Richard Feynman:

"It is imperative in science to doubt; it is absolutely necessary, for progress in science, to have uncertainty as a fundamental part of your inner nature."

"The FEYNMAN Problem-Solving Algorithm:
i) write down the problem;
ii) think very hard;
iii) write down the answer."

Model-Based vs. Method-Based Decisioning

Model-Based Approach

- An analyst focuses only on modeling the problem of interest (WHAT)
- The determination of a decision(s) is left to general-purpose, off-the-shelf decision engine (HOW)

Method-Based Approach

- Along with modeling the problem (WHAT) an analyst also specifies HOW to determine a decision(s) in different situations
- The decision engine executes a decision determination algorithm specified by an analyst

Example: Vacation Days

- Problem Description

The number of vacation days depends on age and years of service.

Every employee receives at least 22 days.
Additional days are provided according to the following criteria:

1) Only employees younger than 18 or at least 60 years, or employees with at least 30 years of service will receive 5 extra days.

2) Employees with at least 30 years of service and also employees of age 60 or more, receive 3 extra days, on top of possible additional days already given.

3) If an employee has at least 15 but less than 30 years of service, 2 extra days are given. These 2 days are also provided for employees of age 45 or more. These 2 extra days can not be combined with the 5 extra days.

Method-Based Approach

An analyst should define all possible combinations and exceptions:

DecisionTableMultiHit CalculateVacationDays				
If	If	If	Conclusion	
Eligible for Extra 5 Days	Eligible for Extra 3 Days	Eligible for Extra 2 Days	Vacation Days	
			=	22
TRUE			+=	5
	TRUE		+=	3
FALSE		TRUE	+=	2

DecisionTable SetEligibleForExtra5Days		
If	If	Then
Age in Years	Years of Service	Eligible for Extra 5 Days
< 18		TRUE
>= 60		TRUE
	>= 30	TRUE
		FALSE

DecisionTable SetEligibleForExtra3Days		
If	If	Then
Age in Years	Years of Service	Eligible for Extra 3 Days
	>= 30	TRUE
>= 60		TRUE
		FALSE

DecisionTable SetEligibleForExtra2Days		
If	If	Then
Age in Years	Years of Service	Eligible for Extra 2 Days
	[15..30)	TRUE
>= 45		TRUE
		FALSE

- Now imagine that we want also to give 4 extra days to "veterans" and to people who worked more than 35 years.
- However, the total number of vacation days should not exceed 29 days
- How many exceptions on top of exceptions should we define?
- The exception logic hides the main intention (<=29 days) and quickly become unmanageable

Model-Based Approach

For every Type of Extra Days define

$x_t \in \{0,1\}$ = 1 if an employee is given extra days of the type *t*

= 0 if otherwise

$\text{Total}_e = 22 + 5*x_{t1} + 3*x_{t2} + 2*x_{t3} + 4*x_{t4}$

Define total vacation days as

$\text{Total}_e = baseDays + \sum_t extraDays_t * xt$

Subject to

$\text{Total}_e <= 29$

Maximize

Total_e => Max

This model replaces all rules about relationships between different types of extra days and replaces a "greedy" one path to solution to an ability to find an optimal solution

Business rules without exceptions

DecisionTable SetEligibleForExtra5Days		
If	If	Then
Age in Years	**Years of Service**	**Eligible for Extra 5 Days**
< 18		TRUE
>= 60		TRUE
	>= 30	TRUE
		FALSE

DecisionTable SetEligibleForExtra3Days		
If	If	Then
Age in Years	**Years of Service**	**Eligible for Extra 3 Days**
	>= 30	TRUE
>= 60		TRUE
		FALSE

DecisionTable SetEligibleForExtra2Days		
If	If	Then
Age in Years	**Years of Service**	**Eligible for Extra 2 Days**
	[15..30)	TRUE
>= 45		TRUE
		FALSE

Example: Flight Rebooking

- *A flight was cancelled, and we need to re-book passengers to other flights considering their frequent flyer status, miles, and seat availability.*
- *Here is a sample data and flight assignment rules:*
- Flights:

Flight	From	To	Dep	Arr	Capacity	Status
UA123	SFO	SNA	1/1/07 6:00 PM	1/1/07 7:00 PM	5	cancelled
UA456	SFO	SNA	1/1/07 7:00 PM	1/1/07 8:00 PM	2	scheduled
UA789	SFO	SNA	1/1/07 9:00 PM	1/1/07 11:00 PM	2	scheduled
UA1001	SFO	SNA	1/1/07 11:00 PM	1/2/07 5:00 AM	0	scheduled
UA1111	SFO	LAX	1/1/07 11:00 PM	1/2/07 5:00 AM	2	scheduled

- Passengers:

Name	Status	Miles	Flight
Jenny	gold	500000	UA123
Harry	gold	100000	UA123
Igor	gold	50000	UA123
Dick	silver	100	UA123
Tom	bronze	10	UA123

Example: Flight Rebooking

- Rules:

1. Alternate flight must depart from the same place as the cancelled flight
2. Alternate flight must arrive at the same place as the cancelled flight
3. Alternate flight must depart after the cancelled flight
4. There must be room on the alternate flight
5. Passenger status determines who gets allocated first

- Possible Results:

Jenny is confirmed on UA456 departing SFO at 1/1/2007 19:00:00 arriving SNA at 1/1/2007 20:00:00
Harry is confirmed on UA456 departing SFO at 1/1/2007 19:00:00 arriving SNA at 1/1/2007 20:00:00
Igor is confirmed on UA789 departing SFO at 1/1/2007 21:00:00 arriving SNA at 1/1/2007 23:00:00
Dick is confirmed on UA789 departing SFO at 1/1/2007 21:00:00 arriving SNA at 1/1/2007 23:00:00
Tom could not be rebooked

Method-Based Approach

Defines a greedy algorithm to build passenger-flight assignments:

1. First, sort all passengers using their GOLD, SILVER or BRONZE status. If two passengers have the same status use miles as a tiebreaker.
2. Repeat for every passenger from the sorted list:
   - Build a list of "suitable flight" for the selected passenger. A "suitable" flight should have the same departure and arrival airports as the cancelled flight and it also should still have an available seat
   - Sort the flights inside this list by an earlier departure time
   - Assign the flight on the top of the list to the current passenger
   - Decrement the flight's capacity

Implementation example using DMN boxed expressions

(from DMCommunity.org Challenge)

rebooking		
(unbooked *(tBookingList)*, rebooked *(tBookingList)*, fList *(tFList)*, originalFList *(tFList)*)		
thePassenger *(tPassenger)*	unbooked[1]	
originalFlight *(Text)*	originalFList[fnum=thePassenger.flight]	
originalDepart *(Date and time)*	originalFlight.depart	
theDestination *(Text)*	originalFlight.to	
availableFlights *(tFList)*	fList[status="scheduled" and to=theDestination and seatsAvailable!=0]	
isFlightAvailable *(Boolean)*	if count(availableFlights)>0 then true else false	
firstArrival *(Date and time)*	min(availableFlights.date and time(arrive))	
bookedFlight *(tFlight)*	availableFlights[arrive=firstArrival]	
newBooking *(tBooking)*	name *(Text)*	thePassenger.name
	flight *(Text)*	if isFlightAvailable=true then bookedFlight.fnum else "none"
	arrive *(Date and time)*	if isFlightAvailable=true then firstArrival else "-"
newRebooked *(tBookingList)*	append(rebooked,newBooking)	
newUnbooked *(tBookingList)*	remove(unbooked,1)	
newFlightList *(tFList)*	for i in availableFlights return newFlight(i,bookedFlight)	
bookings *(tBookingList)*	if count(newUnbooked)>0 then rebooking(newUnbooked,newRebooked,newFlightList) else newRebooked	
bookings		

Implementation example

Method-Based Approach

DecisionTableSort SortPassengers
Array of Objects
Passengers

DecisionTable ComparePassengers							
Condition		Condition		Condition		Action	Action
Passenger 1 Status		Passenger 2 Status		Passenger 1 Miles		Passenger 1 Score	Passenger 2 Score
Is	GOLD	Is One Of	SILVER, BRONZE			1	0
Is		Is	GOLD	>	Passenger 2 Miles	1	0
Is		Is		<	Passenger 2 Miles	0	1
Is		Is		=	Passenger 2 Miles	1	1
Is	SILVER	Is	GOLD			0	1
Is		Is	BRONZE			1	0
Is		Is	SILVER	>	Passenger 2 Miles	1	0
Is		Is		<	Passenger 2 Miles	0	1
Is		Is		=	Passenger 2 Miles	1	1
Is	BRONZE	Is One Of	GOLD,SILVER			0	1
Is		Is	BRONZE	>	Passenger 2 Miles	1	0
Is		Is		<	Passenger 2 Miles	0	1
Is		Is		=	Passenger 2 Miles	1	1

Plus two embedded loops

Model-Based Approach

DecisionTableMultiHit CalculatePenalties			
If	If	Conclusion	
Passenger Status	Passenger Miles	Passenger Penalty For Delayed Hour	
		=	10
GOLD		+=	15
SILVER		+=	8
BRONZE		+=	5
	500000+	+=	4
	[250000..500000)	+=	3
	[100000..250000)	+=	2
	[25000..100000)	+=	1

Model-Based Approach

Given

F set of flights

P set of passengers from the canceled flight

For every passenger $p \in P$ and flight $f \in F$ Determine

$x_{pf} \in \{0,1\}$ = 1 if passenger *p* is assigned to flight *f*

= 0 if otherwise

$delay_{pf}$ = number of hours between arrivals of the flight *f* and the passenger *p*'s canceled flight

= 100 if the passenger *p* is assigned to not scheduled flight *f*

$penalty_{pf}$ = $delay_{pf}$* $penaltyPerDelayedHour_p$

Subject to constraints

Each Passenger can be assigned to no more than 1 flight:

$x_{pf1} + x_{pf2} + \ldots + x_{pfn} <= 1$ for each passenger *p*

Number of passengers assigned to the same flight cannot exceed the flight's capacity

$x_{fp1} + x_{fp2} + \ldots + x_{fpn} <= f_{capacity}$ for each flight *f*

Minimize

$\sum_{p \in P} \sum_{f \in F} (penalty_{pf} * x_{pf})$ => MIN

Model-Based Approach

- New decision variables "penalties for delayed hours"
- Last night I did a quick implementation of this model
- I used the above decision table "CalculatePenalties" with OpenRules and JSR-331
- It quickly produced the same decision

- Benefits of the new model:
  - Removes programming – now we have no loops, sorting, filters, or recursion created by a user
  - Instead of one "greedy" decision it allows to analyze different decision and find an optimal one

Model-Based Approach in the Loan Origination Domain

- Don't simply decline a loan application with a fixed loan amount and rate based on a "greedy" risk management algorithm written in rules
- Instead, build a much smarter decision model capable to offer the best possible combination of the amount and rate while controlling risk
- Example
  - A borrower requested $50K education loan with a certain rate. Traditional decision model may reject this application with explanation which rules caused the rejection.
  - But a smarter decision model without adding additional rules may offer a $48.5K loan with a slightly different rate

Model-Based Approach in the Insurance Domain

- **Insurance Example**
  - Traditionally insurers define writing rules that cover different combinations of eligible discounts
  - A smarter approach:
    - Give a customer as many discounts as possible
    - While limiting the total amount of discounts as a percent of the premium

DecisionCAMP-2019

- Preliminary Location: Bolzano, Italy
- Preliminary Time: Sep-2019
- Preliminary Name of a larger summit:

  BRAINS - Bolzano Rules and Artificial Intelligence Summit

THANK YOU FOR ATTENDING DECISION CAMP 2018!

Please send me your feedback with suggestions for improvements.