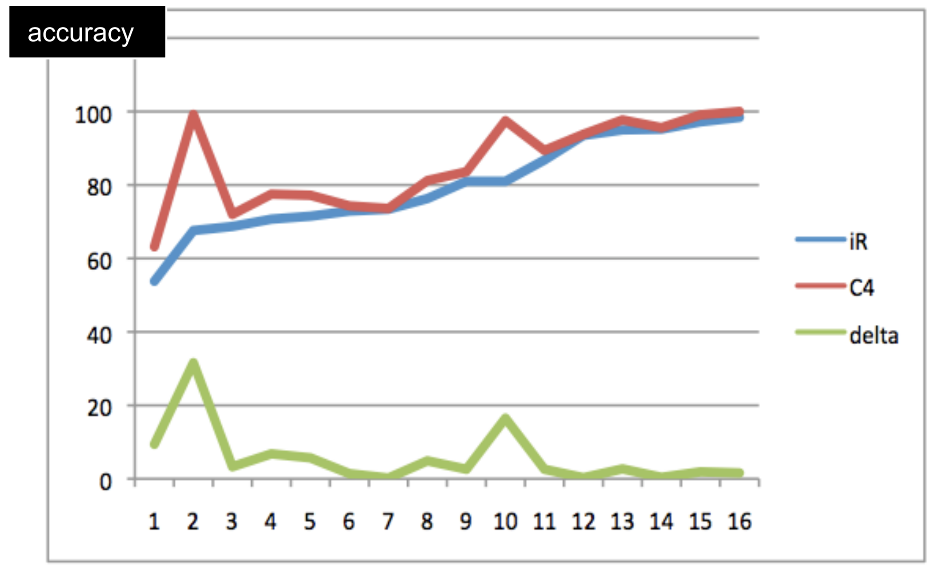
The rest of this talk: contrast set learning and "W"

W= a local lessons finder

- Bayesian case-based contrast-set learner
  - uses greedy search
  - illustrates the "local lessons" effect
  - offers functionality missing in the effort-estimation literature

- Fast generator of baseline results
  - There are too few baseline results
  - And baseline results can be very interesting (humbling).

- A very (very) simple algorithm
  - Should add it to your toolkit
  - At least, as the "one to beat"



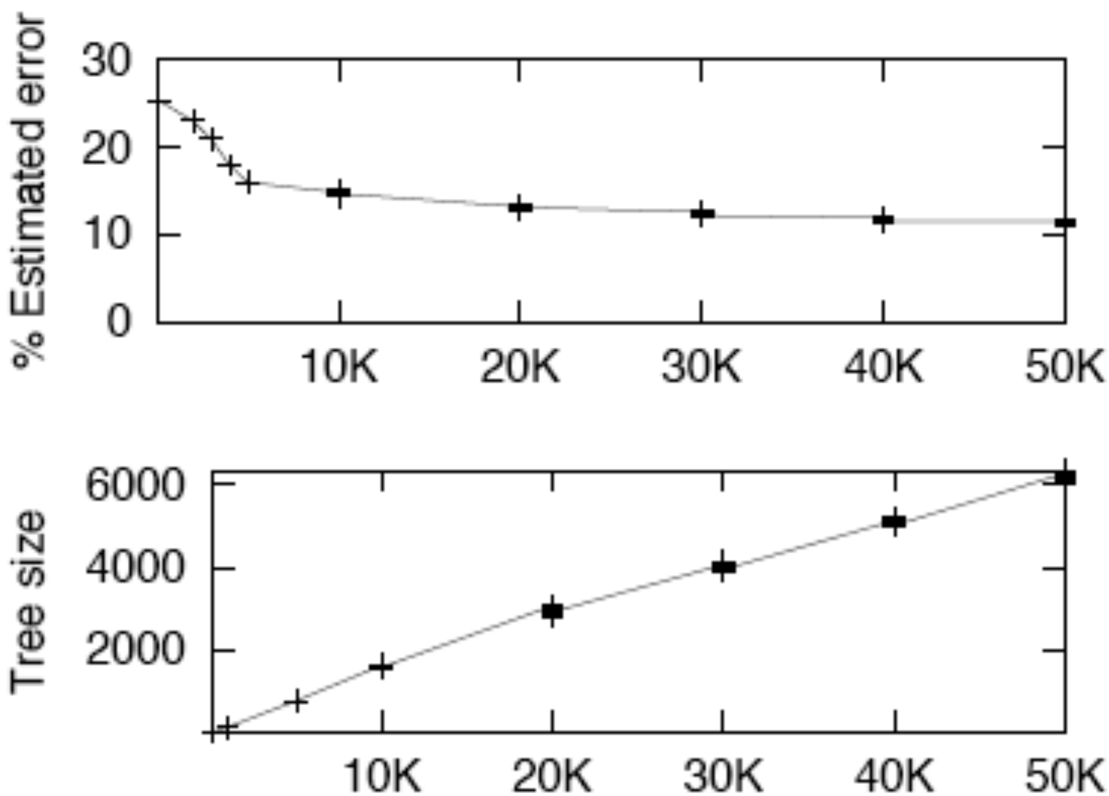
Holte'85
• C4: builds decision trees "N" deep
• 1R: builds decision trees "1" deep
• For datasets with 2 classes, 1R ≈ C4

Roadmap

- Motivation: generality in SE
- A little primer: DM for SE
- “W”: finding contrast sets
- “W”: case studies
- “W”: drawbacks
- “NOVA”: a better “W”
- Conclusions

Problem

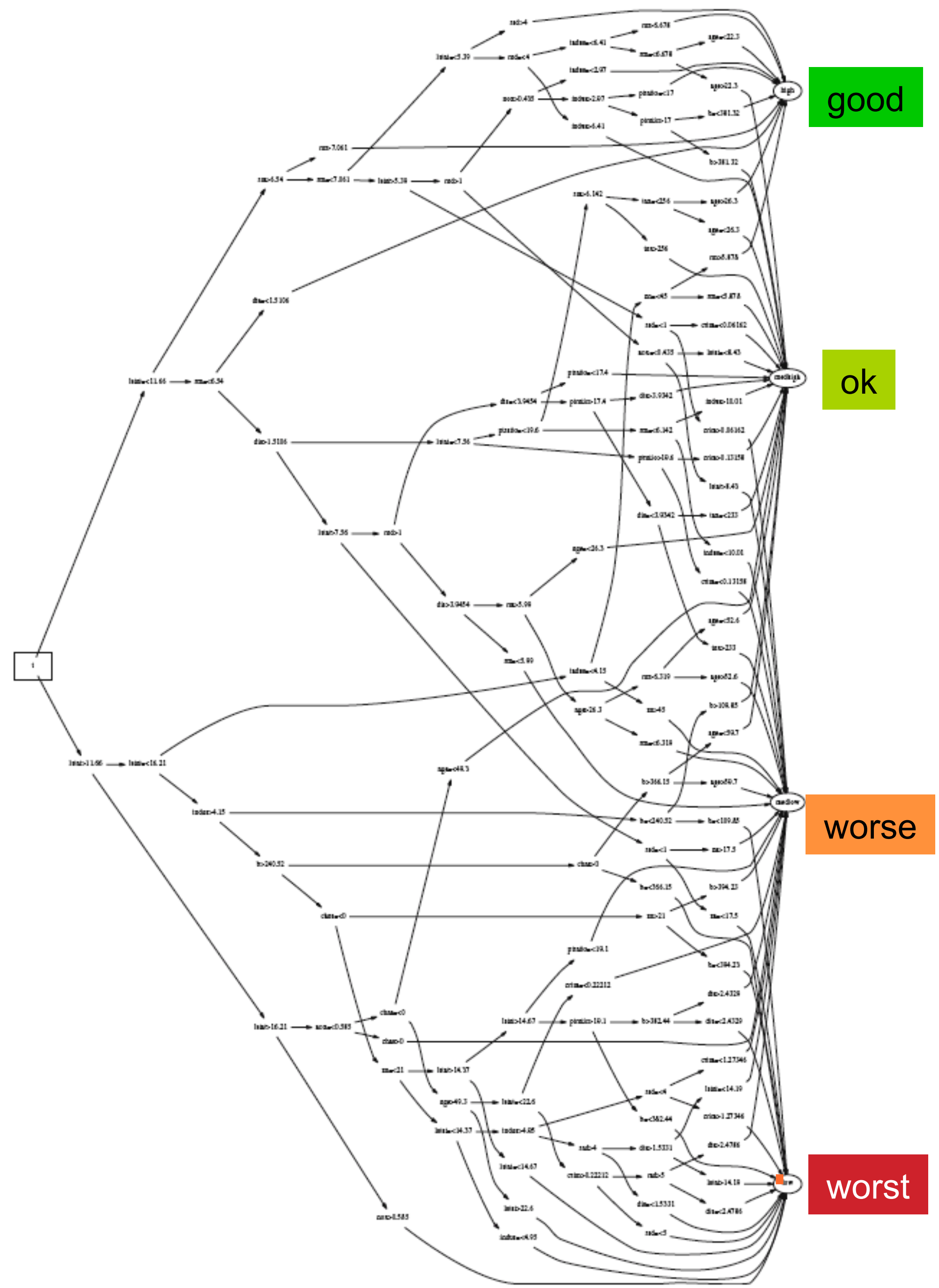
Too much information



Tree Pruning

Can you see the big picture?

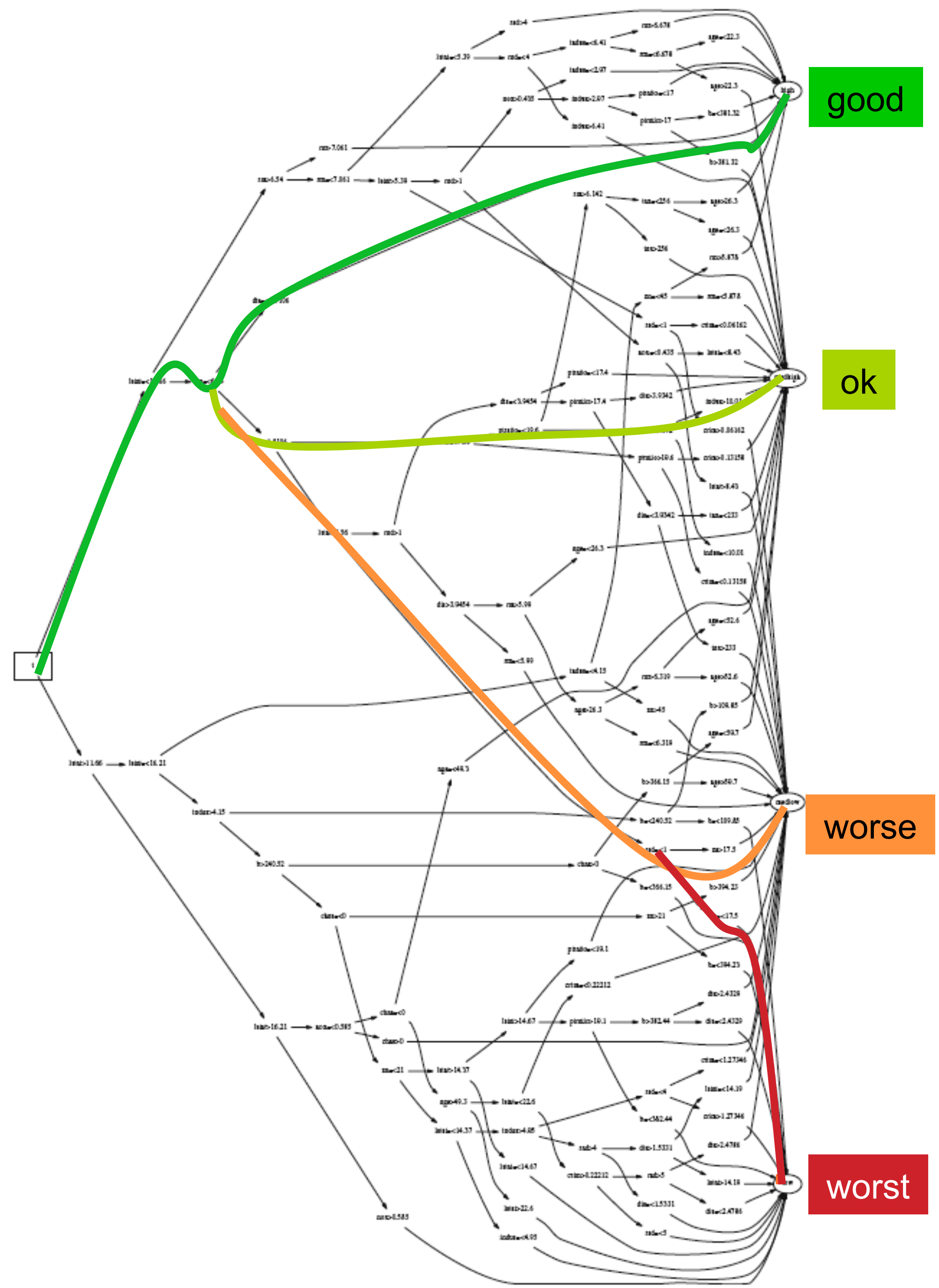
- Good branches go to good goals
- Bad branches go to bad goals
- Select decisions that select for
  - Most good
  - Least bad
- TARZAN:
  - swings through the trees
  - Post-processor to C4.5



Tree Pruning

Can you see the big picture?

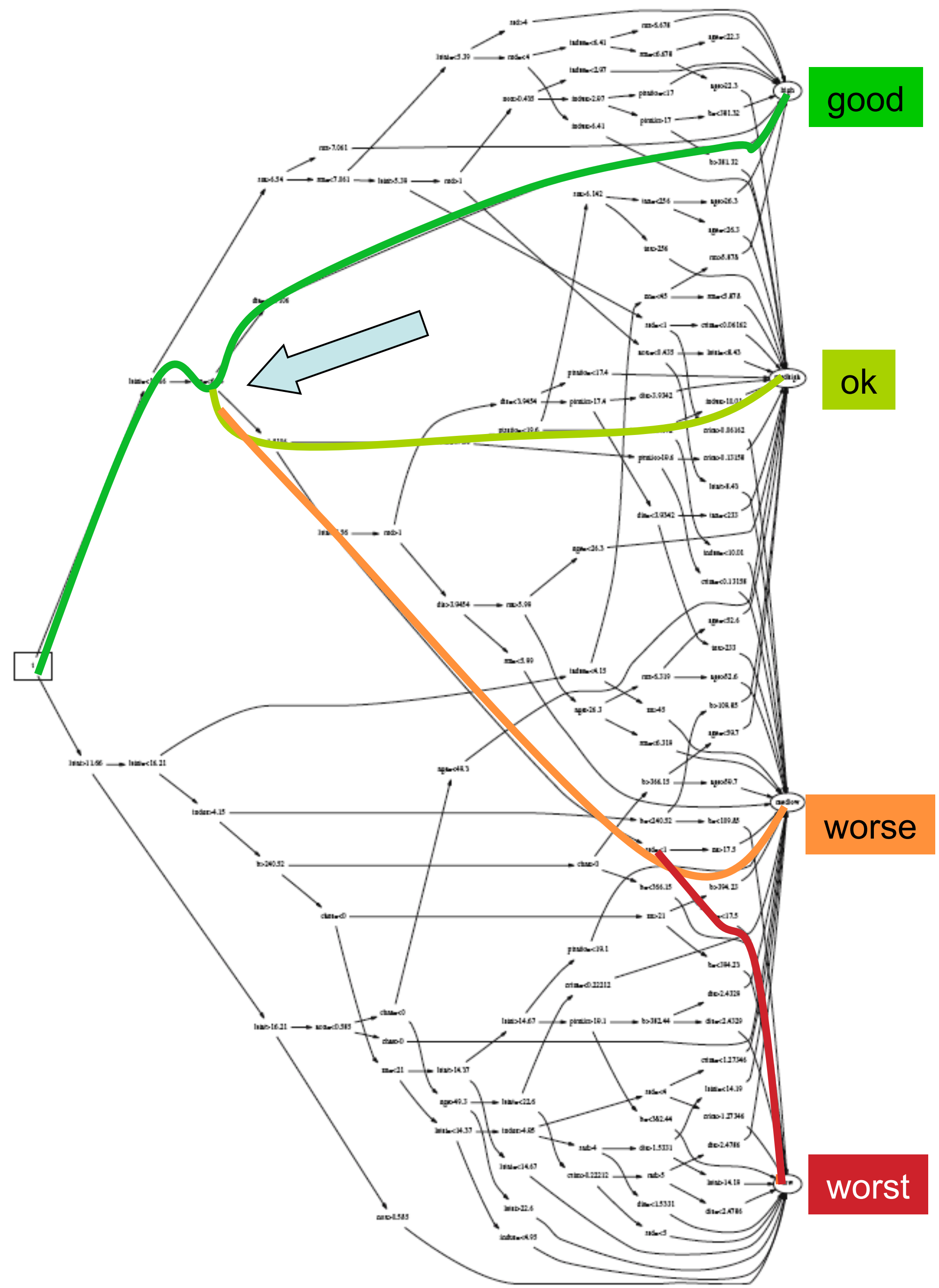
- Good branches go to good goals
- Bad branches go to bad goals
- Select decisions that select for
  - Most good
  - Least bad
- TARZAN:
  - swings through the trees
  - Post-processor to C4.5



Tree Pruning

Can you see the big picture?

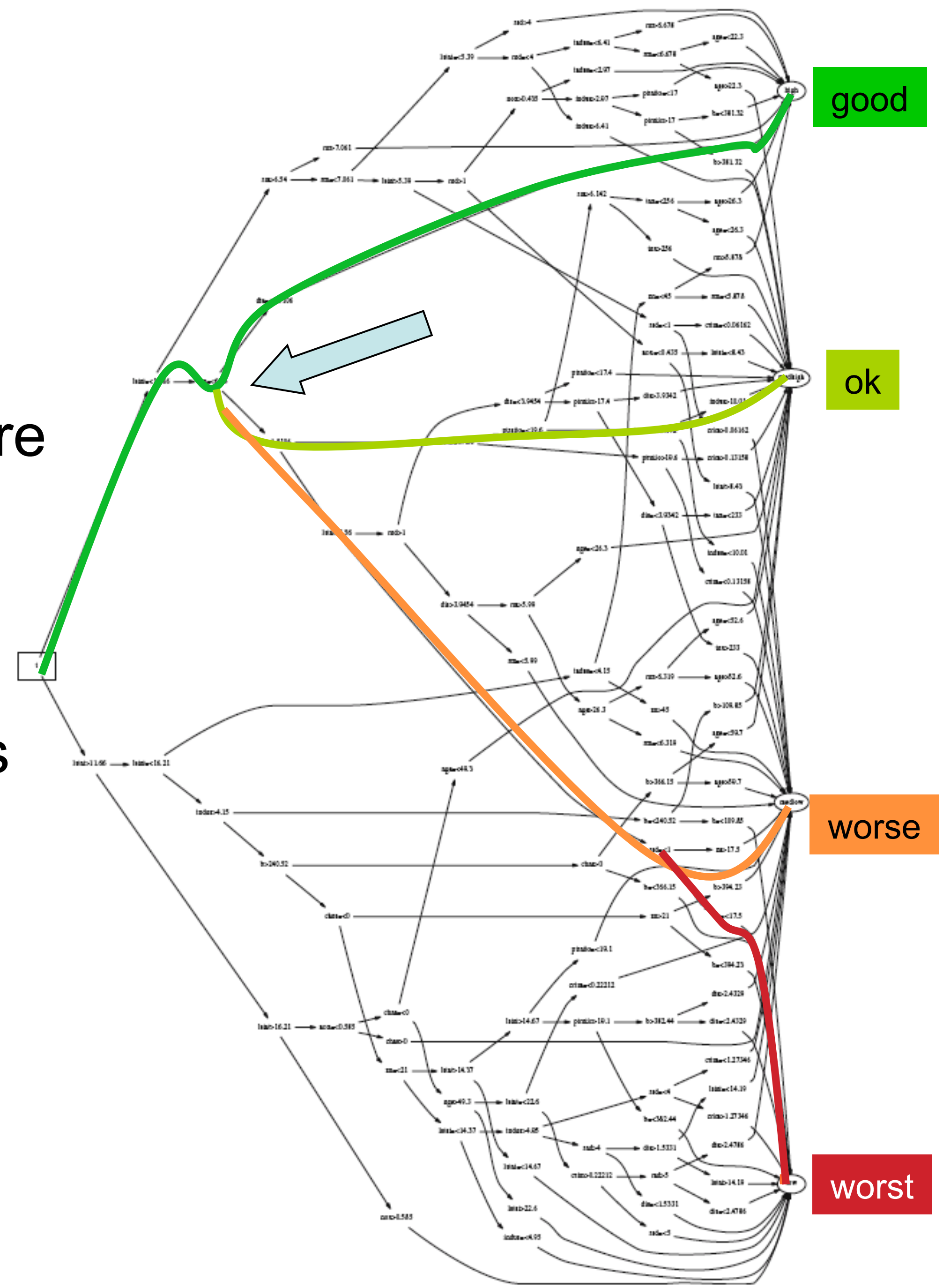
- Good branches go to good goals
- Bad branches go to bad goals
- Select decisions that select for
  - Most good
  - Least bad
- TARZAN:
  - swings through the trees
  - Post-processor to C4.5



Comment

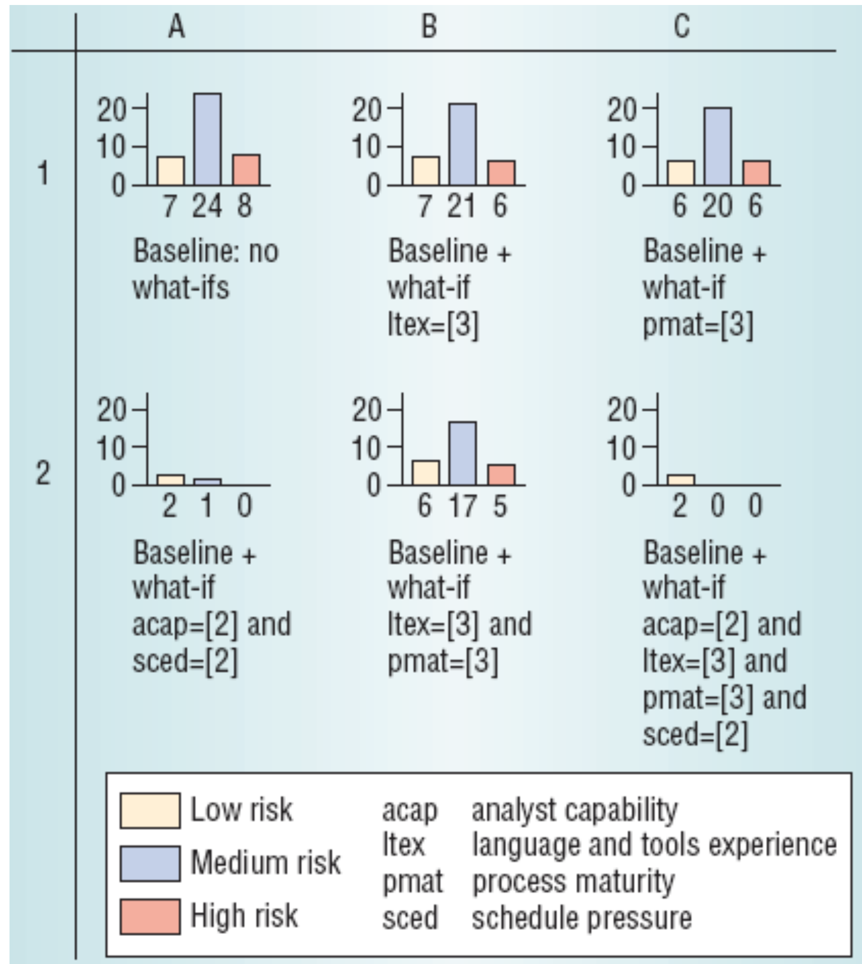
Less is best

- Higher decisions prune more branches
- #nodes at level I much smaller than level I+1.
- So tree pruning often yields very small sets of recommendations



Don’t bury me in data

Don’t show me “what is”; just tell what “to do”



Roadmap

- Motivation: generality in SE
- A little primer: DM for SE
- “W”: finding contrast sets
- “W”: case studies
- “W”: drawbacks
- “NOVA”: a better “W”
- Conclusions

"W"= Simple (Bayesian) Contrast Set Learning (in linear time)

Mozina: KDD'04

- "best" = target class (e.g. "survive")
- "rest" = other classes
- x = any range (e.g. "sex=female")
- $f(x|c)$ = frequency of x in class c

- $b = f(x \mid best) / F(best)$
- $r = f(x \mid rest) / F(rest)$

- LOR= log(odds ratio) = log(b/r)
  - ? normalize 0 to max = 1 to 100

- s = sum of LORs
  - e = 2.7183 …
  - p = F(B) / (F(B) + F(R))
  - P(B) = 1 / (1 + e^(-1*ln(p/(1 - p)) - s))

“W”:Simpler (Bayesian) Contrast Set Learning (in linear time)

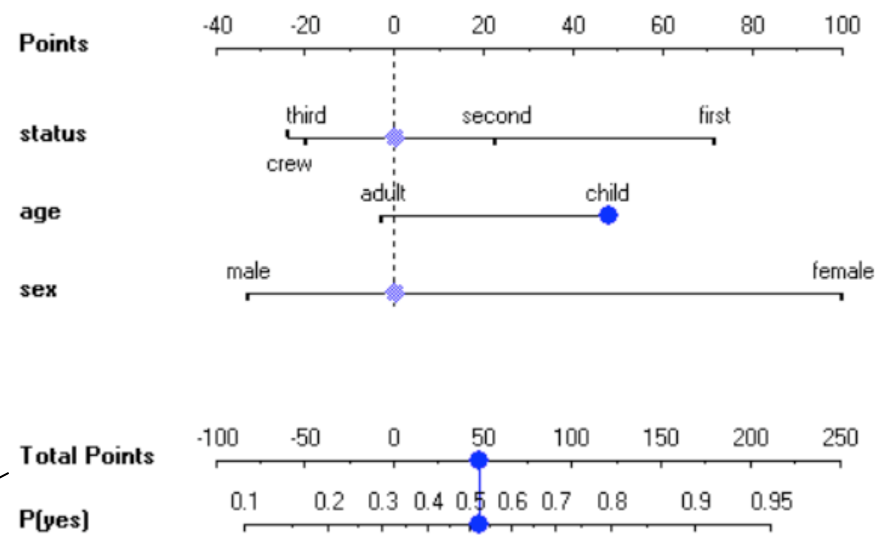
Mozina: KDD’04

- “best” = target class
- “rest” = other classes
- x = any range (e.g. sex = female)
- f(x|c) = frequency of x in class c

- b = f(x | best) / F(best)
- r = f(x | rest) / F(rest)

- LOR= log(odds ratio) = log(b/r)
  - ? normalize 0 to max = 1 to 100

- s = sum of LORs
  - e = 2.7183 …
  - p = F(B) / (F(B) + F(R))
  - P(B) = 1 / (1 + e^(-1*ln(p/(1 - p)) - s))

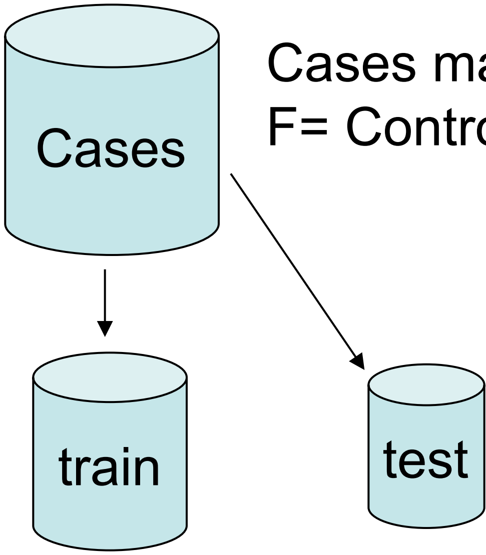


“W”:
1) Discretize data and outcomes
2) Count frequencies of ranges in classes
3) Sort ranges by LOR
4) Greedy search on top ranked ranges

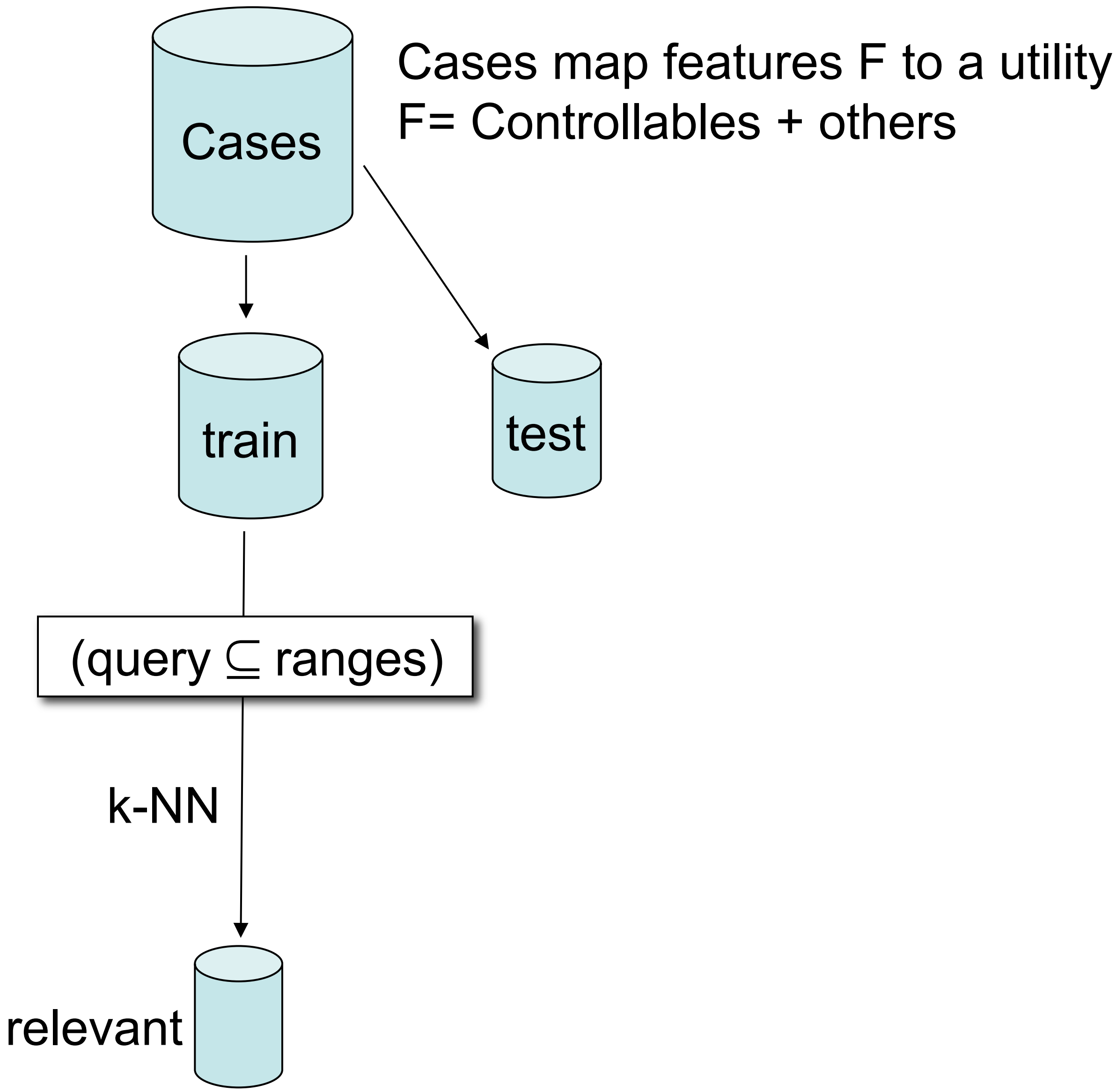
“W” + CBR

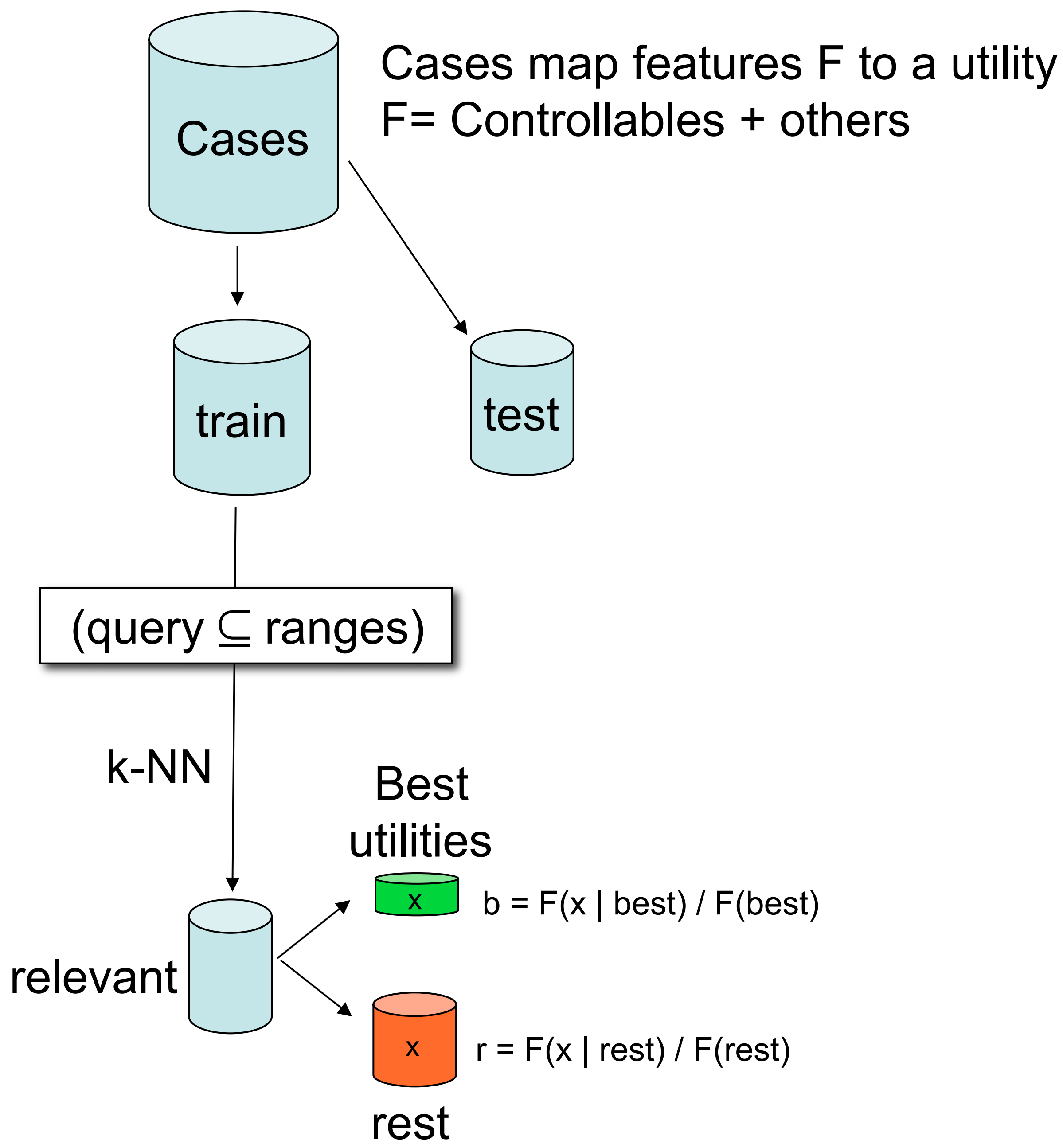
Preliminaries

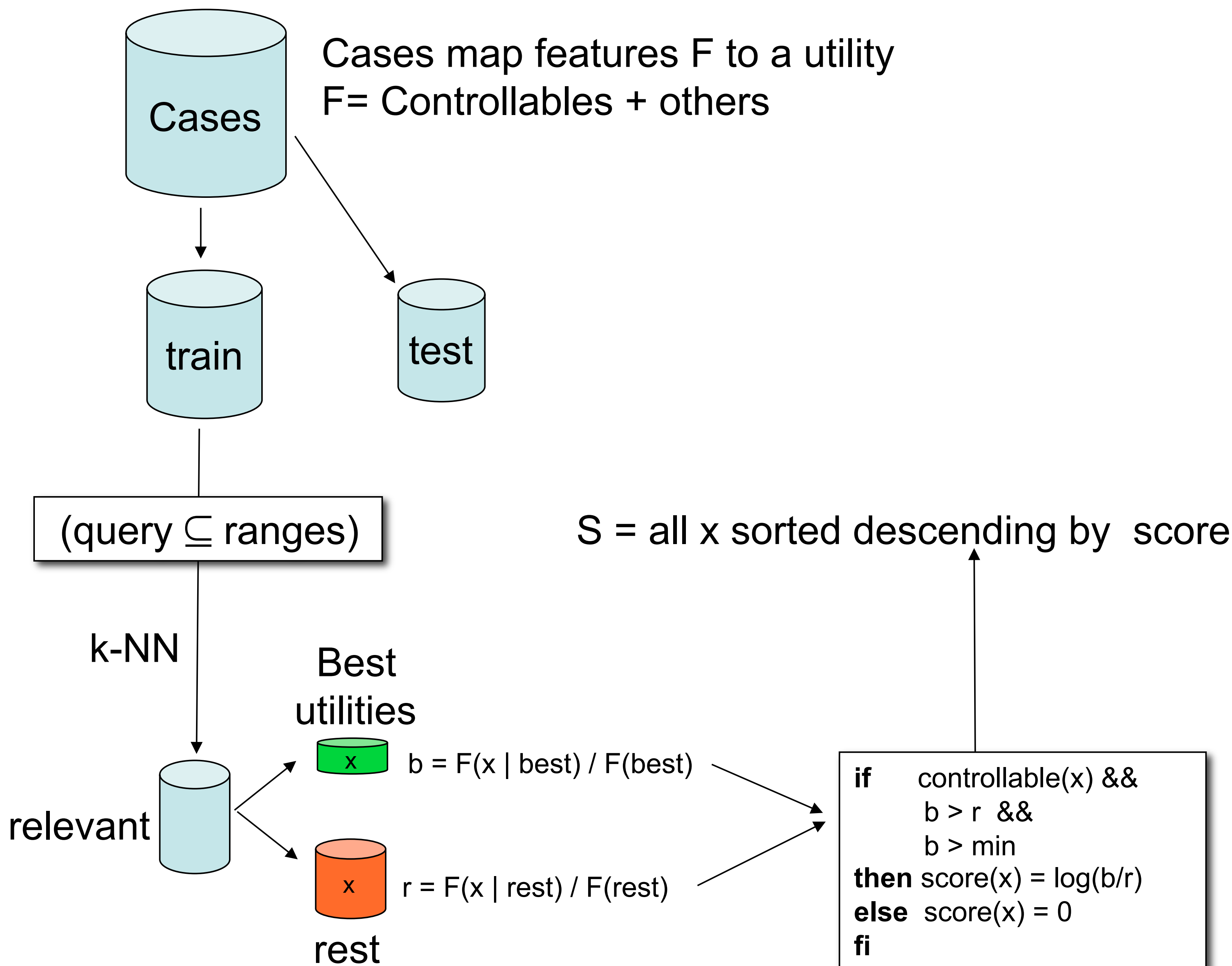
- “Query”
  - What kind of project you want to analyze; e.g.
    - Analysts not so clever,
    - High reliability system
    - Small KLOC
- “Cases”
  - Historical records, with their development effort
- Output:
  - A recommendation on how to change our projects in order to reduce development effort

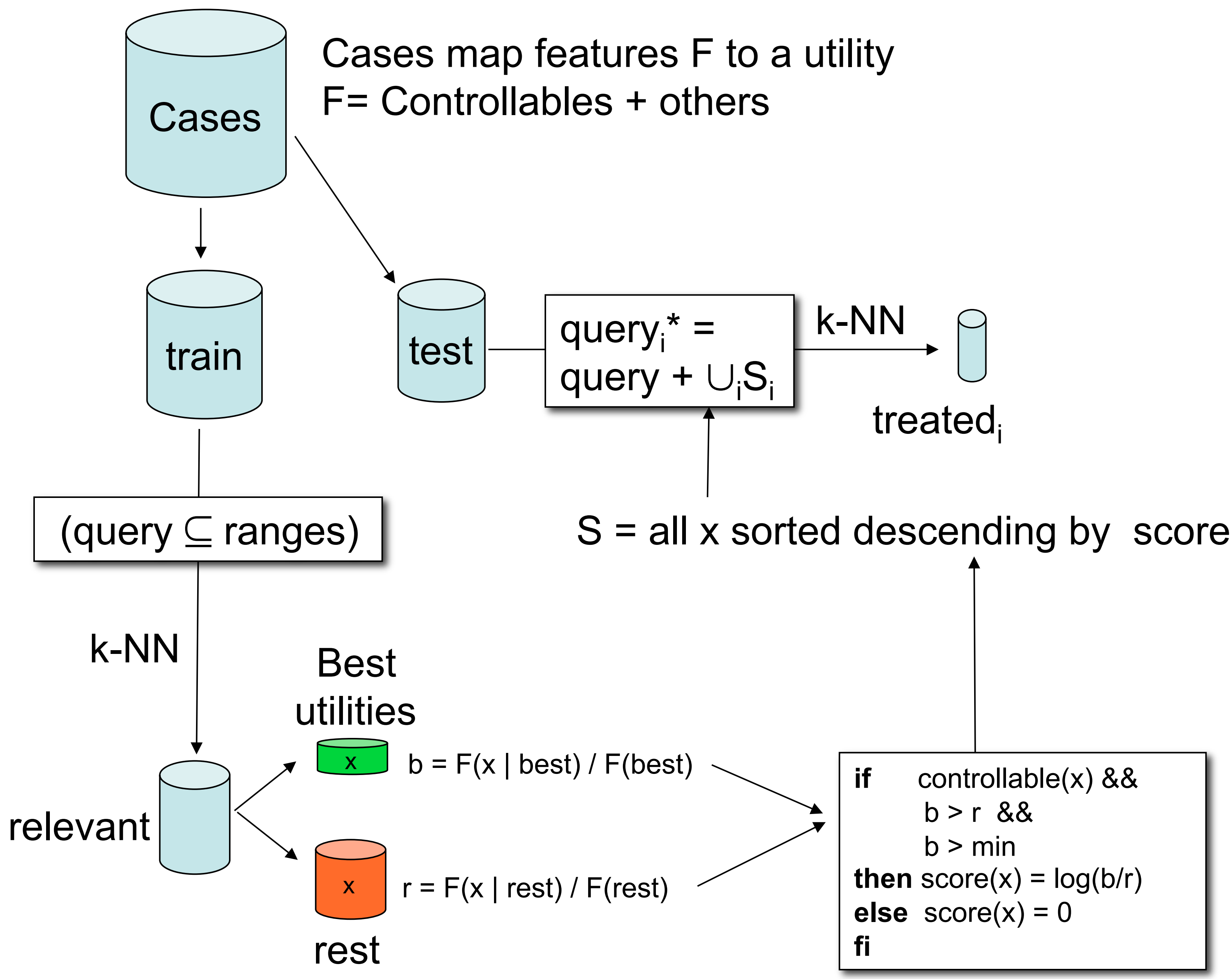


Cases map features F to a utility
F= Controllables + others

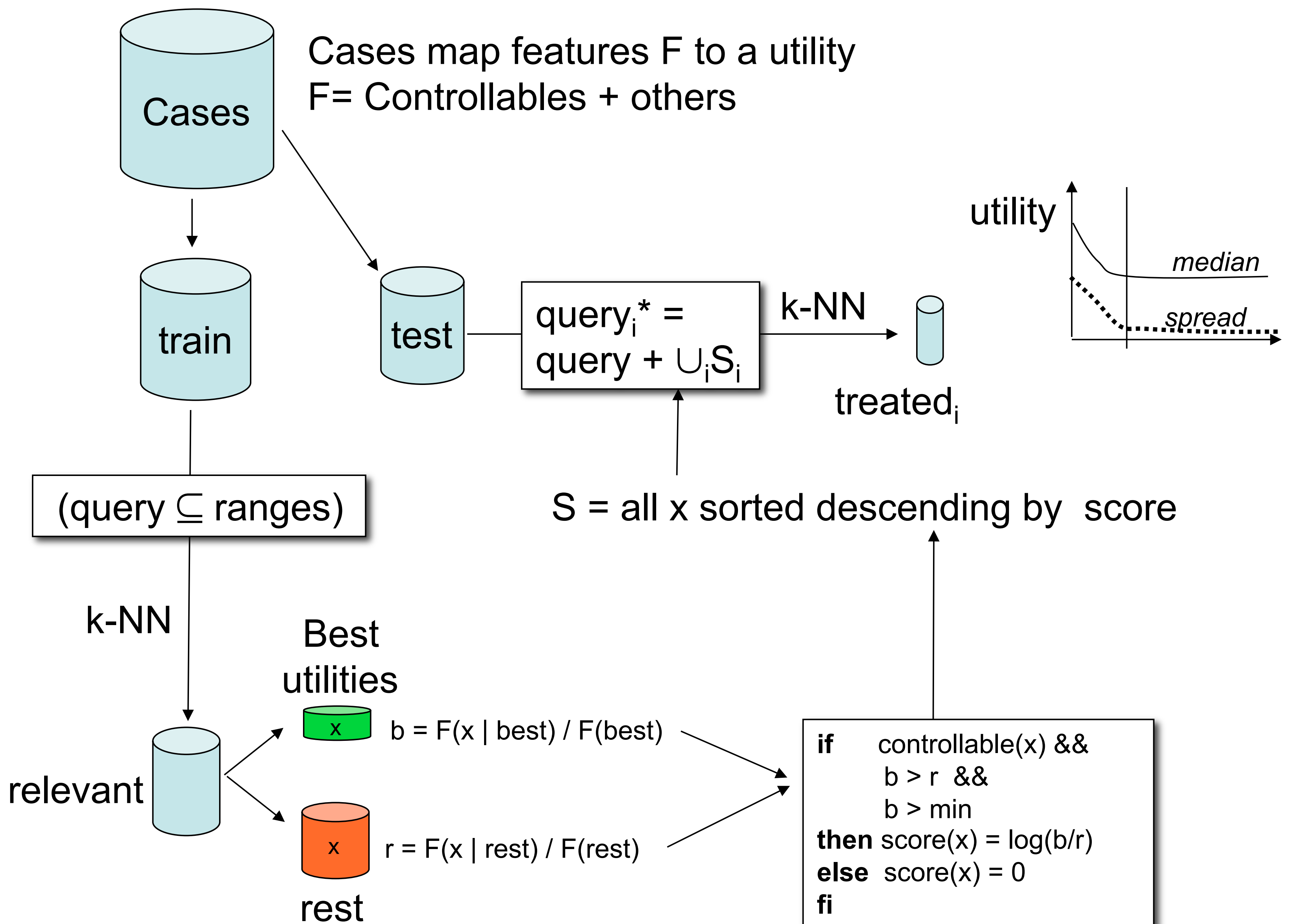

Cases
Cases map features F to a utility
F= Controllables + others
train
test
(query ⊆ ranges)
k-NN
relevant


Cases map features F to a utility
F= Controllables + others
Cases
train
test
(query ⊆ ranges)
k-NN
relevant
Best
utilities
x
b = F(x | best) / F(best)
x
r = F(x | rest) / F(rest)
rest

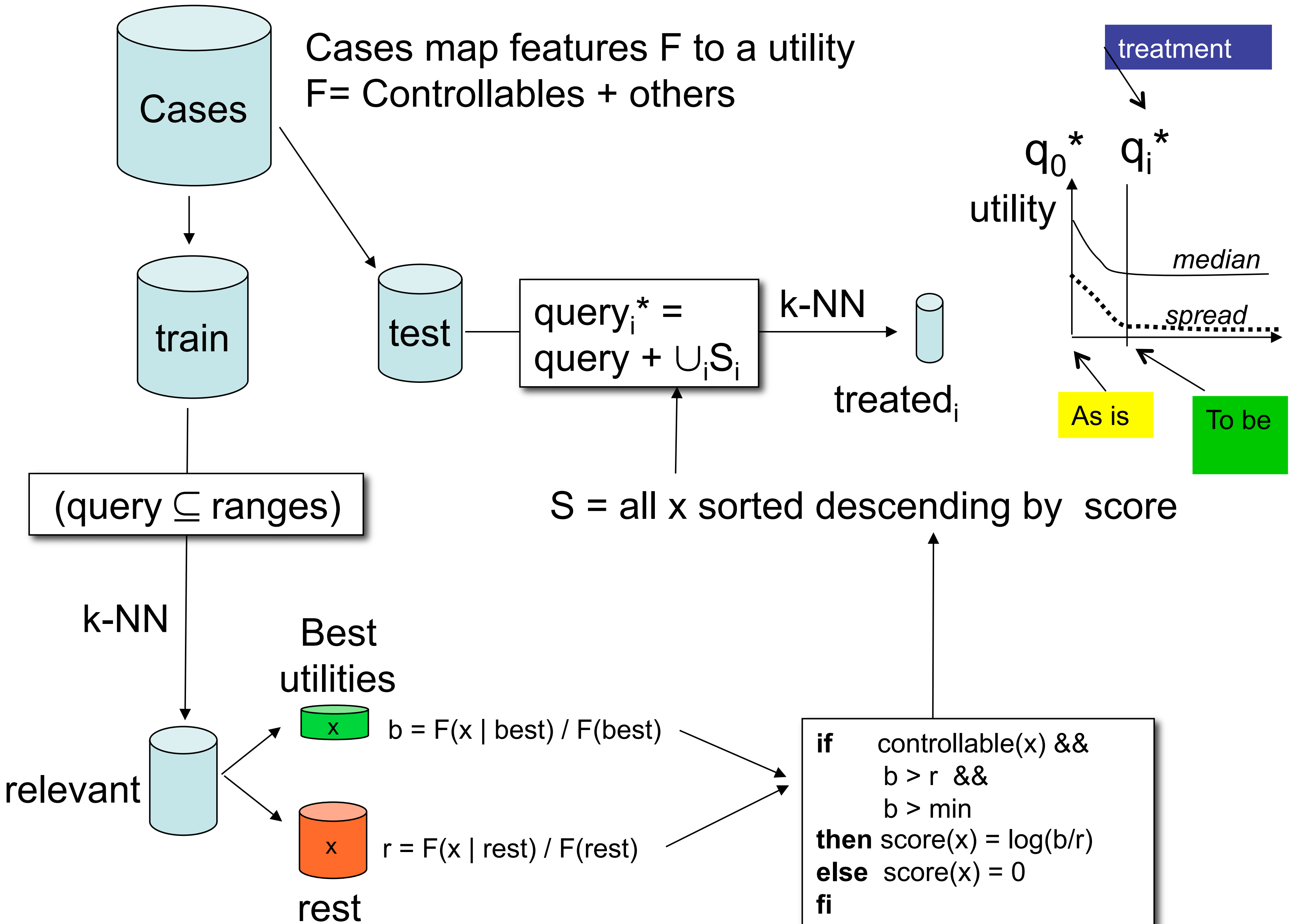

Cases
Cases map features F to a utility
F= Controllables + others
train
test
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
Best
utilities
x
b = F(x | best) / F(best)
relevant
x
r = F(x | rest) / F(rest)
rest
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi



Cases
Cases map features F to a utility
F= Controllables + others
train
test
query$_i$* =
query + $\cup_i S_i$
k-NN
treated$_i$
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
Best
utilities
x
b = F(x | best) / F(best)
relevant
x
r = F(x | rest) / F(rest)
rest
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi



Cases
Cases map features F to a utility
F= Controllables + others
train
test
query$_i$* =
query + $\cup_i S_i$
k-NN
treated$_i$
utility
median
spread
i
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
Best
utilities
x
b = F(x | best) / F(best)
relevant
x
r = F(x | rest) / F(rest)
rest
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi


Cases
Cases map features F to a utility
F= Controllables + others
treatment
$q_0{}^*$
$q_i{}^*$
utility
median
spread
i
As is
To be
train
test
$query_i{}^* =$
$query + \cup_i S_i$
k-NN
$treated_i$
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
Best
utilities
x
b = F(x | best) / F(best)
relevant
x
r = F(x | rest) / F(rest)
rest
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi

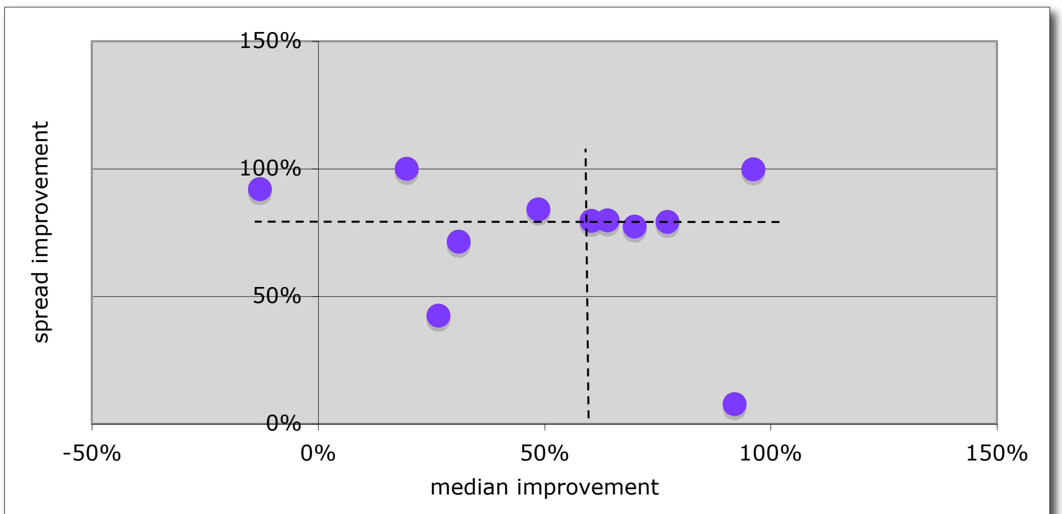
Roadmap

- Motivation: generality in SE
- A little primer: DM for SE
- “W”: finding contrast sets
- “W”: case studies
- “W”: drawbacks
- “NOVA”: a better “W”
- Conclusions

Results (distribution of development efforts in q_i*)

Using cases from http://promisedata.org

		X = as is		Y = to be		(X-Y) / X	
cases	query	median	spread	median	spread	median	spread
coc81	allSmall	70	920	79	73	-13%	92%
coc81	flight	87	281	70	0	20%	100%
nasa93	osp2	409	653	300	376	27%	42%
coc81	osp2	87	483	60	138	31%	71%
nasa93	osp	409	781	210	125	49%	84%
nasa93	allSmall	409	588	162	120	60%	80%
coc81	allLarge	50	158	18	32	64%	80%
nasa93	allLarge	300	660	90	150	70%	77%
nasa93	ground	360	481	82	100	77%	79%
coc81	osp	88	483	7	446	92%	8%
coc81	ground	156	478	6	1	96%	100%
nasa93	flight	360	474				



Cases from *promisedata.org/data*

Median = 50% percentile
Spread = 75% - 25% percentile

Improvement = (X - Y) / X
- X = as is
- Y = to be
- <u>more</u> is better

Usually:
- spread ≥ 75% improvement
- median ≥ 60% improvement

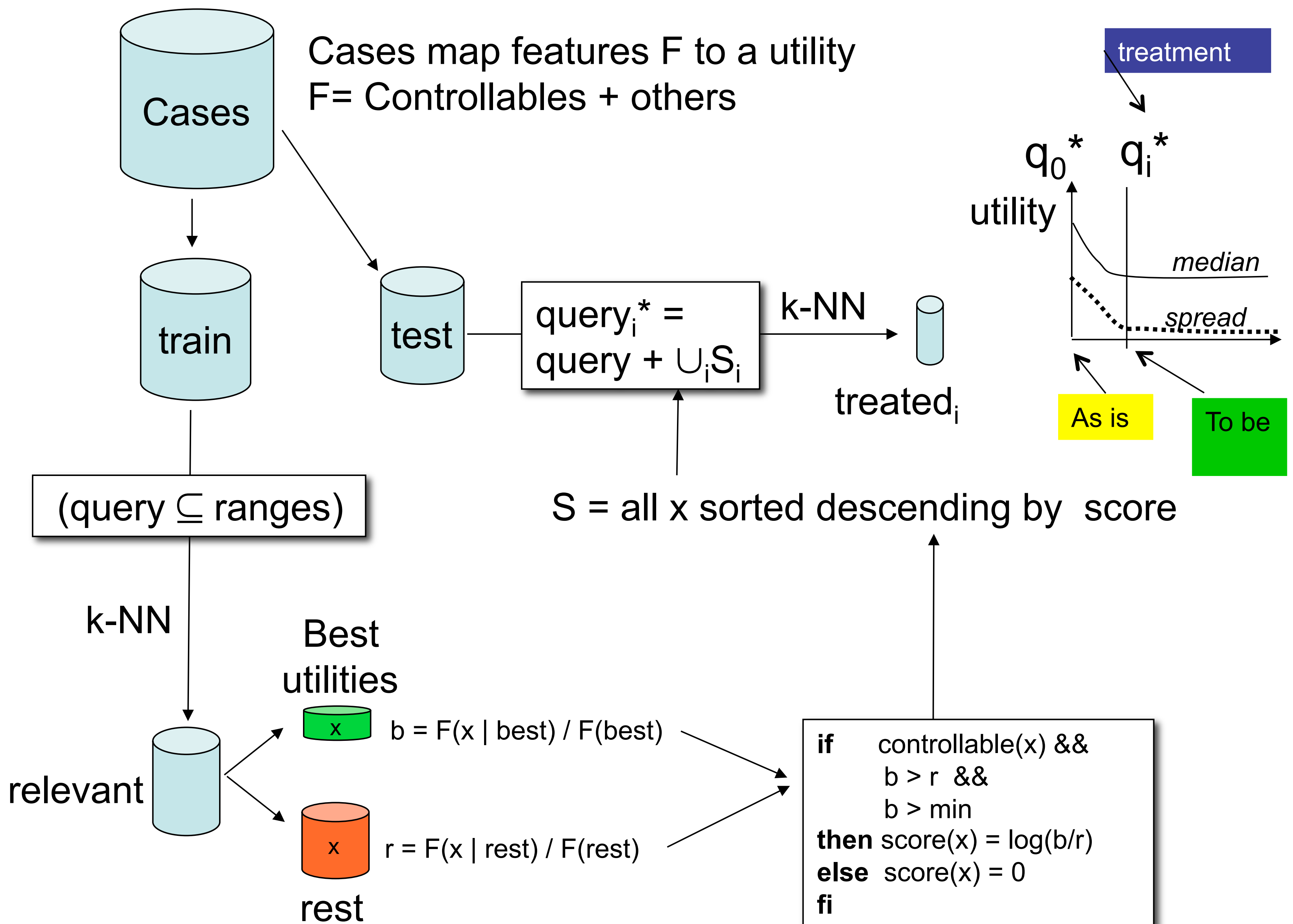
Not-so-good news

Local lessons are very localized

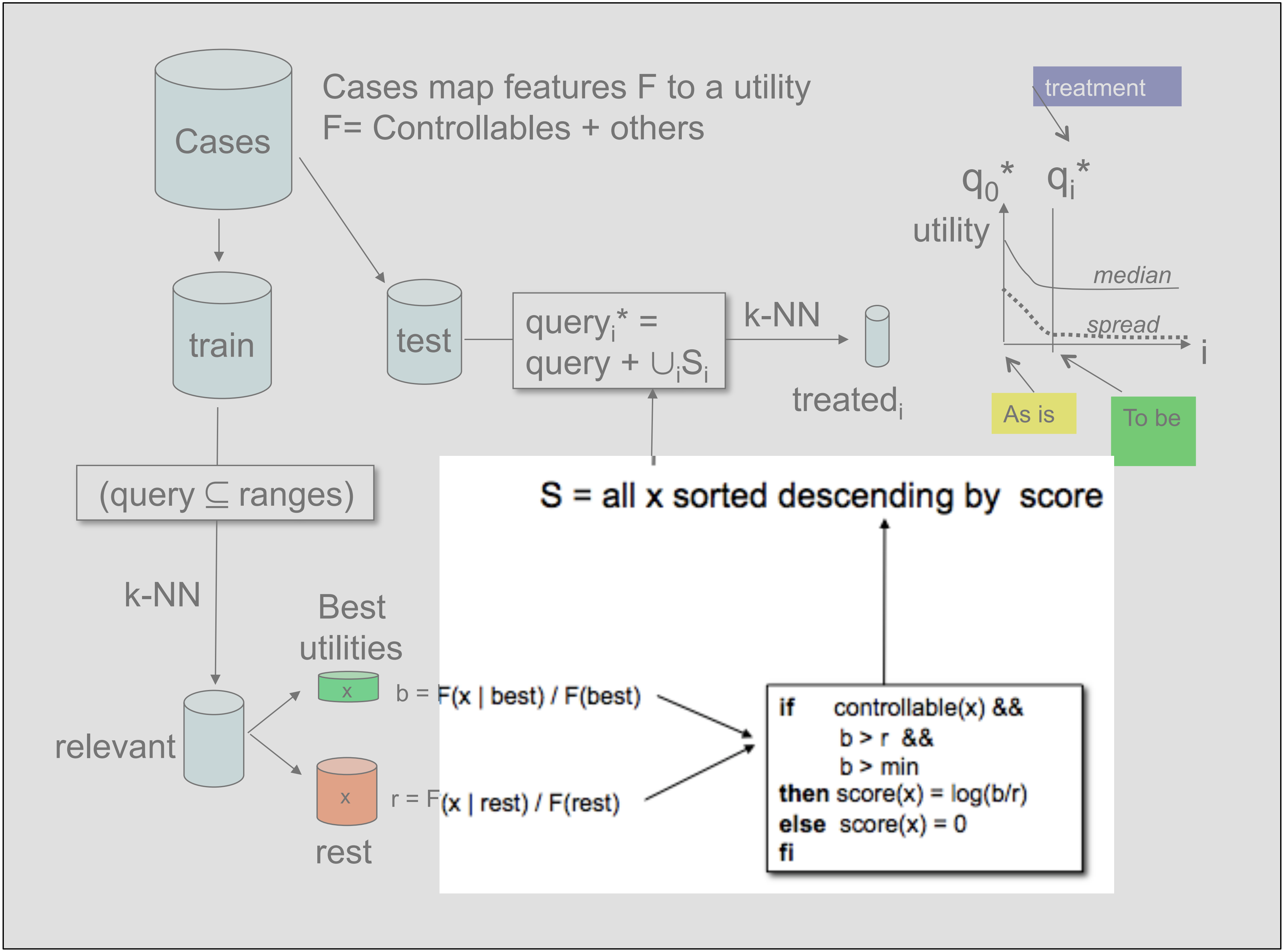
		acap	aexp				cplx	data		modp	pcap	sced			stor			time	tool	turn	vexp	
cases	*query*	*3*	*1*	*2*	*4*	*5*	*3*	*2*	*3*	*3*	*4*	*1*	*2*	*3*	*3*	*4*	*5*	*3*	*3*	*2*	*3*	
coc81	allSmall							■													■	2
coc81	flight	■						■								■				■		4
nasa93	osp2												■	■								1
coc81	osp2				■							■		■								2
nasa93	osp				■										■				■			3
nasa93	allSmall								■	■	■											3
coc81	allLarge																					0
nasa93	allLarge									■									■			2
nasa93	ground																■	■				2
coc81	osp		■	■																		1
coc81	ground	■					■								■							3
nasa93	flight																					0
		2	1	1	2		1	2	1	2	1	1	1	2	2	1	1	1	2	1	1	

Roadmap

- Motivation: generality in SE
- A little primer: DM for SE
- “W”: finding contrast sets
- “W”: case studies
- “W”: drawbacks
- “NOVA”: a better “W”
- Conclusions

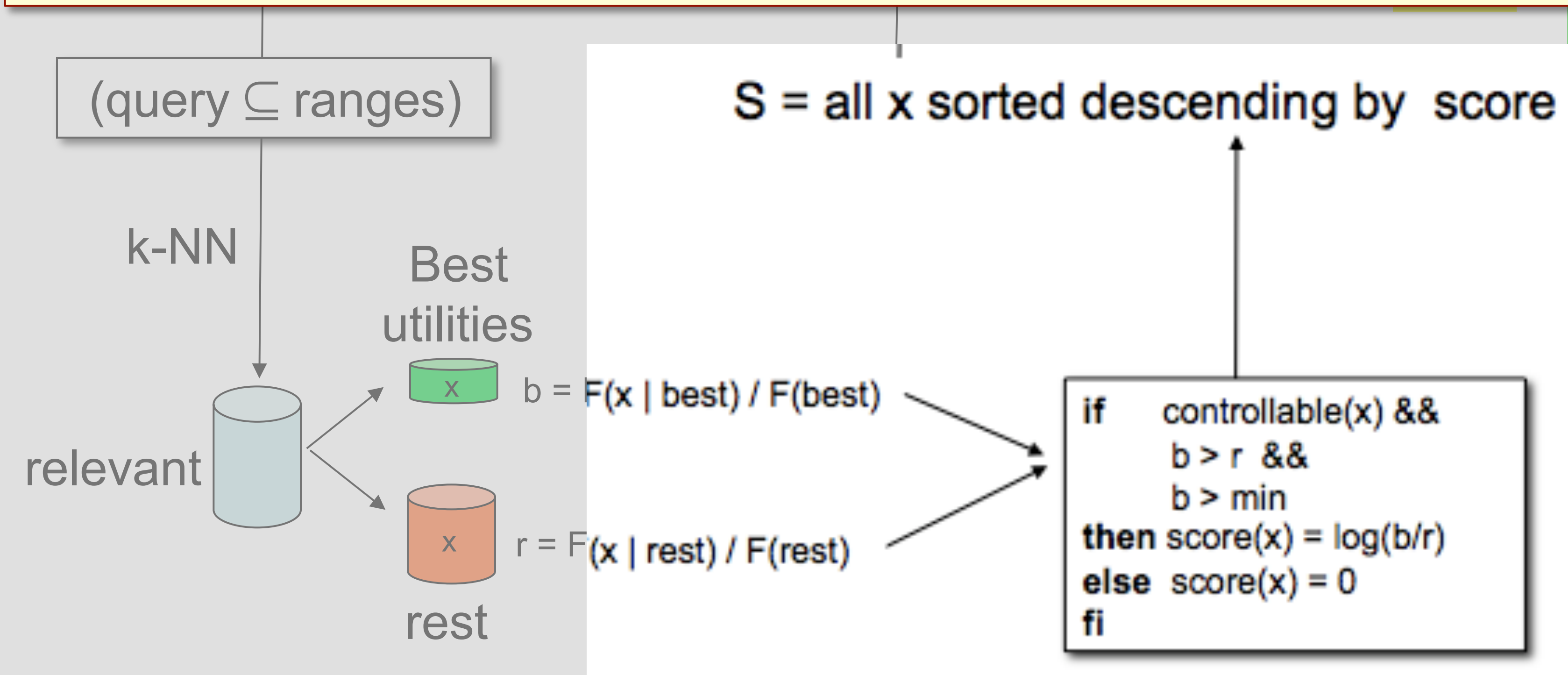


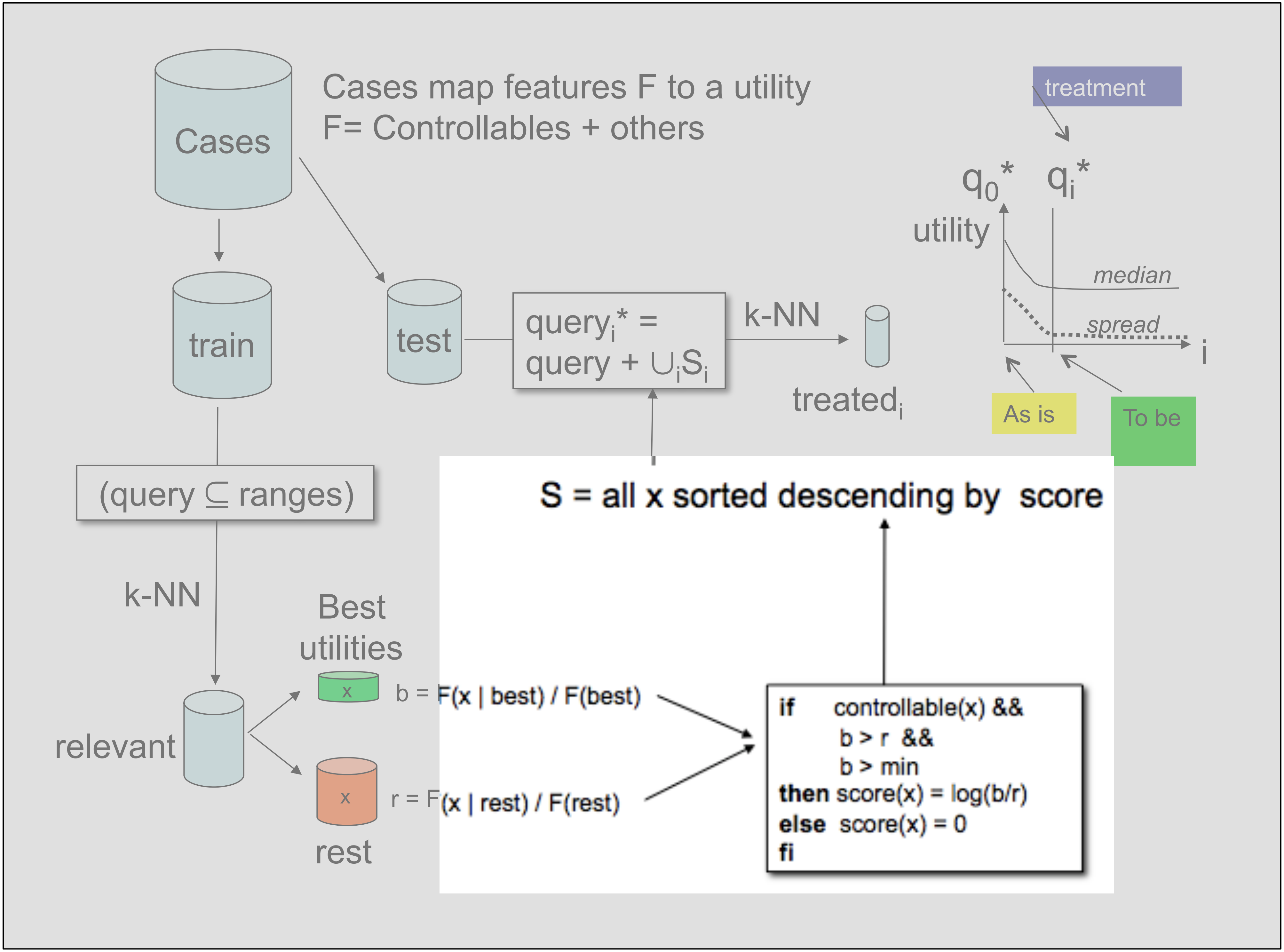
Cases
Cases map features F to a utility
F= Controllables + others
treatment
q_0^*
q_i^*
utility
median
spread
i
As is
To be
train
test
query$_i^*$ =
query + $\cup_i S_i$
k-NN
treated$_i$
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
Best
utilities
x
b = F(x | best) / F(best)
relevant
x
r = F(x | rest) / F(rest)
rest
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi


Cases
Cases map features F to a utility
F= Controllables + others
treatment
q0*
qi*
utility
median
spread
i
train
test
query_i* =
query + U_i S_i
k-NN
treated_i
As is
To be
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
Best
utilities
x
b = F(x | best) / F(best)
relevant
x
r = F(x | rest) / F(rest)
rest
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi

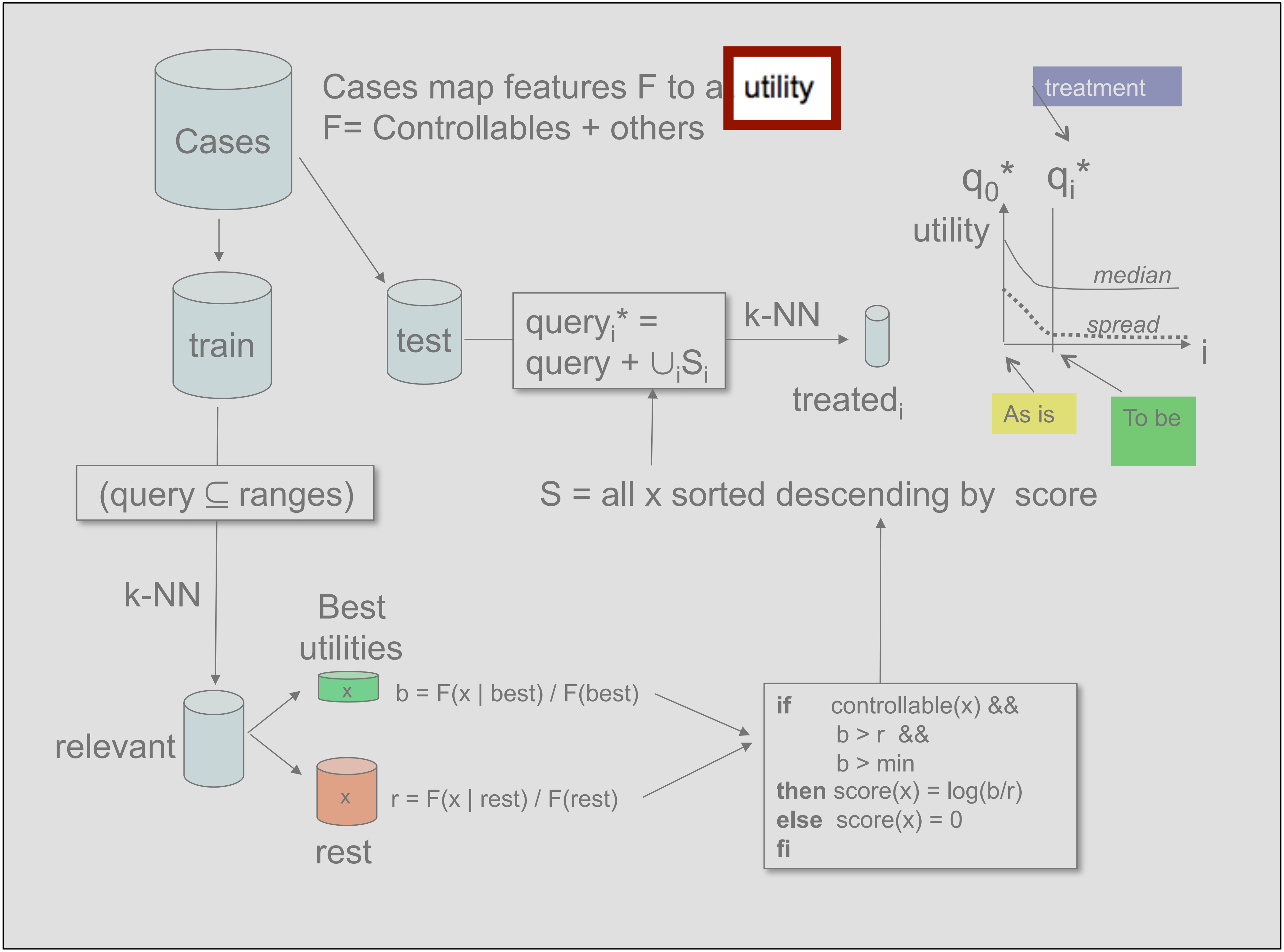
A greedy linear time search?

- Need to use much better search algorithms
- Simulated annealing, Beam, Astar, ISSAMP, MaxWalkSat
- SEESAW (home brew)





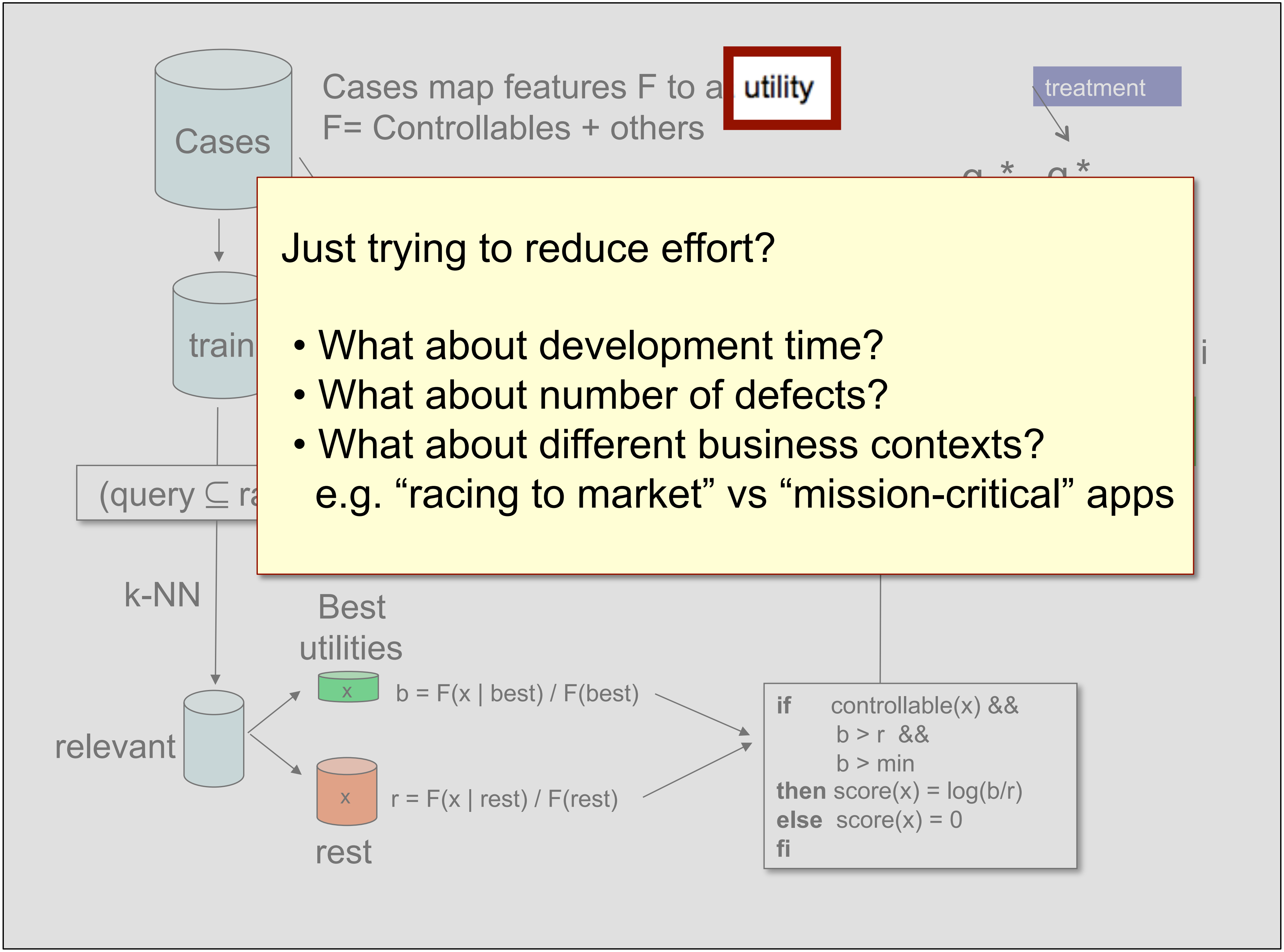
Cases
Cases map features F to a utility
F= Controllables + others
treatment
train
test
query$_i$* =
query + $\cup_i S_i$
k-NN
treated$_i$
q_0*
q_i*
utility
median
spread
i
As is
To be
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
Best
utilities
x
b = F(x | best) / F(best)
relevant
x
r = F(x | rest) / F(rest)
rest
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi

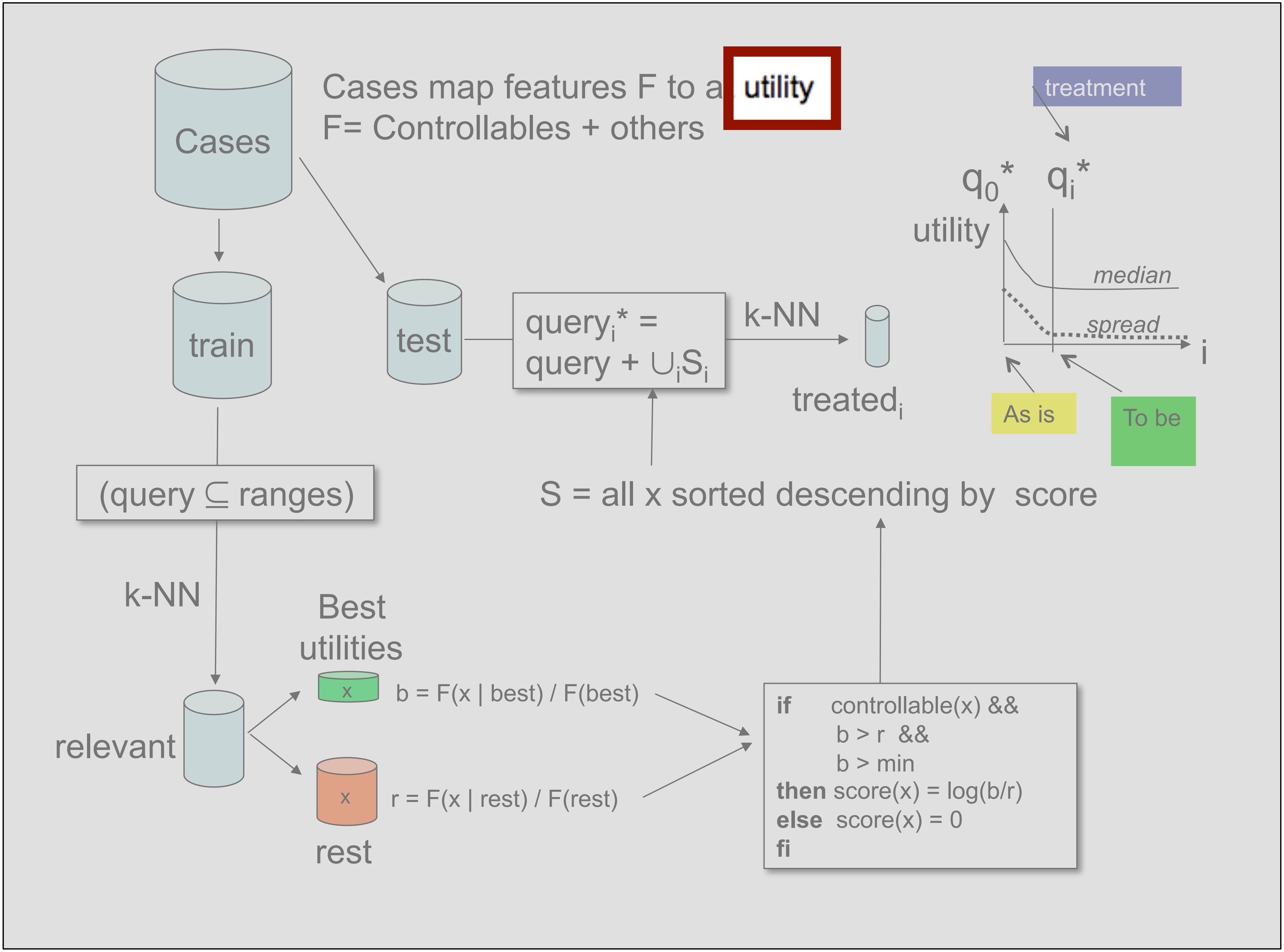


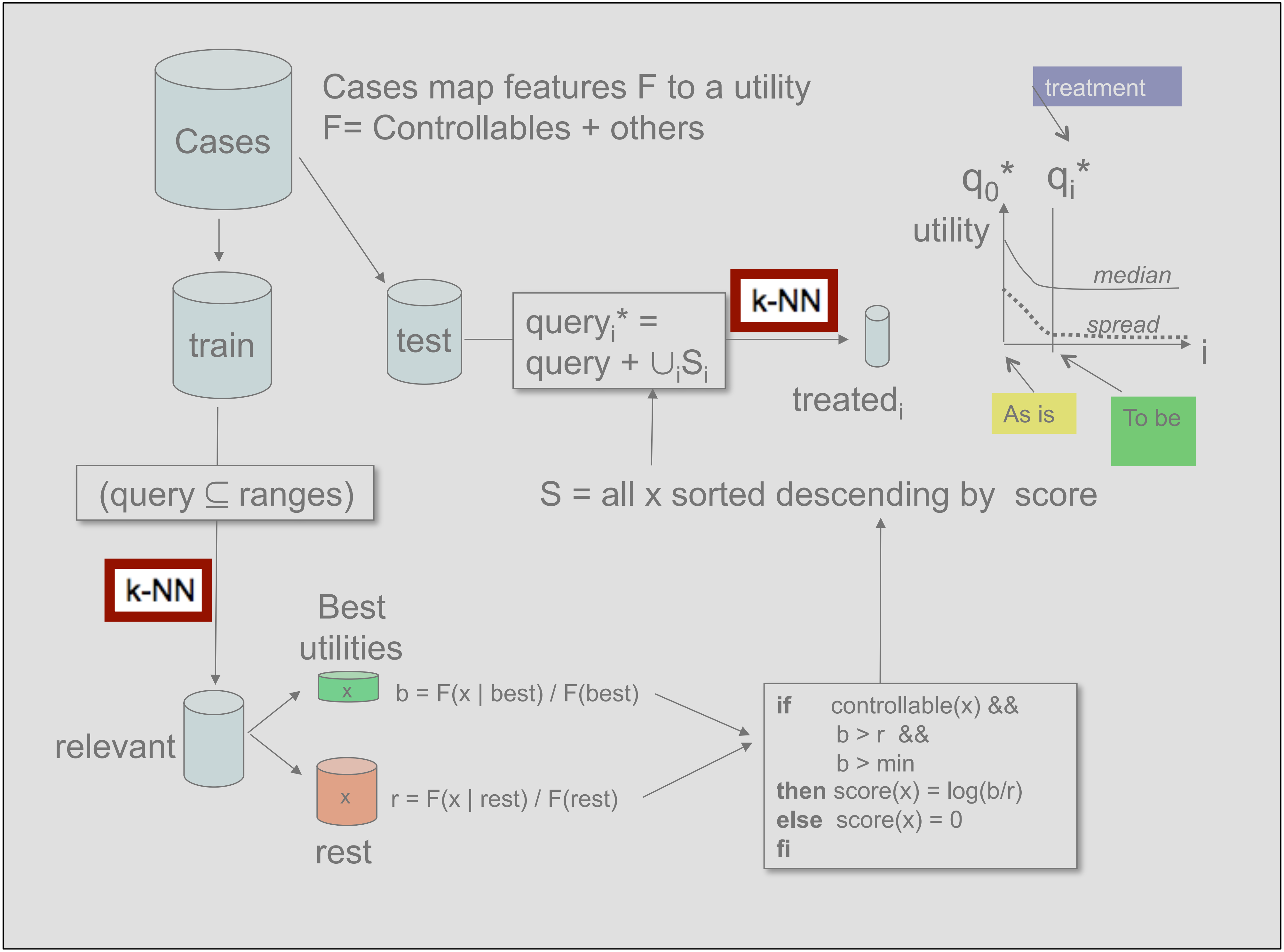
Cases map features F to a utility
F= Controllables + others
Cases
train
test
query_i* =
query + ∪_iS_i
k-NN
treated_i
treatment
q_0*
q_i*
utility
median
spread
i
As is
To be
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
Best
utilities
x
b = F(x | best) / F(best)
relevant
x
r = F(x | rest) / F(rest)
rest
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi

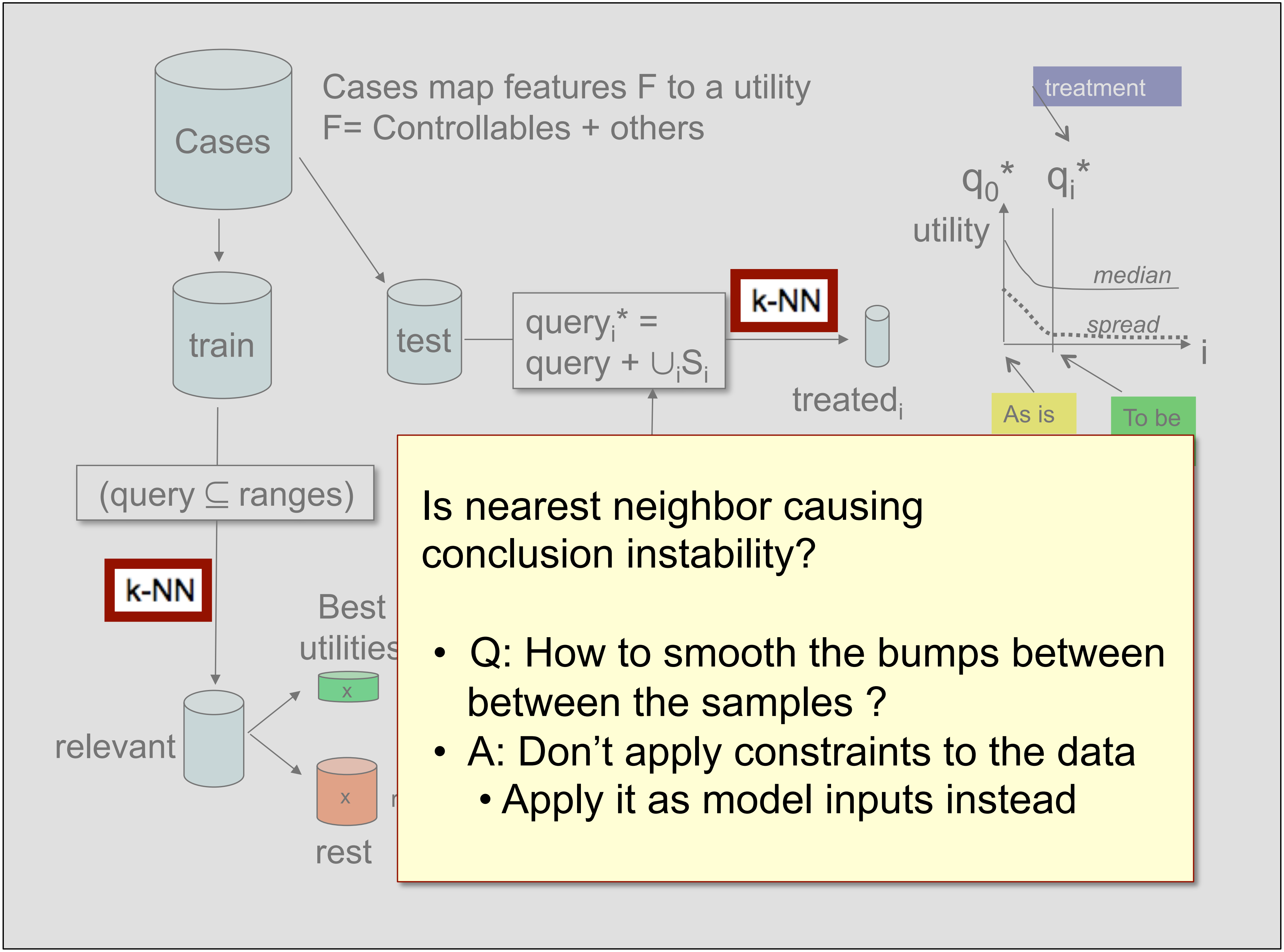
Just trying to reduce effort?

- What about development time?
- What about number of defects?
- What about different business contexts?
  e.g. “racing to market” vs “mission-critical” apps



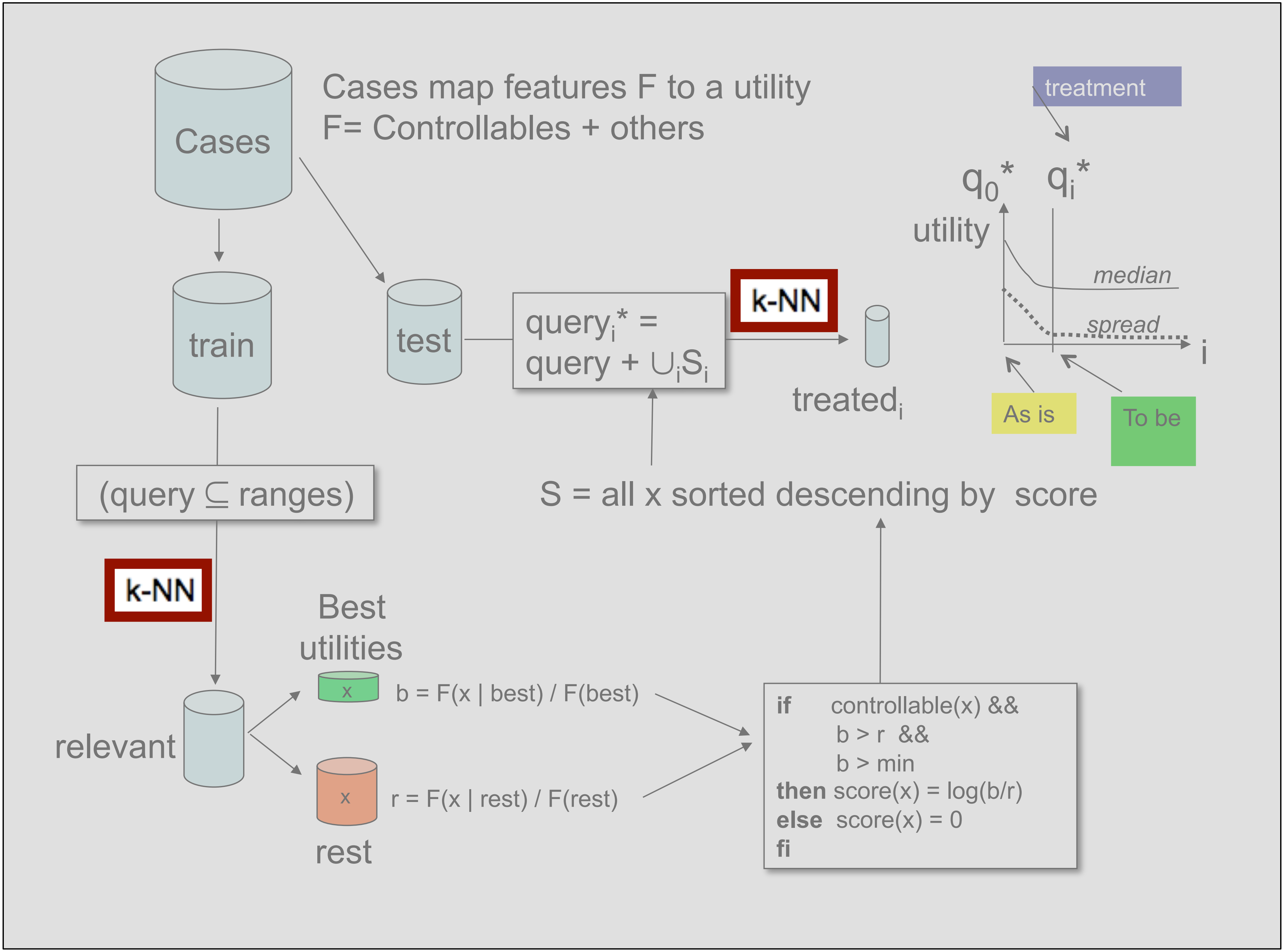

Cases
Cases map features F to a utility
F= Controllables + others
treatment
q0* qi*
utility
median
spread
i
As is
To be
train
test
query_i* = query + ∪_i S_i
k-NN
treated_i
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
Best utilities
x
b = F(x | best) / F(best)
relevant
x
r = F(x | rest) / F(rest)
rest
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi

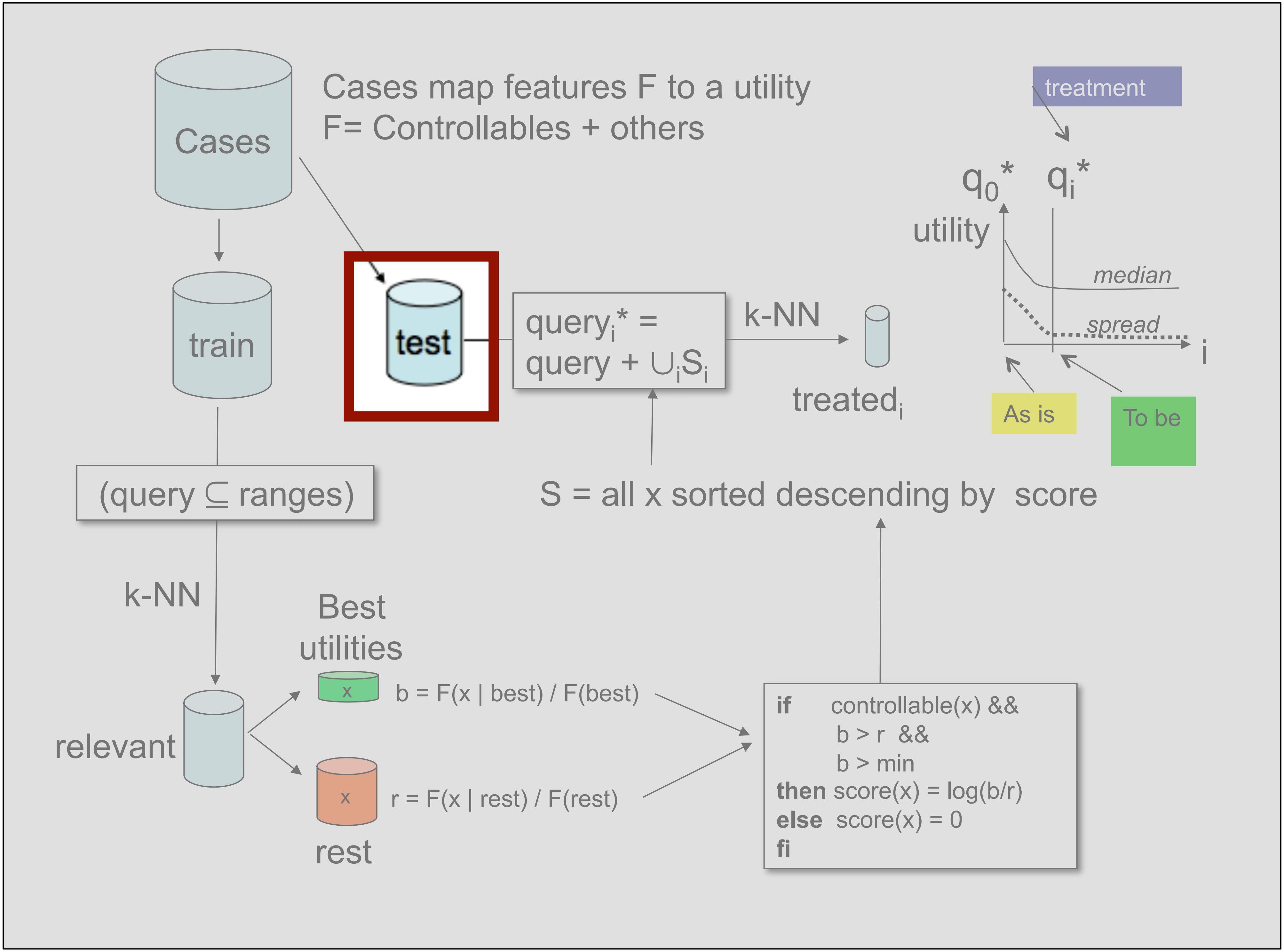

Cases
Cases map features F to a utility
F= Controllables + others
treatment
q_0*
q_i*
utility
median
spread
i
train
test
k-NN
query$_i$* =
query + $\cup_i S_i$
treated$_i$
As is
To be
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
Best
utilities
x
b = F(x | best) / F(best)
relevant
x
r = F(x | rest) / F(rest)
rest
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi



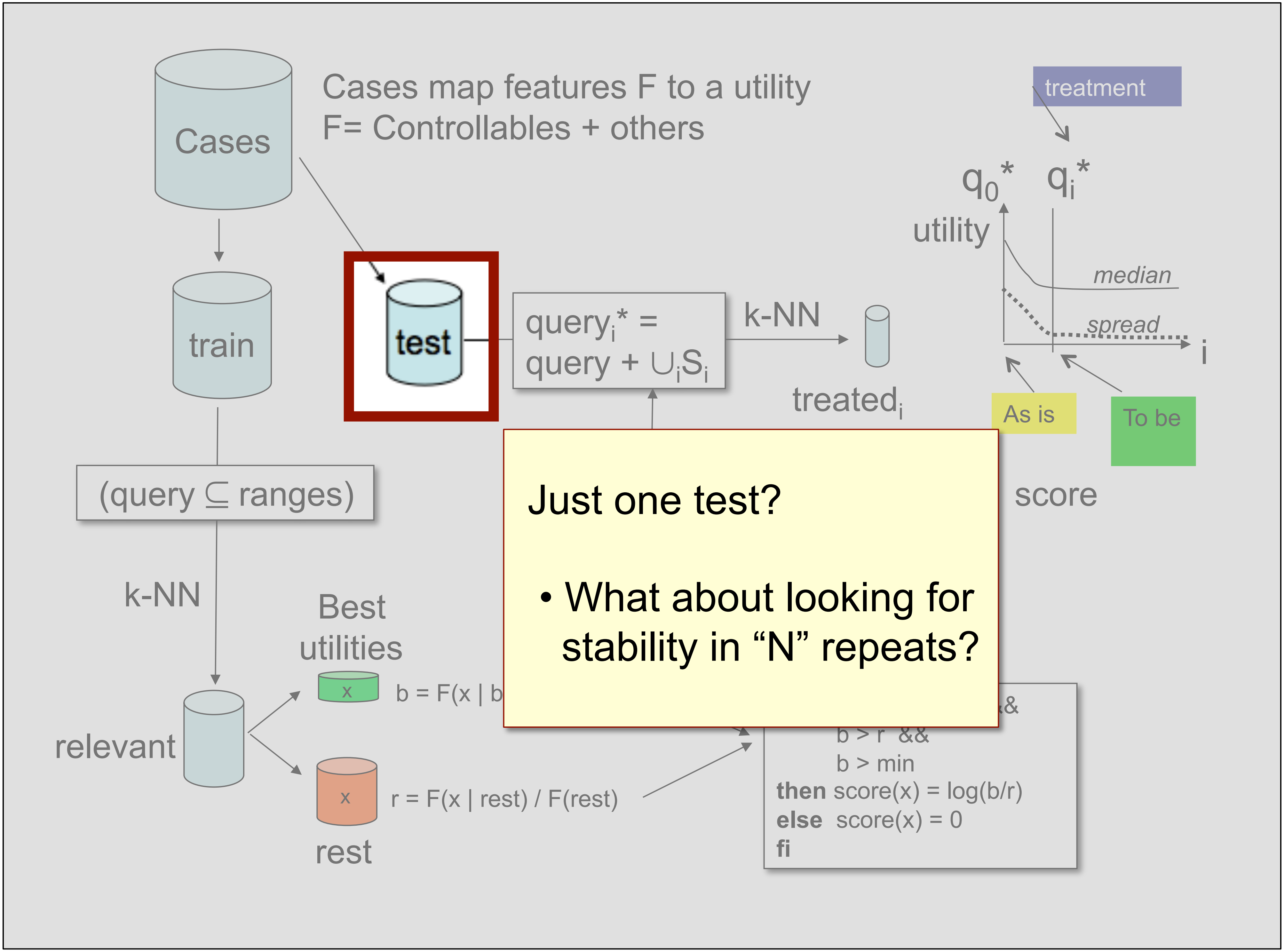
Is nearest neighbor causing conclusion instability?

- Q: How to smooth the bumps between between the samples ?
- A: Don't apply constraints to the data
  - Apply it as model inputs instead


Cases
Cases map features F to a utility
F= Controllables + others
train
test
query$_i$* =
query + $\cup_i S_i$
k-NN
treated$_i$
treatment
q$_0$*
q$_i$*
utility
median
spread
i
As is
To be
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
Best
utilities
x
b = F(x | best) / F(best)
relevant
x
r = F(x | rest) / F(rest)
rest
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi

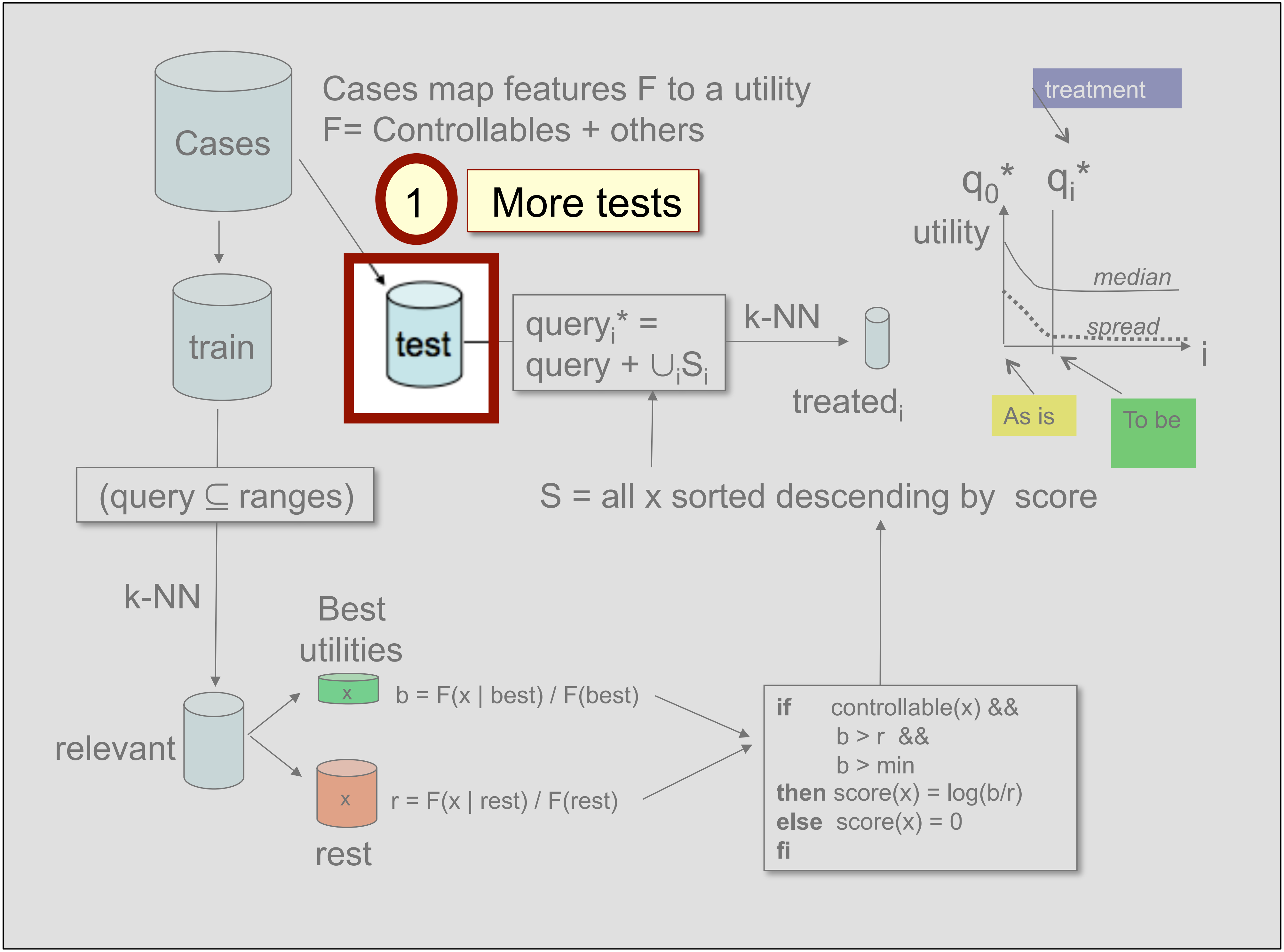


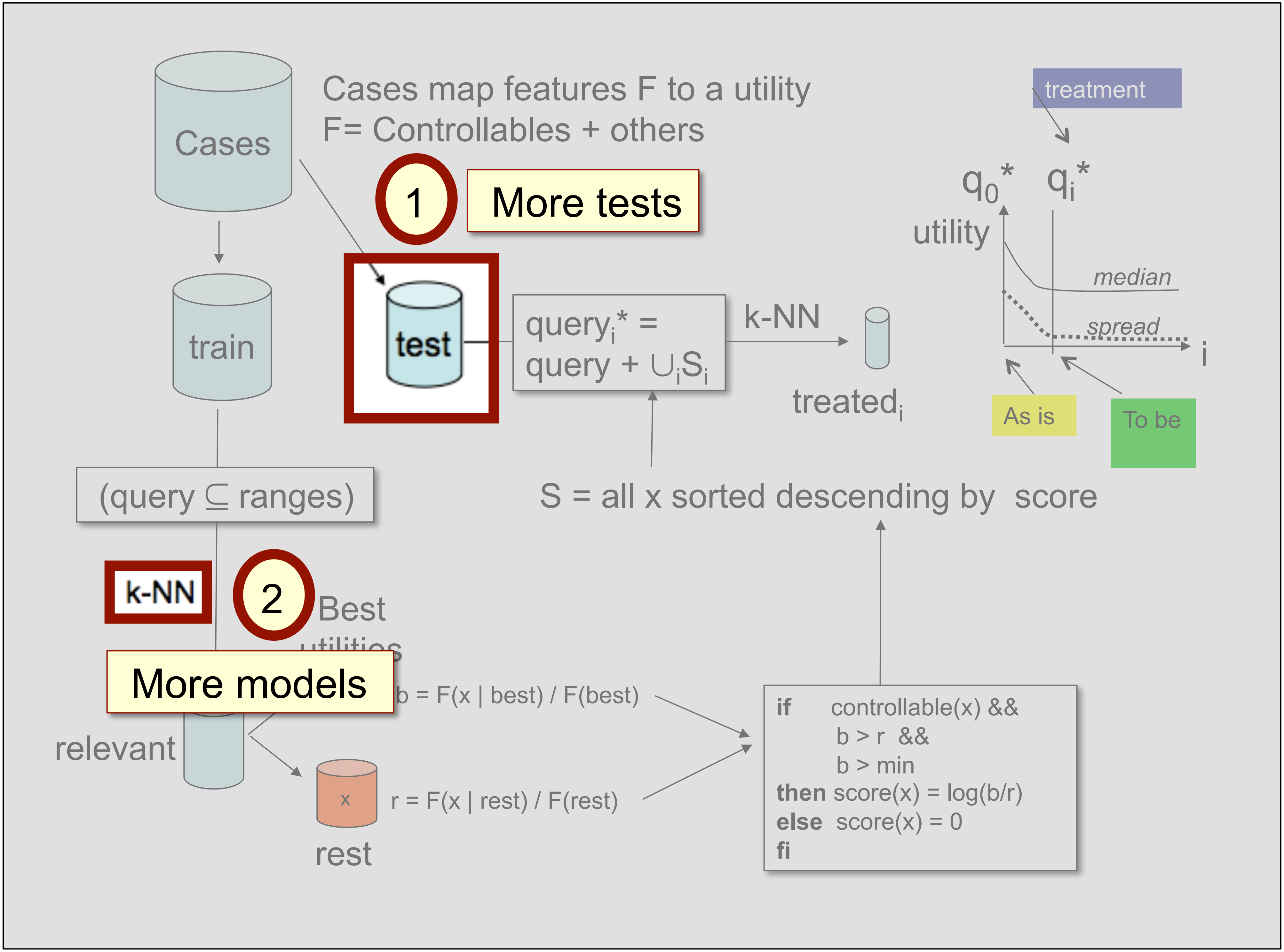
Cases
Cases map features F to a utility
F= Controllables + others
treatment
train
test
query_i* =
query + ∪_i S_i
k-NN
treated_i
q_0*
q_i*
utility
median
spread
i
As is
To be
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
Best
utilities
x
b = F(x | best) / F(best)
relevant
x
r = F(x | rest) / F(rest)
rest
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi



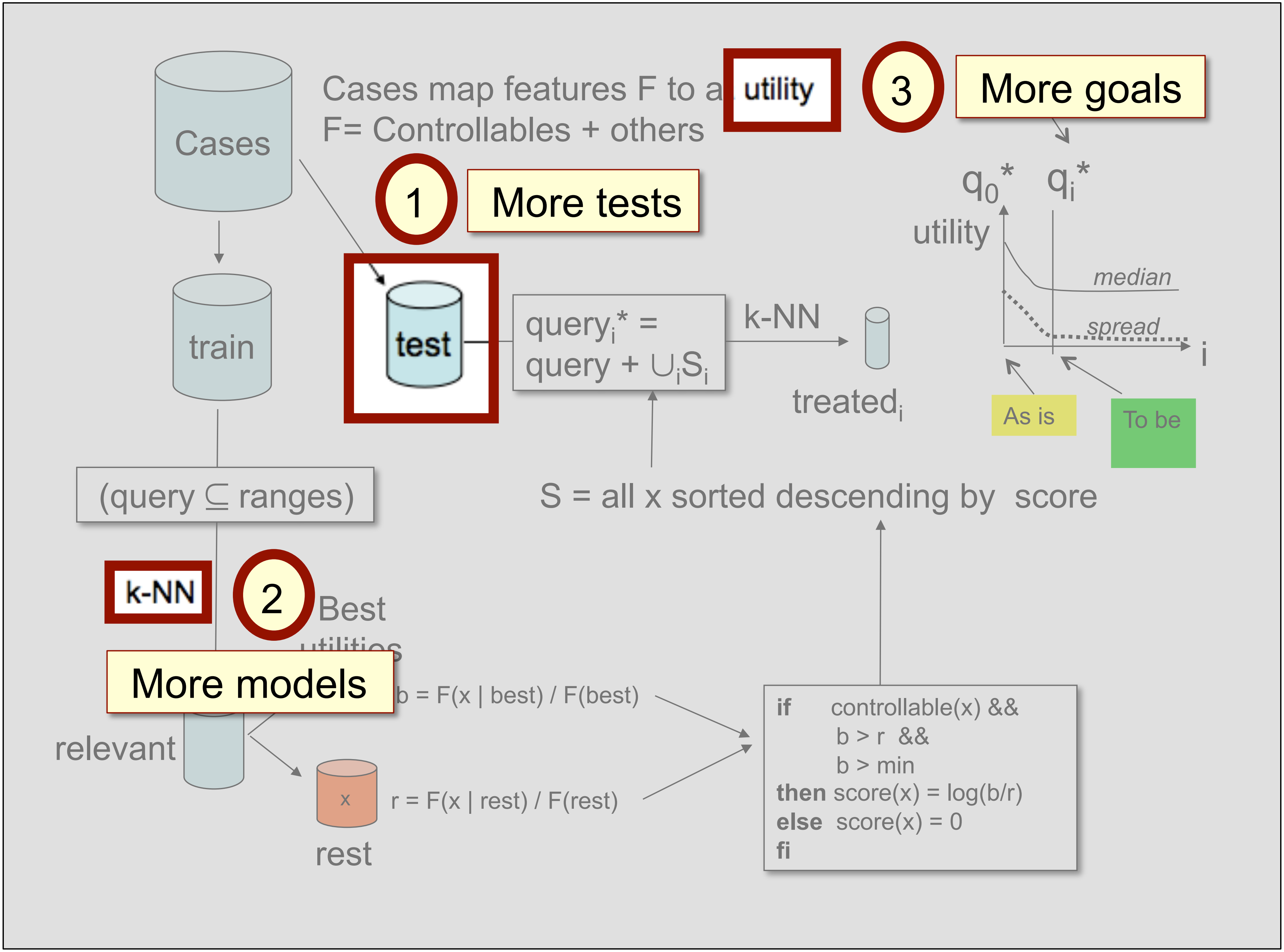
Just one test?

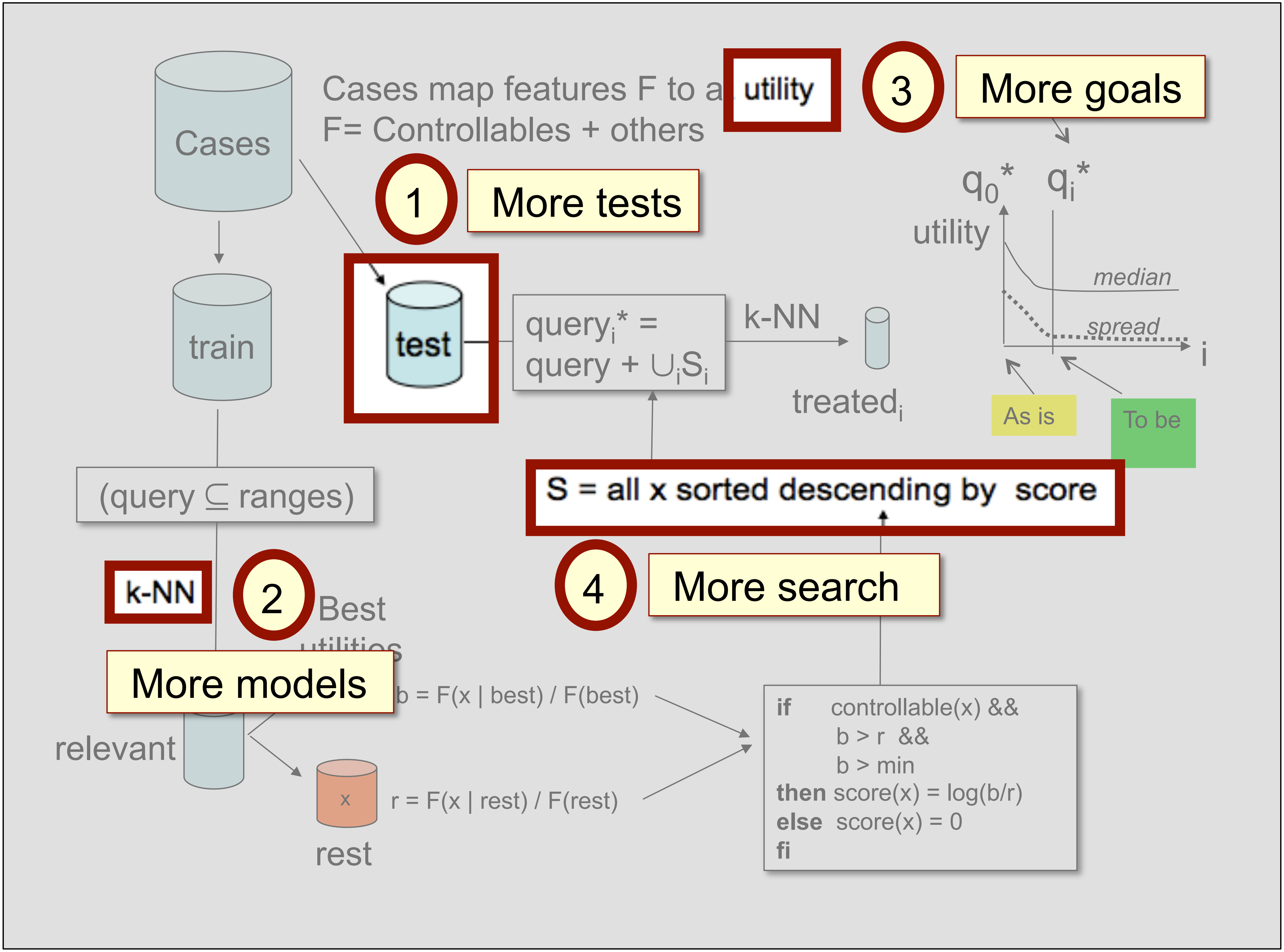
- What about looking for stability in “N” repeats?


Cases
Cases map features F to a utility
F= Controllables + others
1
More tests
treatment
q0*
qi*
utility
median
spread
i
train
test
queryi* =
query + ∪iSi
k-NN
treatedi
As is
To be
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
Best
utilities
x
b = F(x | best) / F(best)
relevant
x
r = F(x | rest) / F(rest)
rest
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi



Cases map features F to a utility
F= Controllables + others
Cases
train
test
1
More tests
treatment
q_0^*
q_i^*
utility
median
spread
i
As is
To be
$query_i^* = query + \cup_i S_i$
k-NN
$treated_i$
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
2
Best utilities
More models
b = F(x | best) / F(best)
relevant
x
rest
r = F(x | rest) / F(rest)
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi


Cases
Cases map features F to a utility
F= Controllables + others
3
More goals
1
More tests
train
test
$query_i^* = query + \cup_i S_i$
k-NN
$treated_i$
q_0^*
q_i^*
utility
median
spread
i
As is
To be
(query ⊆ ranges)
S = all x sorted descending by score
k-NN
2
Best
More models
b = F(x | best) / F(best)
relevant
x
r = F(x | rest) / F(rest)
rest
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi



Cases
Cases map features F to a utility
F= Controllables + others
3
More goals
1
More tests
train
test
query$_i$* = query + $\cup_i S_i$
k-NN
treated$_i$
q_0*
q_i*
utility
median
spread
i
As is
To be
(query ⊆ ranges)
S = all x sorted descending by score
4
More search
k-NN
2
Best utilities
More models
b = F(x | best) / F(best)
relevant
x
rest
r = F(x | rest) / F(rest)
if controllable(x) &&
b > r &&
b > min
then score(x) = log(b/r)
else score(x) = 0
fi

Roadmap

- Motivation: generality in SE
- A little primer: DM for SE
- “W”: finding contrast sets
- “W”: case studies
- “W”: drawbacks
- “NOVA”: a better “W”
- Conclusions

More models

USC Cocomo suite (Boehm 1981, 2000)

COCOMO
- Time to build it (calendar months)
- Effort to build it (total staff months)

COQUALMO
- defects per 1000 lines of code

Estimate = model(p, t)
- P = project options
- T = tuning options
- Normal practice: Adjust “t” using local data
- NOVA: Stagger randomly all tunings even seen before

?

$$\arg\max_{x} \left(\overbrace{r_x \subseteq p}^{AI\ search}, \underbrace{t \subseteq T, value(model(r_x, t))}_{Monte\ Carlo} \right)$$

More goals

B = BFC

Goal #1:

- better, faster, cheaper

Try to minimize:

- Development time _and_
- Development effort _and_
- # defects

X = XPOS

Goal #2

- minimize risk exposure

Rushing to beat the competition

- Get to market, soon as you can
- Without too many defects

More search engines

Not greedy search

Simulated Annealling

ISSAMP

ASTAR

BEAM

MaxWalkSat

SEESAW : MaxWalkSat + boundary mutation
- Local favorite
- Does best at reduction defects or effort or time

More tests

Four data sets, repeat N=20 times

ranges			fixed settings	
feature	low	high	feature	setting
prec	1	2	data	3
flex	2	5	pvol	2
resl	1	3	rely	5
team	2	3	pcap	3
pmat	1	4	plex	3
stor	3	5	site	3
ruse	2	4		
docu	2	4		
acap	2	3		
pcon	2	3		
apex	2	3		
ltex	2	4		
tool	2	3		
sced	1	3		
cplx	5	6		
KSLOC	75	125		

Data sets
- OSP= orbital space plane GNC
- OSP2 = second generation GNC
- Flight = JPL flight systems
- Ground = JPL ground systems

For each data set
- Search N= 20 times (with SEESAW)
- Record how often decisions are found

Frequency% of range in 20 repeats

(ignore all ranges found < 50%)

Better, faster, cheaper

Minimize risk exposure (rushing to market)

Data	Range	value: B=BFC	value: X=XPOS	$\frac{B}{B+X}$
ground	rely = 4	70	20	77
	aa = 6	70	25	73
	resl = 6	65	40	61
	etat = 1	35	65	35
	aexp = 5	45	85	34
	pr = 1	35	80	30
	aa = 1	25	60	29
	data = 2	25	70	26
	rely = 1	15	70	17
flight	rely = 5	65	25	72
	flex = 6	80	50	61
	docu = 1	55	85	39
	site = 6	55	85	39
	resl = 6	45	70	39
	pr = 1	45	70	39
	pvol = 2	45	75	37
	data = 2	35	60	36
	cplx = 3	45	90	33
	rely = 3	15	60	20
OSP	pmat = 4	85	45	65
	resl = 3	45	70	39
	ruse = 2	40	65	38
	docu = 2	25	90	21
OSP2	sced = 2	100	0	100
	sced = 4	0	80	0

If high, then more in BFC

If 50% then same In BFC and XPOS

If low, then usually in XPOS

Mostly: if selected by one, rejected by the other

"Value" (business context) changes everything

And what of defect removal techniques?

Aa = automated analysis
Etat= execution testing and tools
Pr= peer review

Better, faster, cheaper

Minimize risk exposure (rushing to market)

Data	Range	*value* B=BFC	X=XPOS	$\frac{B}{B+X}$
ground	rely = 4	70	20	77
	aa = 6	70	25	73
	resl = 6	65	40	61
	etat = 1	35	65	35
	aexp = 5	45	85	34
	pr = 1	35	80	30
	aa = 1	25	60	29
	data = 2	25	70	26
	rely = 1	15	70	17
flight	rely = 5	65	25	72
	flex = 6	80	50	61
	docu = 1	55	85	39
	site = 6	55	85	39
	resl = 6	45	70	39
	pr = 1	45	70	39
	pvol = 2	45	75	37
	data = 2	35	60	36
	cplx = 3	45	90	33
	rely = 3	15	60	20
OSP	pmat = 4	85	45	65
	resl = 3	45	70	39
	ruse = 2	40	65	38
	docu = 2	25	90	21
OSP2	sced = 2	100	0	100
	sced = 4	0	80	0

Stopping defect introduction is better than defect removal.

Roadmap

- Motivation: generality in SE
- A little primer: DM for SE
- “W”: finding contrast sets
- “W”: case studies
- “W”: drawbacks
- “NOVA”: a better “W”
- Conclusions

Certainly, we should always strive for generality

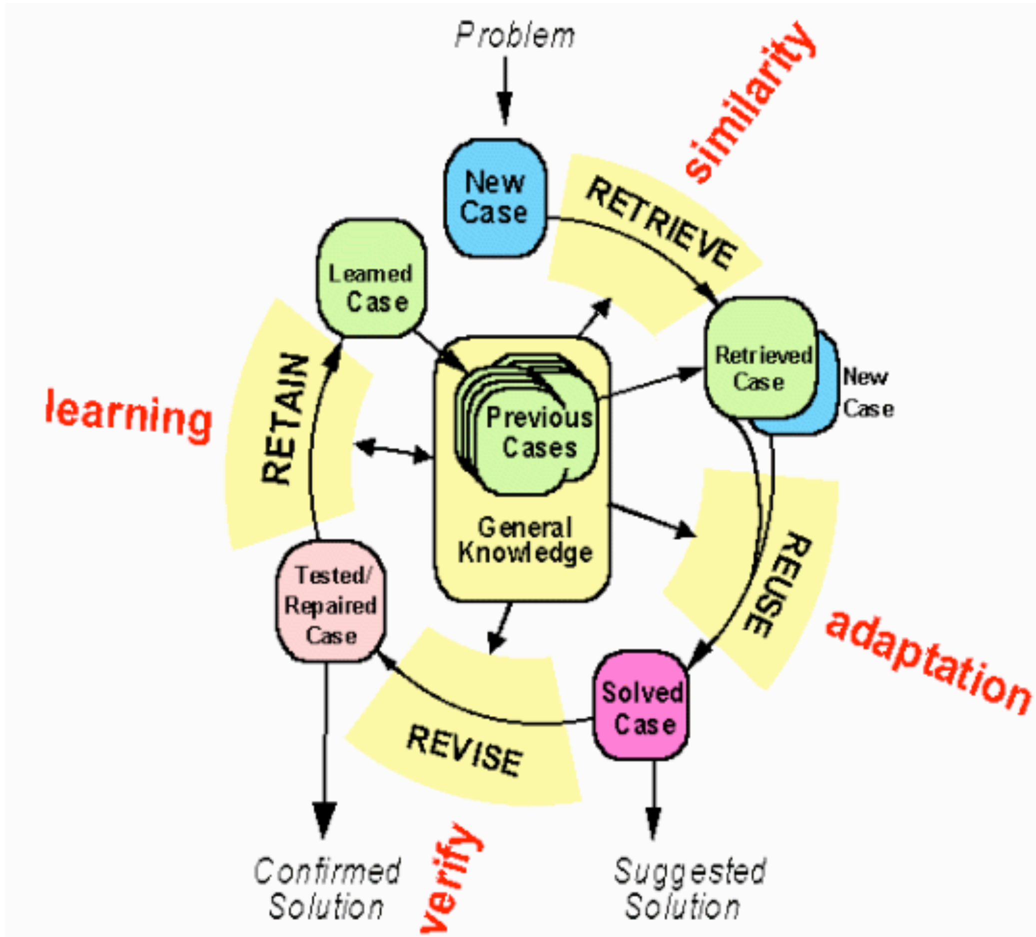
But don’t be alarmed if you can’t find it

- The experience to date is that,
  - with rare exceptions,
  - W and NOVA do not lead to general theories
- But that’s ok
  - Very few others have found general models (in SE)
  - E.g. Turhan, Menzies, Ayse’09
- Anyway
  - If there are few general results, there may be general methods to find local results

Btw, constantly (re)building local models is a general model

Case-based reasoning

- Kolodner's theory of reconstructive memory
- The Yale group
  - Shank & Riesbeck et al.
  - Memory, not models
  - Don't "think", remember



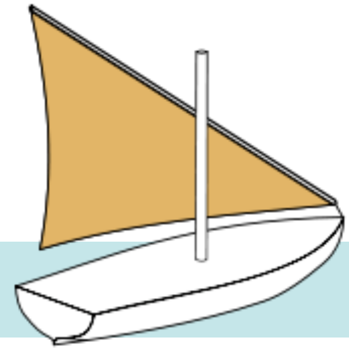
See you at PROMISE'10?

PROMISE

The 2010 International Conference on Predictive Models in Software Engineering

Supplemental slides

Contact details



We know where you live.

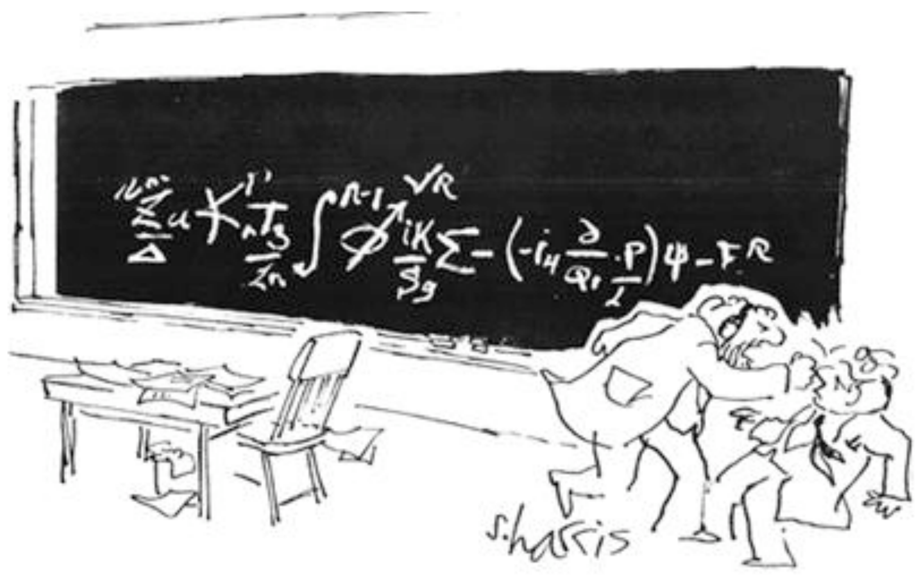
tim@menzies.us

http://menzies.us

http://twitter.com/timmenzies

http://www.facebook.com/tim.menzies

Questions?
Comments?

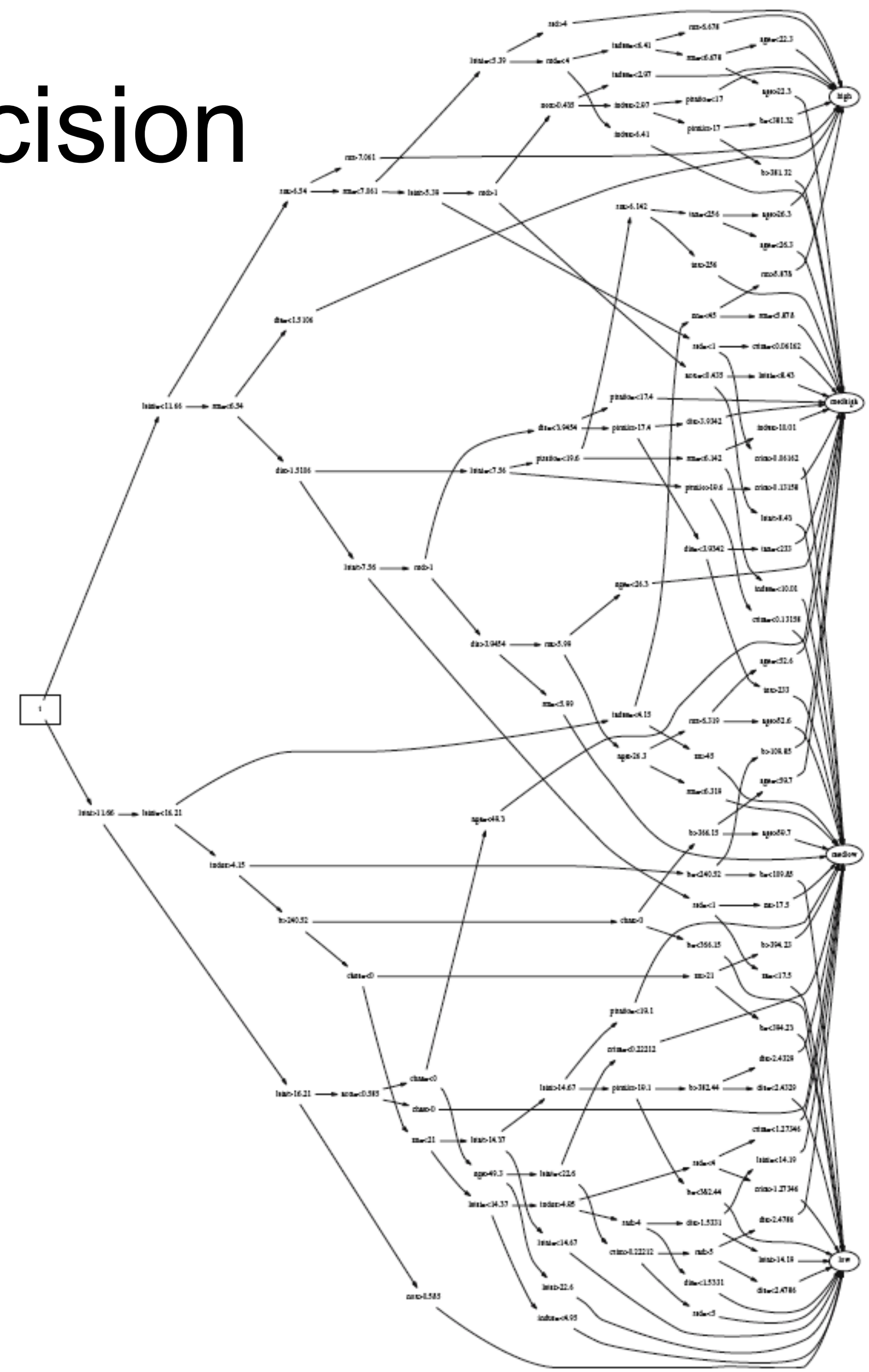


"You want proof? I'll give you proof!"

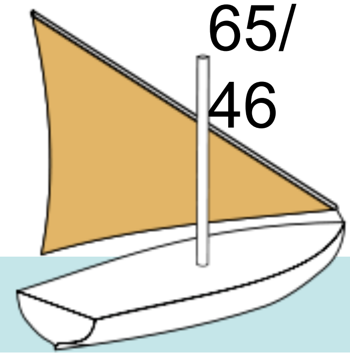
Monte Carlo + Decision Tree Learning

Menzies: ASE'00

- Process models
  - Input: project details
  - Output: (effort, risk)
- Increase #simulations
  - till error minimizes
- Learn decision trees
- Repeat 10 times



The “keys” effect: usually, a few variables set the rest

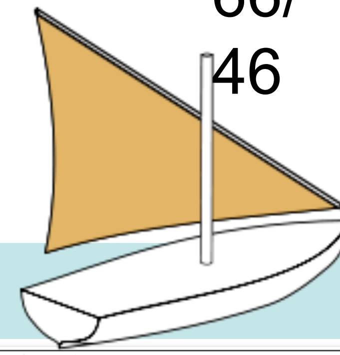


SAILing is easy

- So the complexity of the whole depends on just a small part
- Empirical evidence:
  - Feature subset selection: Kohavi’97
  - Few pathways: Bieman’92, Harrold’98
  - Mutation testing & rapid saturation: Budd’80, Wong’95, Michael’97
  - Surprisingly few internal states: Drezdel’94, Colomb’00, Menzies’99
  - Success of stochastic theorem provers: Crawford’94, Williams & Selman’03
- Theoretical evidence:
  - Menzies & Singh ‘03

- Easy to find these keys
  - Score the outputs
  - Look for ranges more frequent in “best” than “rest”
  - A useful short-cut to data mining, model-based reasoning

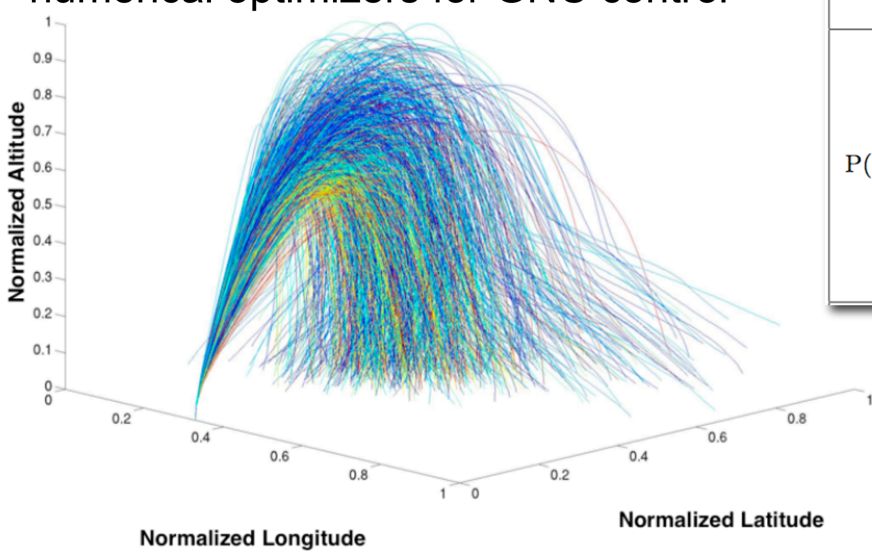
Treatment learning: 9 years later



Gay, Menzies et al.' 09

- TARZAN is no longer a post-processor
  - Branch queries performed directly on discretized data
    - thanks David Poole
  - Stochastic sampling for rule generation

- Benchmarked against state-of-the-art numerical optimizers for GNC control



Metric	Project 1			
	Rank	Program	50%	
Runtime	1	TAR4.1	0.13	
	2	TAR3	0.31	
	3	QN	6	
	4	SA-T4	15	
	4	SA-T3	16	
	Rank	Program	50%	Quartiles (0, 50, 100)
Recall	1	TAR4.1	59	
	1	QN	36	
	2	SA-T4	25	
	3	TAR3	22	
	4	SA-T3	20	
	Rank	Program	50%	Quartiles (0, 50, 100)
P(False Alarm)	1	TAR3	1	
	2	SA-T3	9	
	3	TAR4.1	25	
	4	QN	34	
	4	SA-T4	71	

Still generating tiny rules
(very easy to read, explain, audit, implement)