Exploring the Parameter Space

Pt. 1: Sensitivity Analysis

Nick DeRobertis[1]

[1]University of Florida
Department of Finance, Insurance, and Real Estate

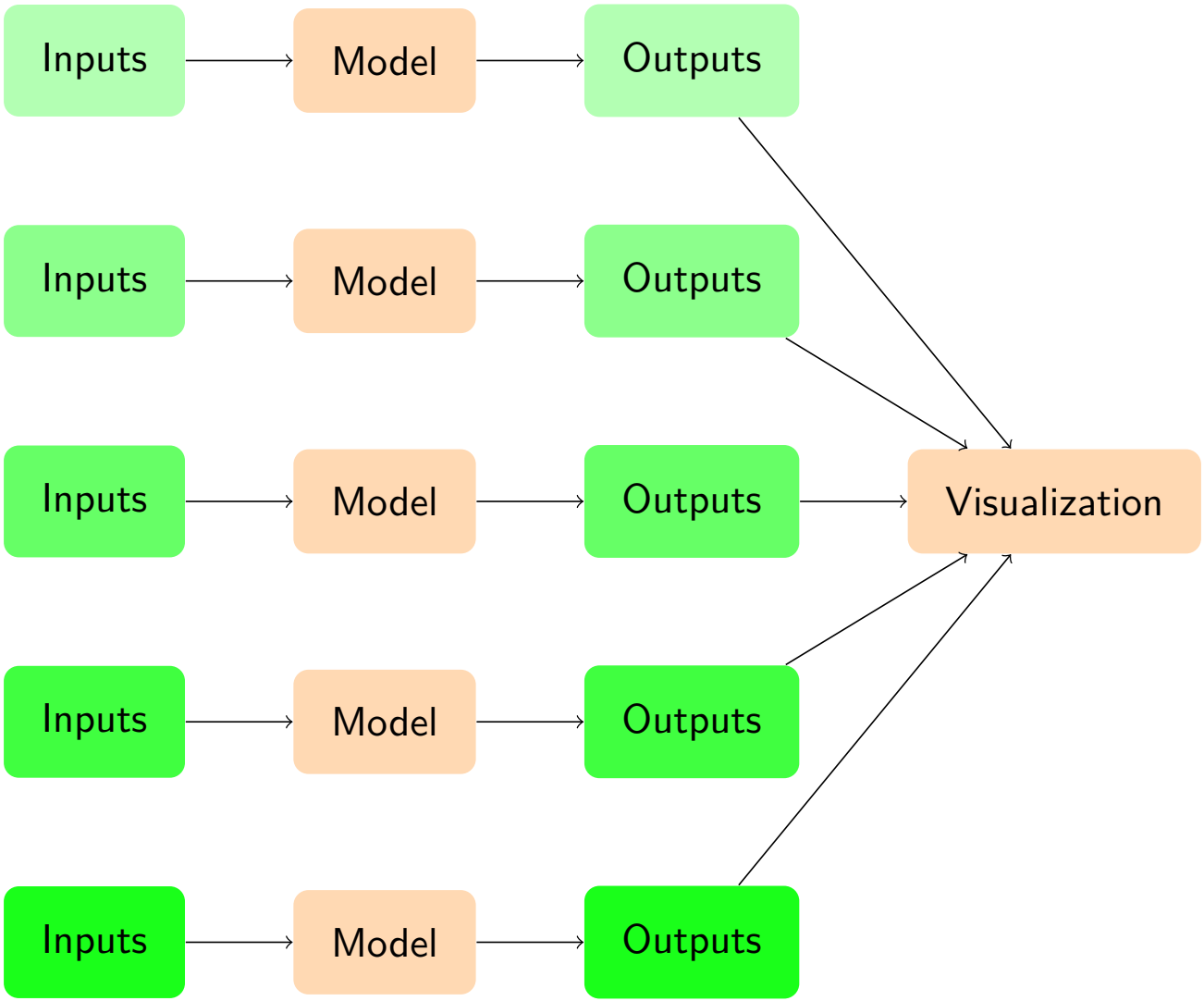
September 1, 2020

Table of Contents

Moving from a Static Model

- So far, I have given you some inputs to use and you have been getting one or more outputs from those inputs
- We have not considered how those inputs may change, and how that affects the outputs
- This is where building a model vs. doing a calculation really starts to pay off

How to Explore Inputs and their Affect on Outputs



Methods of Parameter Exploration

- In this lecture, we will be discussing sensitivity analysis as an approach to exploring the parameter space.
- After we cover probabilistic modeling, we will revisit exploring the parameter space with a method called Monte Carlo Simulation.
- In sensitivity analysis, a fixed set of values for the parameters are chosen, while in Monte Carlo Simulation, each parameter is assigned a distribution.
- Both methods may be used together to fully understand a model.

Table of Contents

Sensitivity Analysis, Formally

For the model given by:

$$y = f(X) \tag{1}$$

$$X = [x_1, x_2, ..., x_n] \tag{2}$$

- y : Model output
- X : Model input matrix
- x_i : Value of ith x variable

Follow the following steps:

1. Choose a set of values for each x_i
2. Take the cartesian product of these values as $[X_1, X_2, ..., X_m]$
3. For each X_i calculate $y_i = f(X_i)$
4. Store the values of X_i mapped to y_i
5. Visualize y_i versus X_i

Sensitivity Analysis Example Model

Let's take a simple demand model as an example:

$$D = c - EP \tag{3}$$

$$X = [c, E, P] \tag{4}$$

- D : Quantity demanded
- c : Demand constant
- E : Elasticity of demand
- P : Price
- X : Model input matrix

Sensitivity Analysis Example

$$D = c - EP \tag{5}$$

Follow the following steps:

1. Choose $c = (60000, 100000), E = (200, 500), P = (50, 100)$
2. Take the cartesian product of these values, yielding $[X_1, X_2, ..., X_m]$:

c	E	P
60000	200	50
60000	200	100
60000	500	50
60000	500	100
100000	200	50
100000	200	100
100000	500	50
100000	500	100

Sensitivity Analysis Example, Pt. 2

$$D = c - EP \tag{6}$$

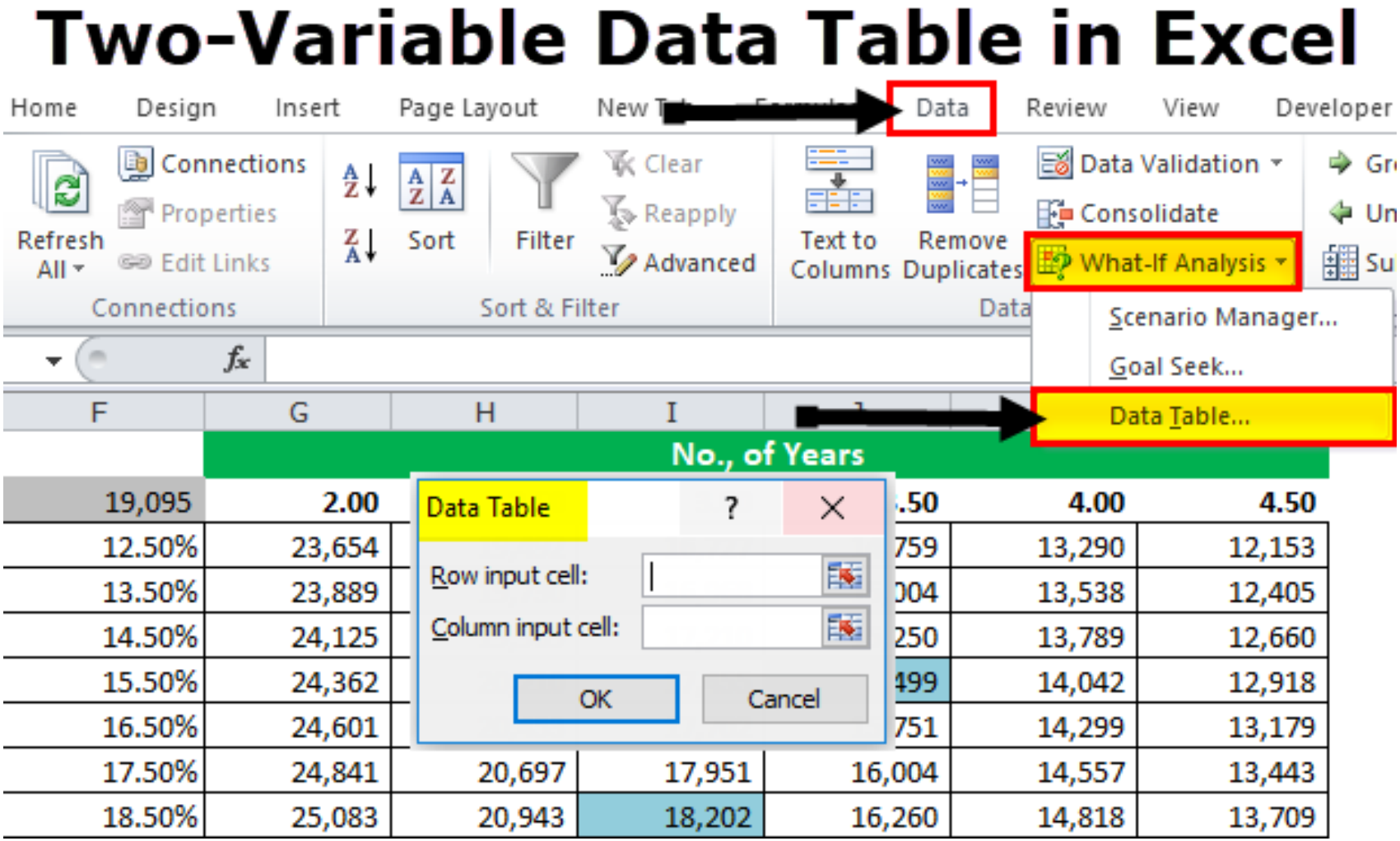
Continue following the steps:

3. For each X_i calculate $y_i = f(X_i)$
4. Store the values of X_i mapped to y_i

c	E	P	D
60000	200	50	50000
60000	200	100	40000
60000	500	50	35000
60000	500	100	10000
100000	200	50	90000
100000	200	100	80000
100000	500	50	75000
100000	500	100	50000

Table of Contents

Sensitivity Analysis in Excel



Visualizing Sensitivity Analysis in Excel

- There are two main ways to visualize sensitivity analysis results in Excel: graphing and conditional formatting.
- Graphing is usually appropriate for one-way data tables
- Conditional formatting is usually appropriate for two-way data tables

Conditional Formatting in Excel

	A	B	C	D	E	F	G
1	City	Jan	Feb	Mar	Apr	May	Jun
2	Barstow	80	84	84	97	95	98
3	California City	78	86	84	96	98	102
4	Cinco	83	86	86	97	95	103
5	Hesperia	78	85	87	98	97	102
6	Lancaster	78	85	86	99	95	101
7	Mojave	82	85	86	98	96	99
8	Palmdale	81	84	85	97	95	101
9	Ridgecrest	81	87	87	97	96	98
10	Rosamond	82	86	88	99	97	101
11	Santa Clarita	79	85	87	95	96	103

Sensitivity Analysis in Excel

Adding Sensitivity Analysis to the Dynamic Retirement Excel Model

- I will now go through adding sensitivity analysis to the Dynamic Salary Retirement Model in Excel
- The completed exercise is in Examples > Sensitivity Analysis > Excel > Dynamic Salary Retirement Model Sensitivity.xlsx

Sensitivity Analysis in Excel Lab, Level 1

Adding Sensitivity Analysis to Project 1, Level 1

1. Add sensitivity analysis to your Excel model from Project 1
2. See how the NPV changes when the number of machines and initial demand change
3. Do a one-way Data Table with a graph for each of the two inputs, then a two-way data table with conditional formatting

Table of Contents

Going Deeper into Python Code Structure

We'll cover a couple more Python patterns and a new data type before jumping into sensitivity analysis

- Dictionaries
- List comprehensions
- Python `import` system and custom code

What is a Dictionary?

- A dictionary, or `dict` for short, is another basic Python data type like lists, numbers, and strings.
- Like a list, it is a collection: it holds other objects.
- Unlike a list, a `dict` is composed of key-value pairs. It holds relationships between objects.

How to Use Dictionaries

Basic Dictionary Example

```
>>> coffee_levels_emotions = {
>>>     'high': 'happy',
>>>     'pretty high': 'happy',
>>>     'medium': 'neutral',
>>>     'low': 'sad',
>>>     'empty': 'desparate'
>>> }
>>> coffee_levels_emotions['pretty high']
'happy'
>>> for coffee_level, emotion in coffee_levels_emotions.items():
>>>     print(f"I'm {emotion} when my coffee is {coffee_level}")

I'm happy when my coffee is high
I'm happy when my coffee is pretty high
I'm neutral when my coffee is medium
I'm sad when my coffee is low
I'm desparate when my coffee is empty
```

How to Modify Dictionaries

Add and Delete Items from Dictionaries

```
>>> coffee_levels_emotions.update({'overflowing': 'burned'})
>>> coffee_levels_emotions['negative'] = 'confused'
>>> high_value = coffee_levels_emotions.pop('high')
>>> coffee_levels_emotions
{'pretty high': 'happy',
 'medium': 'neutral',
 'low': 'sad',
 'empty': 'desparate',
 'overflowing': 'burned',
 'negative': 'confused'}
>>> high_value
'happy'
```

More About Dictionaries in Python

Using Dictionaries

- I will now start going through the example notebook in Examples > Sensitivity Analysis > Python > Python Dicts, List comprehensions, and Imports.ipynb
- I will go through the Dictionaries section for now

Dictionaries Lab, Level 1

Learning How to Use Dictionaries, Level 1

1. For this Python section, lab exercises are in Labs > Sensitivity Analysis > Python > Python Dicts, List comprehensions, and Imports Lab.ipynb
2. Complete the exercises in the dictionaries section for now

An Easier way to Build Simple Lists from Loops

The Original Way

```
>>> out_values = []
>>> for i in range(5):
>>>     out_values.append(i + 10)
>>> out_values
[10, 11, 12, 13, 14]
```

With List Comprehension

```
>>> out_values = [i + 10 for i in range(5)]
>>> out_values
[10, 11, 12, 13, 14]
```

Notice

You **never** need to use list comprehension, it is just for convenience. The original `for` loop syntax will always work fine.

Easier Loops in Python

Using List Comprehensions

- I will continue going through the example notebook in Examples > Sensitivity Analysis > Python > Python Dicts, List comprehensions, and Imports.ipynb
- I will go through the List Comprehensions section for now

List Comprehensions Lab, Level 1

Learning How to Use List Comprehensions, Level 1

1. For this Python section, lab exercises are in Labs > Sensitivity Analysis > Python > Python Dicts, List comprehensions, and Imports Lab.ipynb
2. Complete the exercises in the List Comprehensions section for now

Understanding Python `imports`

- In the past we have used `import` to load packages such as `numpy` and `pandas`
- These packages are just Python files. We can also write our own Python files and `import` them the same way
- When you `import something`, Python first searches the current directory for a file `something.py` and if it doesn't find it, it searches your installed packages
- In fact if you added a `numpy.py` in the current directory and tried to `import numpy` it would `import` the contents of that file rather than the `numpy` package.

Importing Custom Code

- You can write your own functions and classes, then put them in a Python file and `import` them into your notebook.
- When you `import` a file, it executes the contents of that file. So you generally want just function and class definitions, and not really anything outside of `def` or `class` statements.
- Using Python files is a more maintainable structure for building complex models and apps versus Jupyter notebooks only.

Installing Packages

- Sometimes you will need a package which does not already come installed with Anaconda
- The general way to do this is with `pip install mypackage` replacing mypackage with the package you want to install
- You would run this in Anaconda Prompt, or in Jupyter you can run it but you need to put an exclaimation mark before it to say you want to run it in a terminal. So in Jupyter it would be `!pip install mypackage`

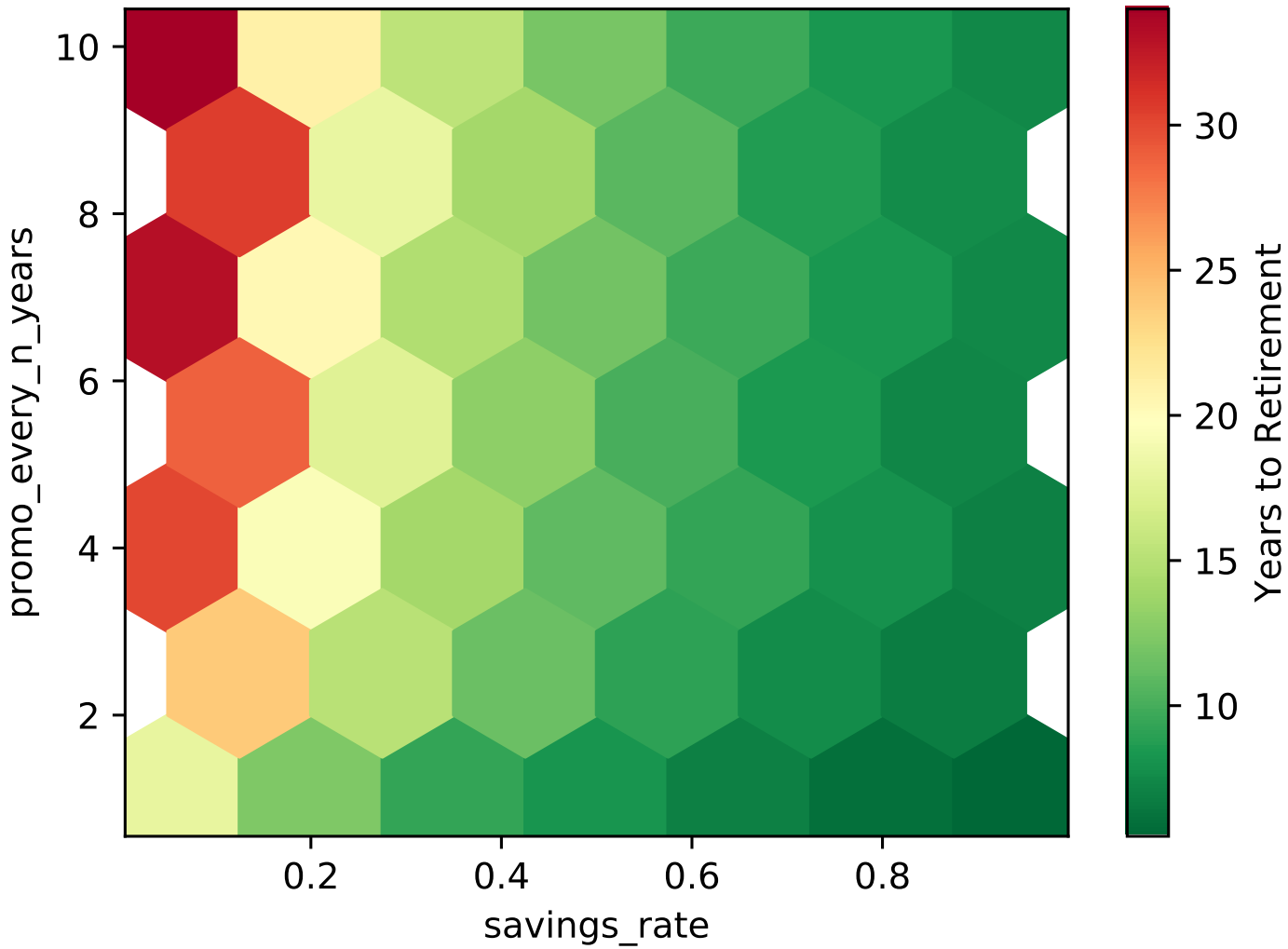
Installing Packages in Python

How to Install Packages

- I will continue going through the example notebook in Examples > Sensitivity Analysis > Python > Python Dicts, List comprehensions, and Imports.ipynb
- I will go through the Imports and Installing Packages section for now

Table of Contents

Sensitivity Analysis in Python - Hex-Bin



Sensitivity Analysis in Python - Styled DataFrame

Result - value1 vs. value2

value1	10	11	12	13	14	15	16	17	18	19
0	15	16	17	18	19	20	21	22	23	24
1	16	17	18	19	20	21	22	23	24	25
2	17	18	19	20	21	22	23	24	25	26
3	18	19	20	21	22	23	24	25	26	27
4	19	20	21	22	23	24	25	26	27	28
5	20	21	22	23	24	25	26	27	28	29
6	21	22	23	24	25	26	27	28	29	30
7	22	23	24	25	26	27	28	29	30	31
8	23	24	25	26	27	28	29	30	31	32
9	24	25	26	27	28	29	30	31	32	33

How to Do Sensitivity Analysis in Python (The Hard Way)

- Generally, to do sensitivity analysis in Python without any special tools, you would just create one nested for loop for each input, and finally within all the loops, run your model with the inputs from the loops
- This will work fine, but you will have many nested loops which can become hard to read. Also it is a fair bit of setup involved.
- You can avoid the nested loops with `itertools.product` but then this becomes more difficult to use and read

Sensitivity Analysis Example (Hard Way)

- Say you have a function which runs your model, called model, which takes inputs of inp1 and inp2

Sensitivity Analysis in Python with No Libraries

```
inp1_values = [1, 2]
inp2_values = [4, 5]
results = []
for inp1 in inp1_values:
    for inp2 in inp2_values:
        result = model(inp1, inp2,)
        results.append(
            (inp1, inp2, result)
        )
pd.DataFrame(results, columns=['inp1', 'inp2',  'Result'])
```

	inp1	inp2	Result
0	1	4	5
1	1	5	6
2	2	4	6
3	2	5	7

How to Do Sensitivity Analysis in Python (The Easy Way)

- When I first created this course, I thought there should be a good sensitivity analysis tool in Python and I couldn't find it
- The beauty of Python is if you want a tool that doesn't exist, you can create it, and share it with others so that nobody else has to deal with the problem.
- So I created `sensitivity` a package for sensitivity analysis in Python, which makes it very easy

Sensitivity Analysis Example (Easy Way)

- Say you have a function which runs your model, called model, which takes inputs of inp1 and inp2

Sensitivity Analysis in Python with `sensitivity`

```
from sensitivity import SensitivityAnalyzer

sensitivity_values = {
    'inp1': [1, 2],
    'inp2': [4, 5],
}
sa = SensitivityAnalyzer(sensitivity_values, model)
sa.df
```

	inp1	inp2	Result
0	1	4	5
1	1	5	6
2	2	4	6
3	2	5	7

Sensitivity Analysis in Python

Adding Sensitivity Analysis to the Dynamic Retirement Python Model

- I will now go through adding sensitivity analysis to the Dynamic Salary Retirement Model in Python
- The completed exercise is in Examples > Sensitivity Analysis > Python > Dynamic Salary Retirement Model Sensitivity.ipynb

Sensitivity Analysis in Python Lab, Level 1

Adding Sensitivity Analysis to Project 1, Level 1

1. Add sensitivity analysis to your Python model from Project 1
2. See how the NPV changes when the number of machines and initial demand change
3. Output both a hex-bin plot and a styled DataFrame