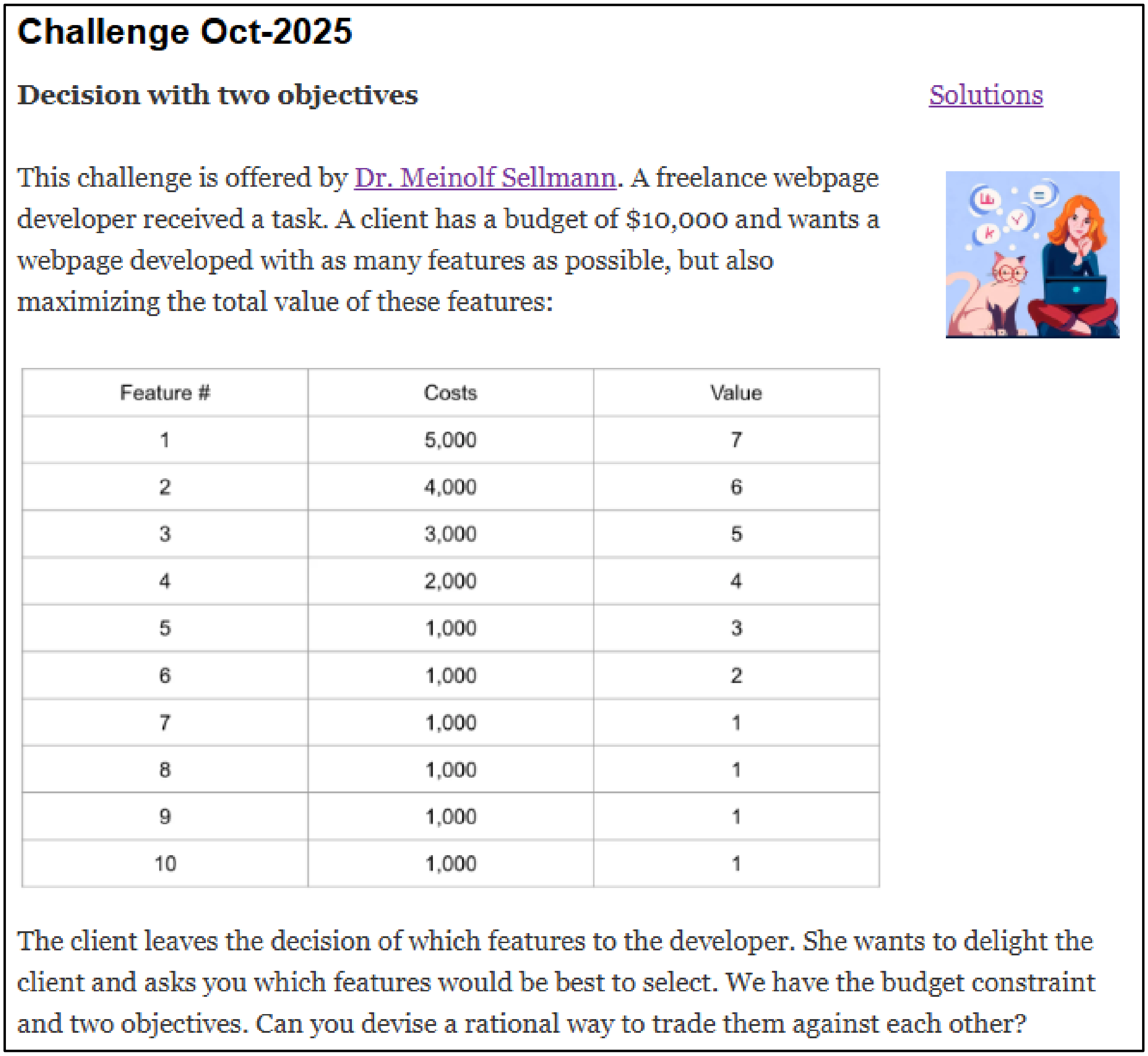
Decision Management Community Challenge Oct-2025

“Decision with two objectives”

https://dmcommunity.org/challenge/challenge-oct-2025/



Challenge Oct-2025

Decision with two objectives

Solutions

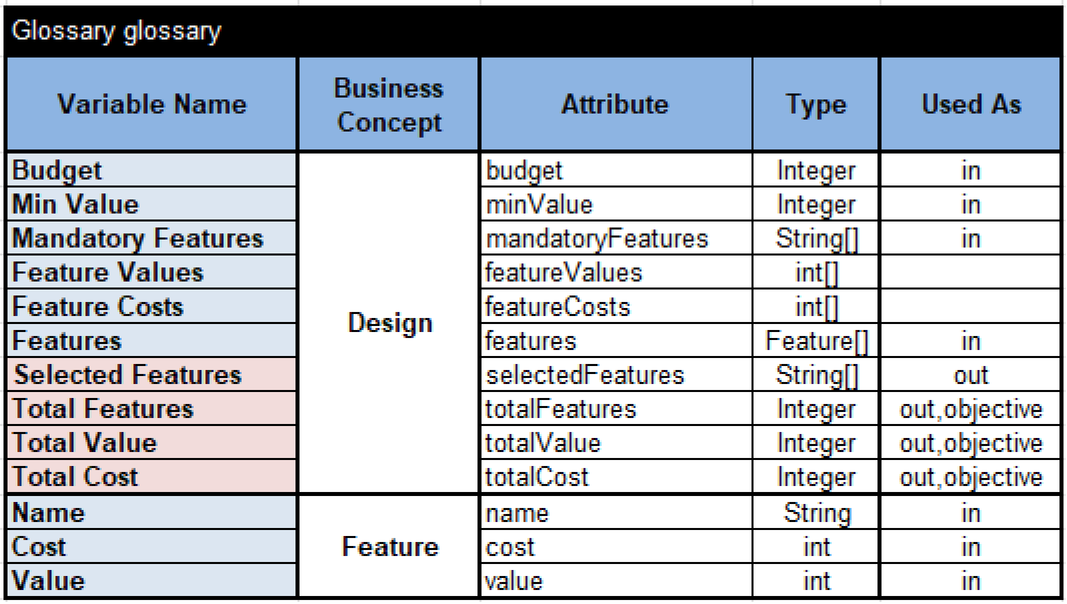
This challenge is offered by Dr. Meinolf Sellmann. A freelance webpage developer received a task. A client has a budget of $10,000 and wants a webpage developed with as many features as possible, but also maximizing the total value of these features:

Feature #	Costs	Value
1	5,000	7
2	4,000	6
3	3,000	5
4	2,000	4
5	1,000	3
6	1,000	2
7	1,000	1
8	1,000	1
9	1,000	1
10	1,000	1

The client leaves the decision of which features to the developer. She wants to delight the client and asks you which features would be best to select. We have the budget constraint and two objectives. Can you devise a rational way to trade them against each other?

Here is the implementation of this challenge using Rule Solver. I added an OpenRules project “Features” to the standard folder for Rule Solver “openrules.solver”.

Here is my business Glossary for this challenge:



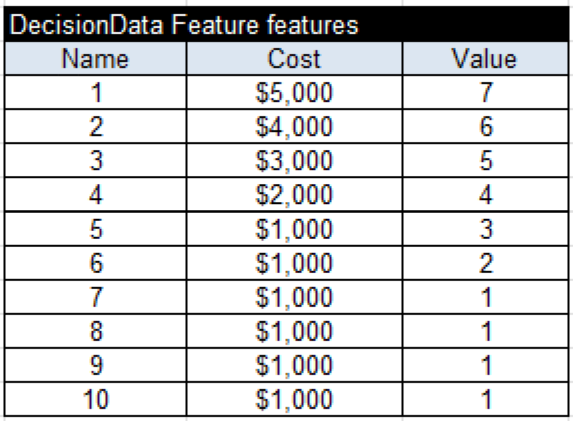
Glossary glossary				
Variable Name	**Business Concept**	**Attribute**	**Type**	**Used As**
Budget	**Design**	budget	Integer	in
Min Value		minValue	Integer	in
Mandatory Features		mandatoryFeatures	String[]	in
Feature Values		featureValues	int[]	
Feature Costs		featureCosts	int[]	
Features		features	Feature[]	in
Selected Features		selectedFeatures	String[]	out
Total Features		totalFeatures	Integer	out,objective
Total Value		totalValue	Integer	out,objective
Total Cost		totalCost	Integer	out,objective
Name	**Feature**	name	String	in
Cost		cost	int	in
Value		value	int	in

It contains the main business concept "Design" that includes the array "Features" with their names, costs, and values as described in the business concept "Feature". They will come to this decision model as input. We want our decision model to produce the output array "Selected Features" and show their "Total Features", "Total Cost" and "Total Value".

I added two more input decision variables that extend the challenge a bit beyond its formulation but will be convenient to manipulate with this decision model:

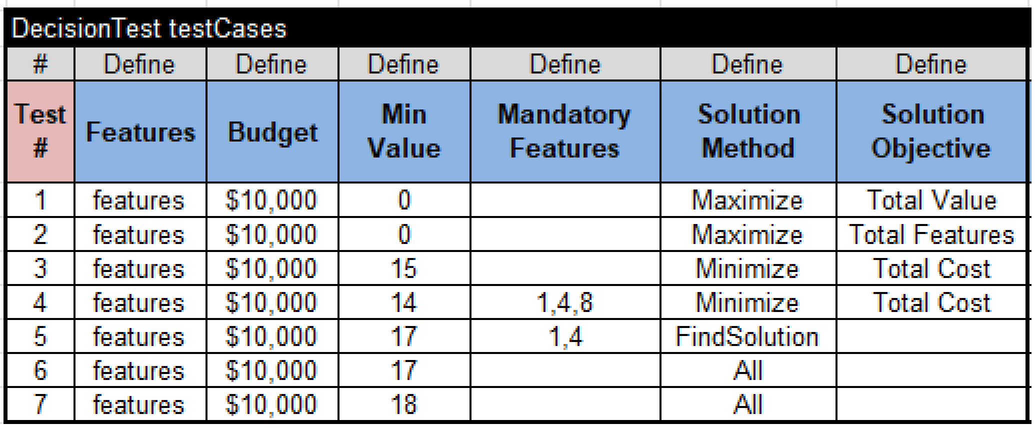
- **Min Value** specifies the minimum value of all Selected Features
- **Mandatory Features** will allow a user to force the decision model to select certain features by listing their names inside this array.

We may define features with costs and values in the following table:



DecisionData Feature features		
Name	Cost	Value
1	$5,000	7
2	$4,000	6
3	$3,000	5
4	$2,000	4
5	$1,000	3
6	$1,000	2
7	$1,000	1
8	$1,000	1
9	$1,000	1
10	$1,000	1

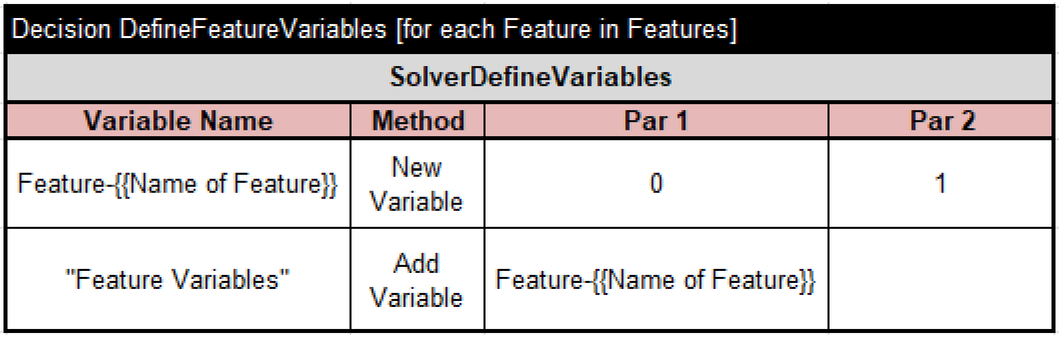
We may refer to this table in our test cases:



DecisionTest testCases						
#	Define	Define	Define	Define	Define	Define
Test #	**Features**	**Budget**	**Min Value**	**Mandatory Features**	**Solution Method**	**Solution Objective**
1	features	$10,000	0		Maximize	Total Value
2	features	$10,000	0		Maximize	Total Features
3	features	$10,000	15		Minimize	Total Cost
4	features	$10,000	14	1,4,8	Minimize	Total Cost
5	features	$10,000	17	1,4	FindSolution	
6	features	$10,000	17		All	
7	features	$10,000	18		All	

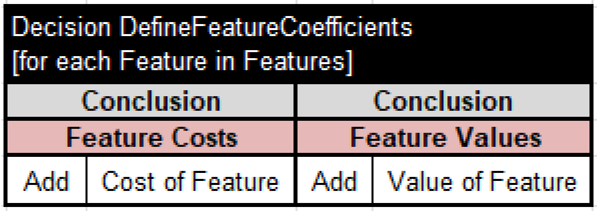
As you can see, I want to ask my decision model to find decision that will minimize/maximize different objectives with different mandatory features and minimal total values.

Then I created this decision table that for each feature defines a Solver decision variable with values 0 (not selected) or 1 (selected) and added all these variables to the array "Feature Variables":



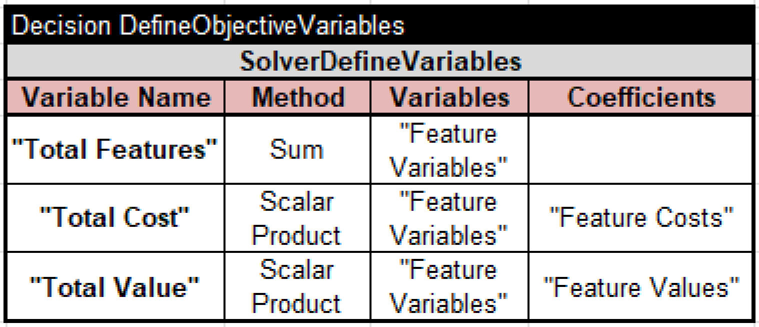
Decision DefineFeatureVariables [for each Feature in Features]			
SolverDefineVariables			
Variable Name	**Method**	**Par 1**	**Par 2**
Feature-{{Name of Feature}}	New Variable	0	1
"Feature Variables"	Add Variable	Feature-{{Name of Feature}}	

The following table defines arrays "Feature Costs" and "Feature Values" that will be needed to specify our objective variables:



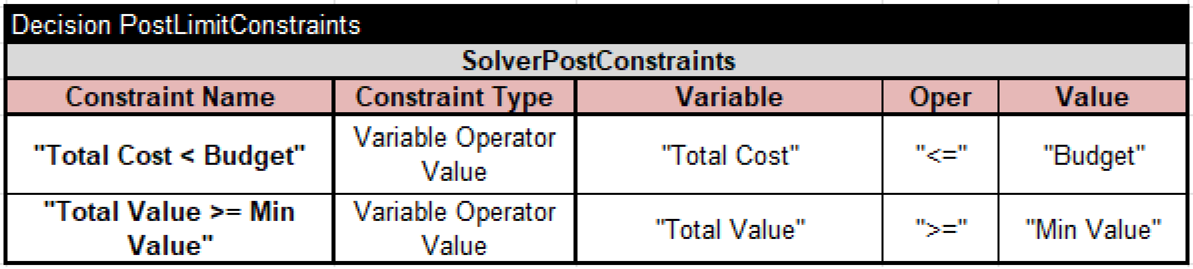
Decision DefineFeatureCoefficients [for each Feature in Features]			
Conclusion		**Conclusion**	
Feature Costs		**Feature Values**	
Add	Cost of Feature	Add	Value of Feature

And here is the table that uses the standard Solver's operator "Scalar Product" to define our problem objectives:



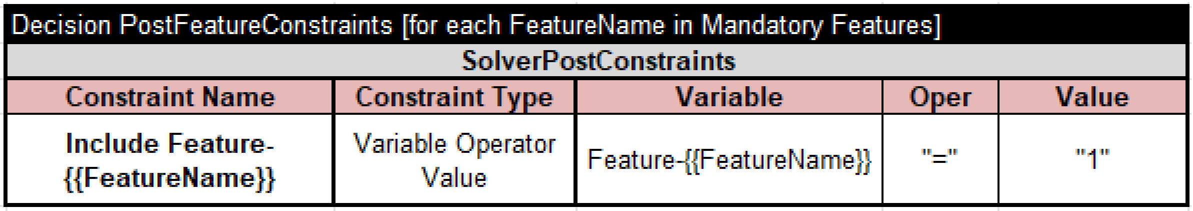
Decision DefineObjectiveVariables			
SolverDefineVariables			
Variable Name	**Method**	**Variables**	**Coefficients**
"Total Features"	Sum	"Feature Variables"	
"Total Cost"	Scalar Product	"Feature Variables"	"Feature Costs"
"Total Value"	Scalar Product	"Feature Variables"	"Feature Values"

The next table posts constraints that limit our objectives:



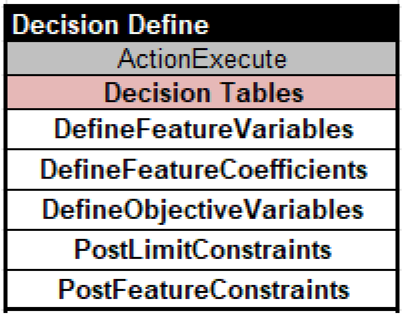
Decision PostLimitConstraints				
SolverPostConstraints				
Constraint Name	**Constraint Type**	**Variable**	**Oper**	**Value**
"Total Cost < Budget"	Variable Operator Value	"Total Cost"	"<="	"Budget"
"Total Value >= Min Value"	Variable Operator Value	"Total Value"	">="	"Min Value"

To make sure that the Selected Features will include all Mandatory Features we post the following constraints:



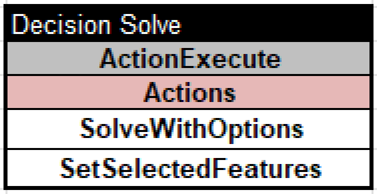
Decision PostFeatureConstraints [for each FeatureName in Mandatory Features]				
SolverPostConstraints				
Constraint Name	**Constraint Type**	**Variable**	**Oper**	**Value**
Include Feature-{{FeatureName}}	Variable Operator Value	Feature-{{FeatureName}}	"="	"1"

This completes our problem definition that we may define in the table "Define":



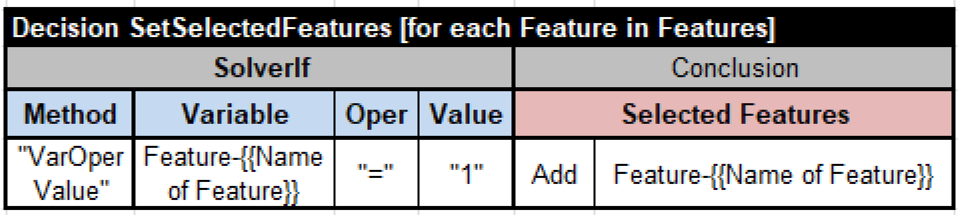
Decision Define
ActionExecute
Decision Tables
DefineFeatureVariables
DefineFeatureCoefficients
DefineObjectiveVariables
PostLimitConstraints
PostFeatureConstraints

To solve the problem, we will use the following table "Solve" with the predefines Solver's method "SolveWithOptions":



Decision Solve
ActionExecute
Actions
SolveWithOptions
SetSelectedFeatures

After solving the problem, we will execute the table “SetSelectedFeatures” to add all features with non-null decision variables Feature-1, Feature-2, etc. to out output array “Selected Features”:



Decision SetSelectedFeatures [for each Feature in Features]					
SolverIf				Conclusion	
Method	Variable	Oper	Value	Selected Features	
"VarOper Value"	Feature-{{Name of Feature}}	"="	"1"	Add	Feature-{{Name of Feature}}

Execution Results

Now we can execute our decision model against the test cases shown above. Below are the execution results for each test case.

Test Case 1 “Maximize Total Value” (Min Value=0, no mandatory features)

Our decision model finds an optimal solution that includes:

Feature-2, Feature-3, Feature-4, Feature-5

Total Features = 4, Total Cost = 10000, Total Value = **18**

Execution time: 46 msec

Test Case 2 “Maximize Total Features” (Min Value=0, no mandatory features)

Our decision model finds an optimal solution that includes:

Feature-4, Feature-5, Feature-6, Feature-7, Feature-8, Feature-9, Feature-10

Total Features = 7, Total Cost = 8000, Total Value = **13**

Execution time: 10 msec

Test Case 3 “Maximize Total Features” (Min Value=17, no mandatory features)

Our decision model finds an optimal solution that includes:

Feature-3, Feature-4, Feature-5, Feature-6, Feature-8, Feature-9, Feature-10

Total Features = 7, Total Cost = 10000, Total Value = **17**

Execution time: 15 msec

Test Case 4 “Minimize Total Cost” (Min Value=14 and mandatory features 1,4, 8)

Our decision model finds an optimal solution that includes:

Feature-1, Feature-4, Feature-6, Feature-8

Total Features = 4, Total Cost = 9000, Total Value = **14**

Execution time: 16 msec

Test Case 5 “Find Solution” (Min Value=17 and mandatory features 1,4)

Our decision model finds a feasible solution that includes:

Feature-1, Feature-4, Feature-5, Feature-6, Feature-10

Total Features = 5, Total Cost = 10000, Total Value = **17**

Execution time: 1 msec

Test Case 6 “Find All Solutions” (Min Value=17 and no mandatory features)

Our decision model finds **21 feasible solutions**:

Solution #1:
Feature-3, Feature-4, Feature-5, Feature-6, Feature-8, Feature-9, Feature-10
Total Features = 7, Total Cost = 10000, Total Value = **17**

Solution #2:
Feature-3, Feature-4, Feature-5, Feature-6, Feature-7, Feature-9, Feature-10
Total Features = 7, Total Cost = 10000,   Total Value = 17
…

Solution #21:
  Feature-1, Feature-3, Feature-5, Feature-6
  Total Features = 4, Total Cost = 10000, Total Value = 17

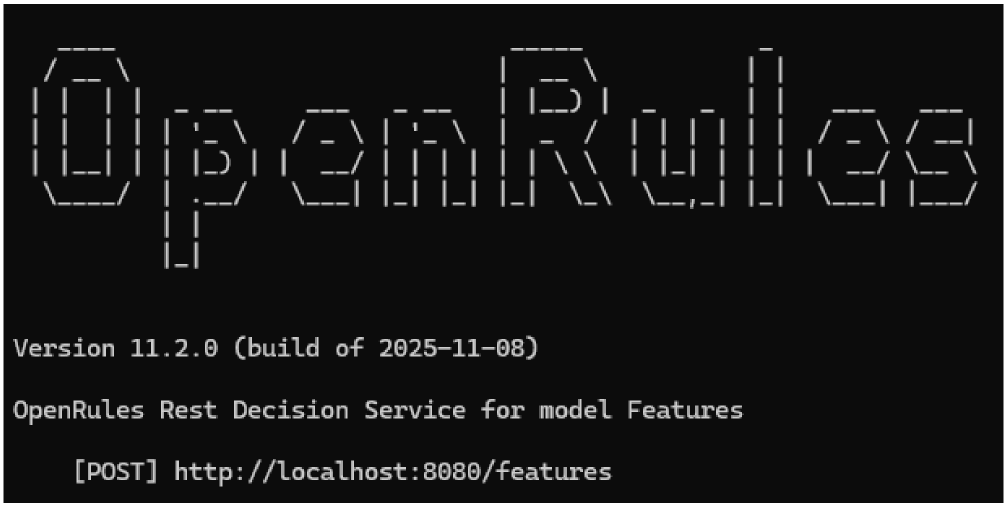
Total Execution time: 84 msec

Test Case 7 "Find All Solutions" (Min Value=18 and no mandatory features)

Our decision model finds only 1 feasible solution:

Solution #1:
  Feature-2, Feature-3, Feature-4, Feature-5
  Total Features = 4, Total Cost = 10000, Total Value = 18
  Execution time: 9 msec

The JSON files that correspond to our test cases were automatically generated by OpenRules in the folder "jsons". I wanted this model to become available as a decision service to work with different inputs provided in the JSON format. With one click on the standard "runLocalServer.bat" I deployed this decision model as a REST service on my local server:

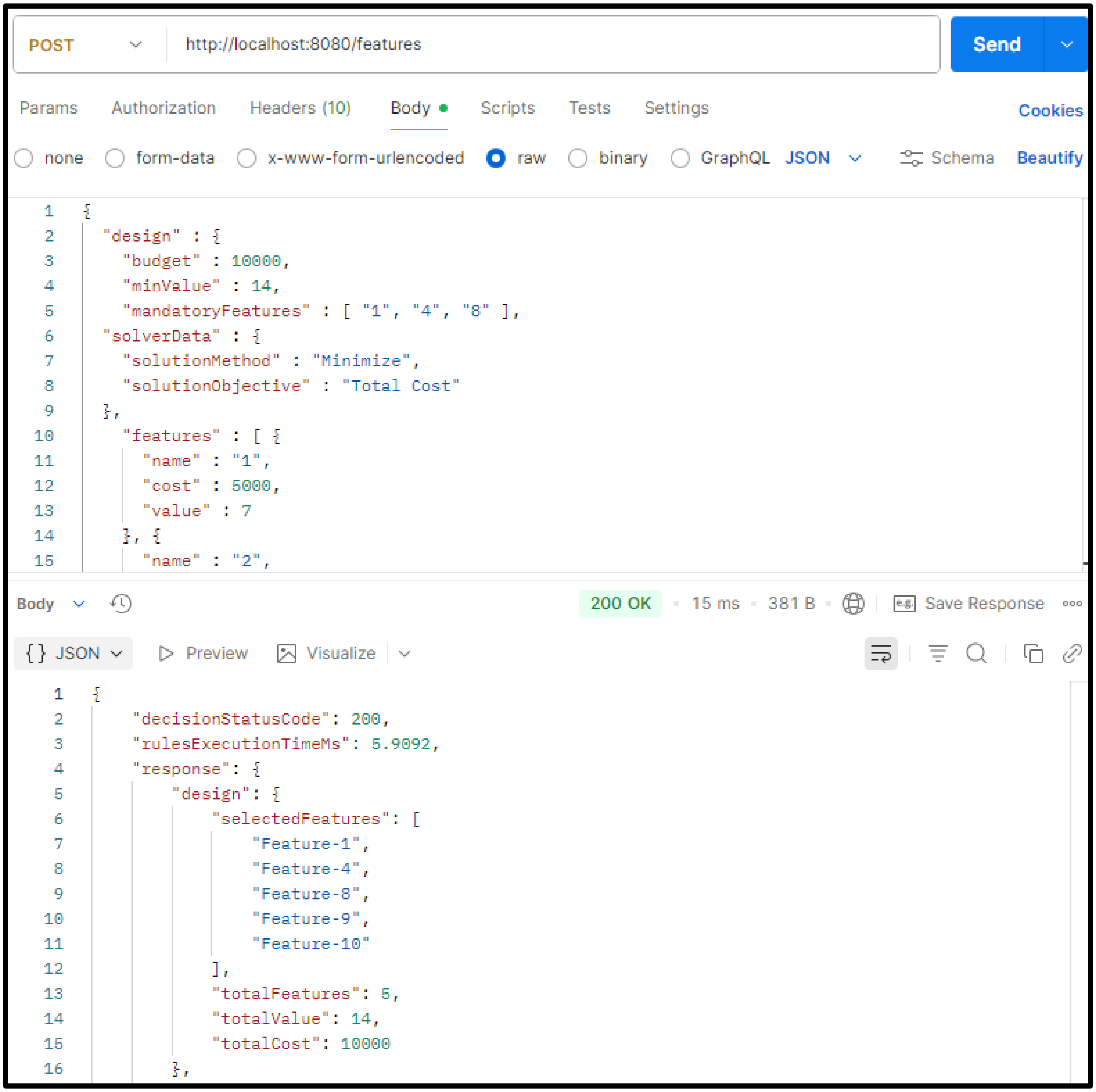


```
Version 11.2.0 (build of 2025-11-08)

OpenRules Rest Decision Service for model Features

    [POST] http://localhost:8080/features
```

Then I tested this decision service using POSTMAN and JSON request from the file jsons/testCases-4.json:



POST http://localhost:8080/features

```
{
  "design" : {
    "budget" : 10000,
    "minValue" : 14,
    "mandatoryFeatures" : [ "1", "4", "8" ],
  "solverData" : {
    "solutionMethod" : "Minimize",
    "solutionObjective" : "Total Cost"
  },
    "features" : [ {
      "name" : "1",
      "cost" : 5000,
      "value" : 7
    }, {
      "name" : "2",
```

200 OK • 15 ms • 381 B

```
{
    "decisionStatusCode": 200,
    "rulesExecutionTimeMs": 5.9092,
    "response": {
        "design": {
            "selectedFeatures": [
                "Feature-1",
                "Feature-4",
                "Feature-8",
                "Feature-9",
                "Feature-10"
            ],
            "totalFeatures": 5,
            "totalValue": 14,
            "totalCost": 10000
        },
```

In this POSTMAN view a user can modify budget, minValue, mandatoryFeatures, solutionMethod, and solutionObjective, click again on “Send” and analyze the results.