

Make model validation sexy again

Sunniva Indrehus

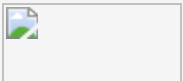
Norwegian Geotechnical Institute



Make model validation sexy ~~again~~

Sunniva Indrehus

Norwegian Geotechnical Institute



Scientist's work?

Understand a simplified version
of the real world



Figure credit: [Ali Bati](#)

RSE's work?

Code and understand a simplified
version of the real world

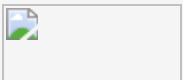


Figure credit: [Ali Bati](#)

Data Model

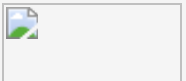
A data model is an abstract model that organizes elements of data and standardizes how they relate to one another and to the properties of real-world entities.

from [Wikipedia](#)



Building a data model

Figure: a model of a turbine



Building a data model

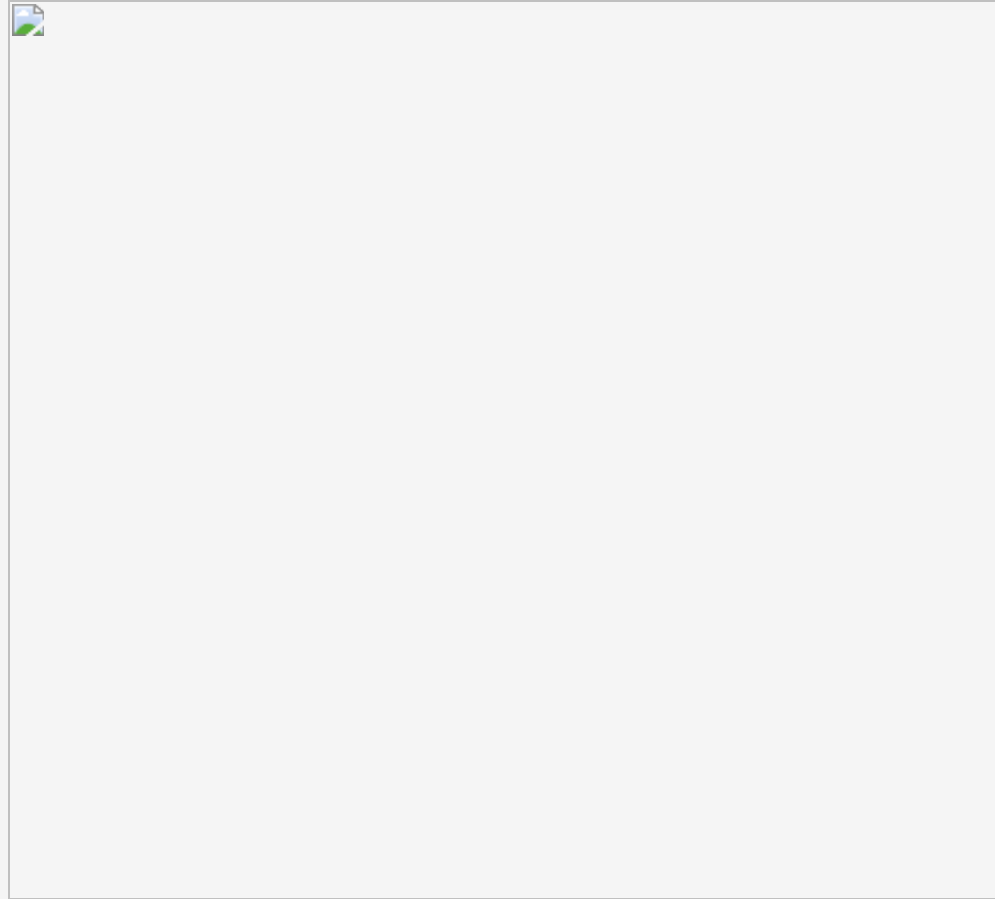
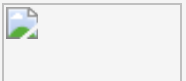


Figure: a simple model of a turbine + environment



Building a data model

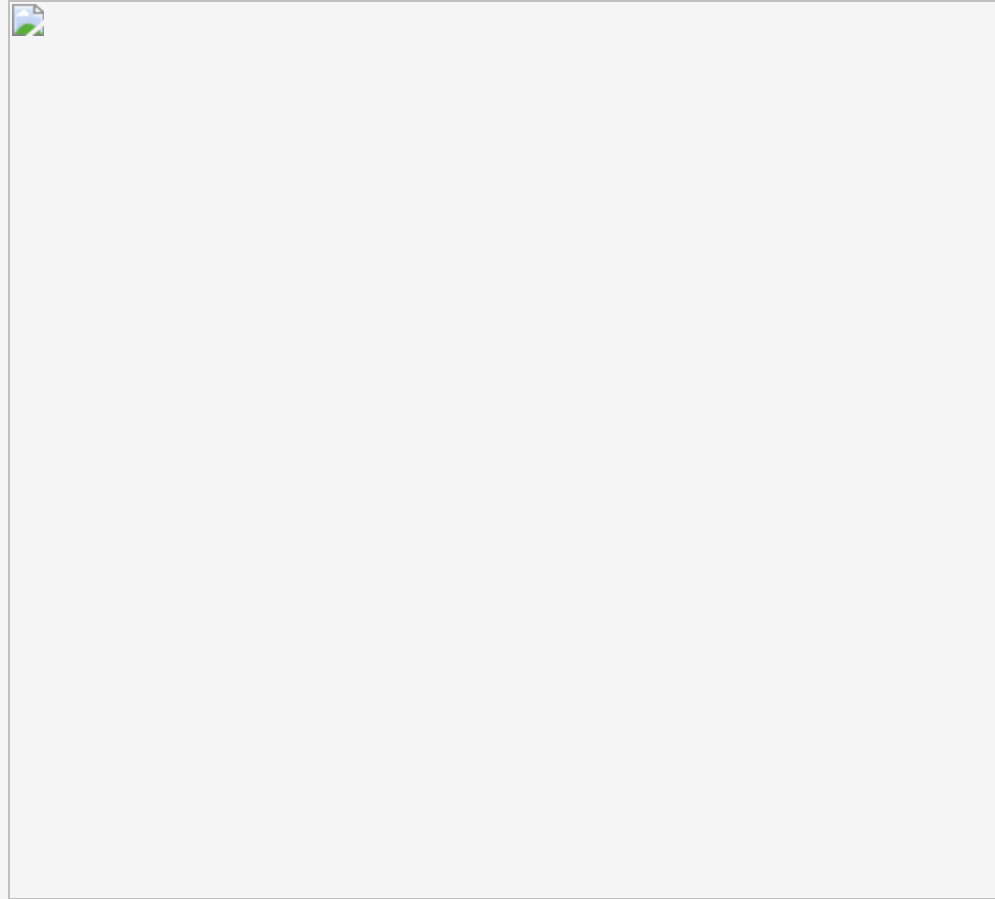
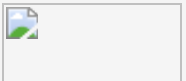


Figure: a less simpler model of a turbine + environment



Building a data model

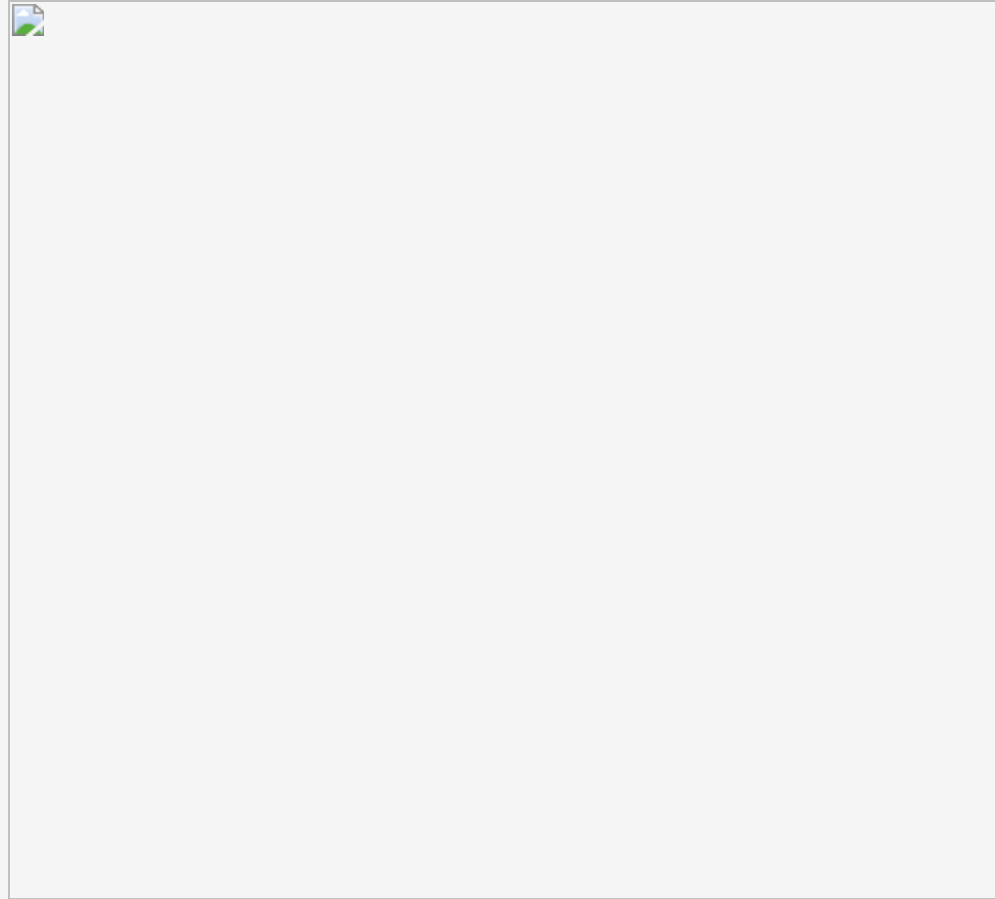
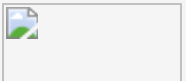
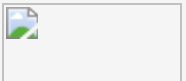


Figure: a full model of a turbine + environment



```
<div align="center">
```

🎵 ... **Then a (*super*)hero comes a long** 🎵

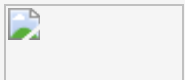


What is pydantic?

Data validation and settings management using python type annotations. Pydantic enforces type hints at runtime, and provides user friendly errors when data is invalid.

From the official docs

GitHub: ★ 11.3k (currently, 18.10.22)



Class definitions

<div class="twocols">

```
from pydantic import BaseModel, Field

class SoilLayer(BaseModel):
    depth: float = Field(description="Depth from seabed to soil later")
    number_of_elements: int = Field(
        description="Number of elements of this material at this depth"
    )
    # ... etc. for validation
```

<p class="break"></p>

```
class TurbineModel(BaseModel):
    soil_layers: list[SoilLayer]
    load_step_num: int = Field(
        default=20, ge=0, description="Number of load steps in cycle"
    )
    # ... etc. for validation
```



</div>

Give input with dictionaries

```
soil_layers = [{"depth": 0, "number_of_elements": 2}, {"depth": 2, "number_of_elements": 3}]  
  
simulation_steps = 20  
  
turbine_model = TurbineModel(  
    soil_layers=soil_layers,  
    load_step_num = simulation_steps  
)  
  
print(f"My turbine model: {turbine_model}")
```

Print message

```
My turbine model: soil_layers=[SoilLayer(depth=0.0, number_of_elements=2), SoilLayer(depth=2.0, number_of_elements=3)] load_step_num=20
```



Give input with dictionaries

```
soil_layers = [{"depth": 0, "number_of_elements": 2}, {"depth": 2, "number_of_elements": 3}]  
  
simulation_steps = -20  
  
turbine_model = TurbineModel(  
    soil_layers=soil_layers,  
    load_step_num = simulation_steps  
)  
  
print(f"My turbine model: {turbine_model}")
```

"Free" error messages

```
pydantic.error_wrappers.ValidationError: 1 validation error for TurbineModel  
load_step_num ensure this value is greater than or equal to 0 (type=value_error.number.not_ge; limit_value=0)
```



Calling procedure

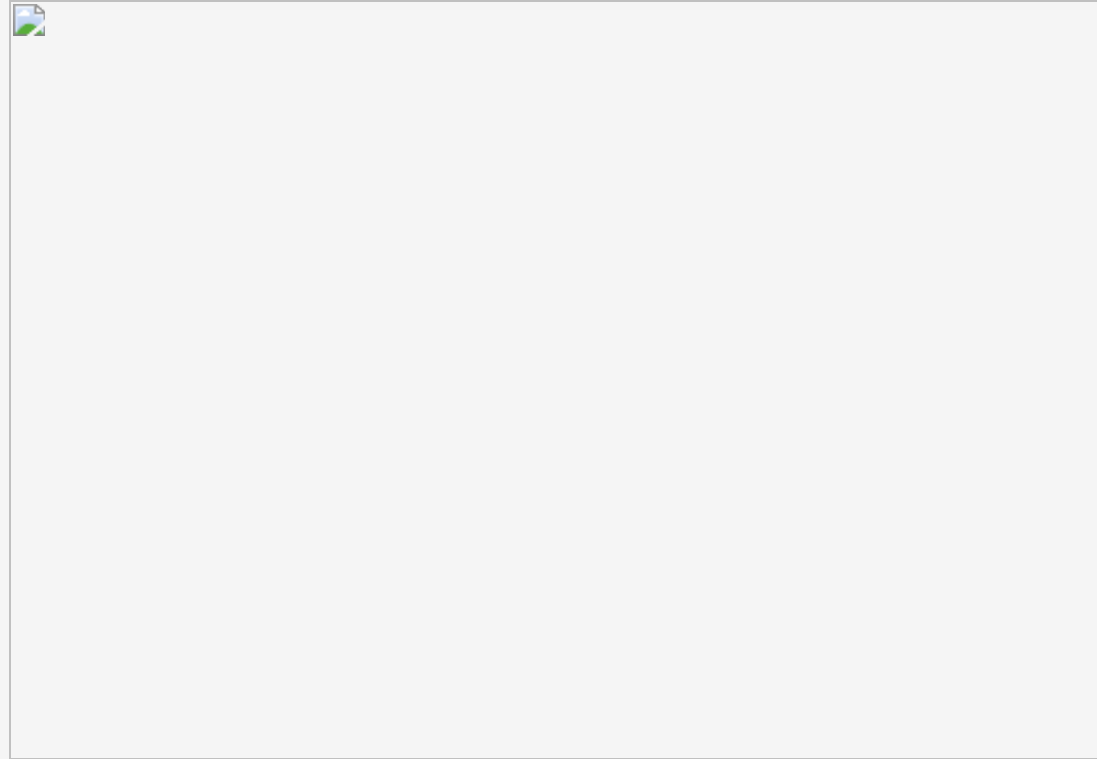
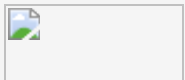


Figure: local usage



Calling procedure

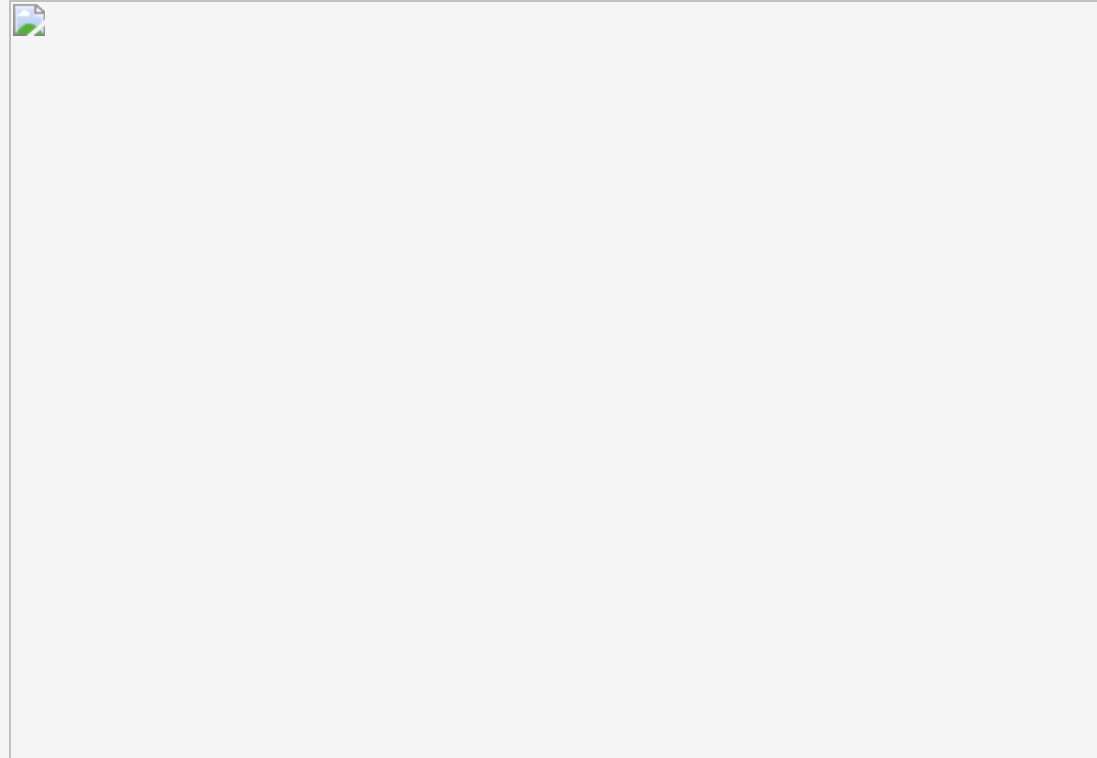
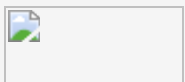


Figure: modernized usage



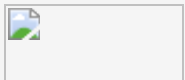
Summary

🌟 Pydantic let's you focus on your algorithm and not data model validation 🌟

Other cool tools for model validation

[Pandera](#), for data-validation on dataframe-like objects

GitHub 🌟 1.7k (currently, 18.10.22)



Demo repository

