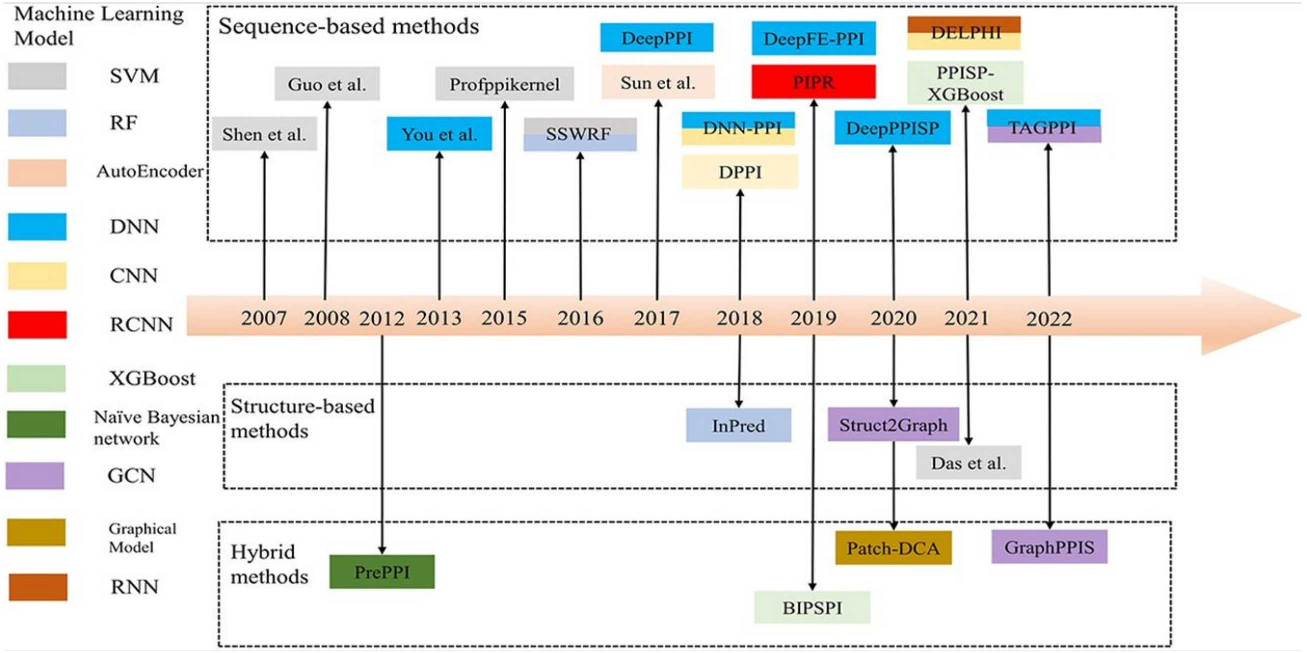
PPInsight: Automating Protein Interaction Benchmarking

Ike K., Maya G.H., Rita K., Fiona M., and Walter A.
CSE 583A

Background

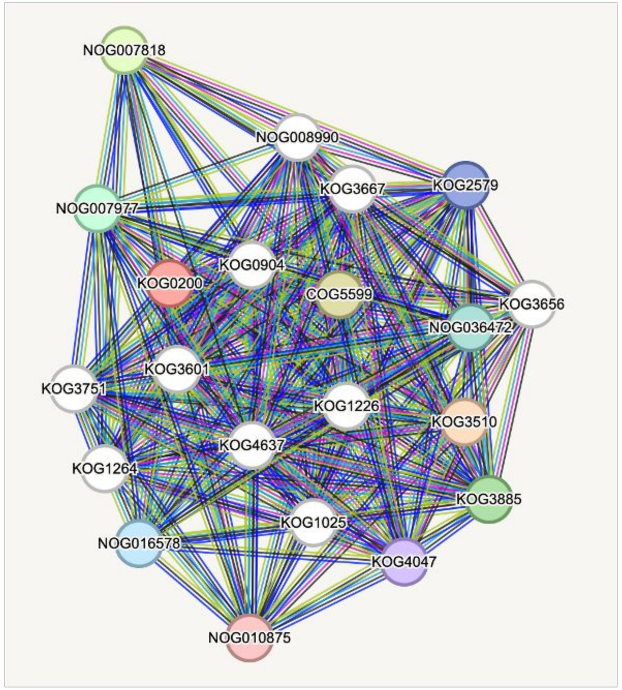


Tang et al, 2023, Briefings in Bioinformatics

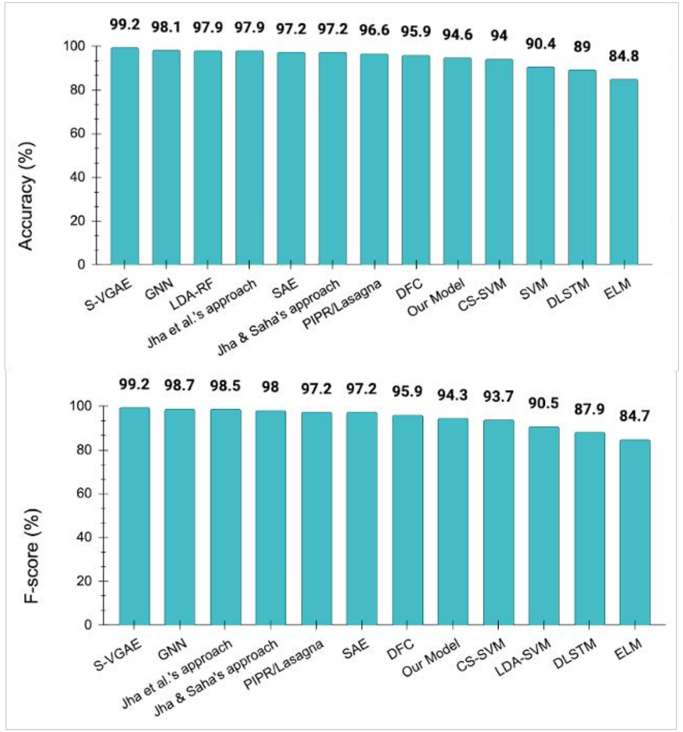
- **Protein–protein interactions** (PPIs) coordinate nearly all cellular processes
- PPI prediction involves the use of structural/sequential methods, limited by structure availability and challenges in capturing complex structural nuances

Background

via STRING



Graphical Classifiers Using Sequential and Structural Data Result in Greater Accuracy



- Researchers seek insight towards the computational boundaries and strengths of different model types

Proposed Approach:
A unified platform for standardized model benchmarking and clear performance comparison.

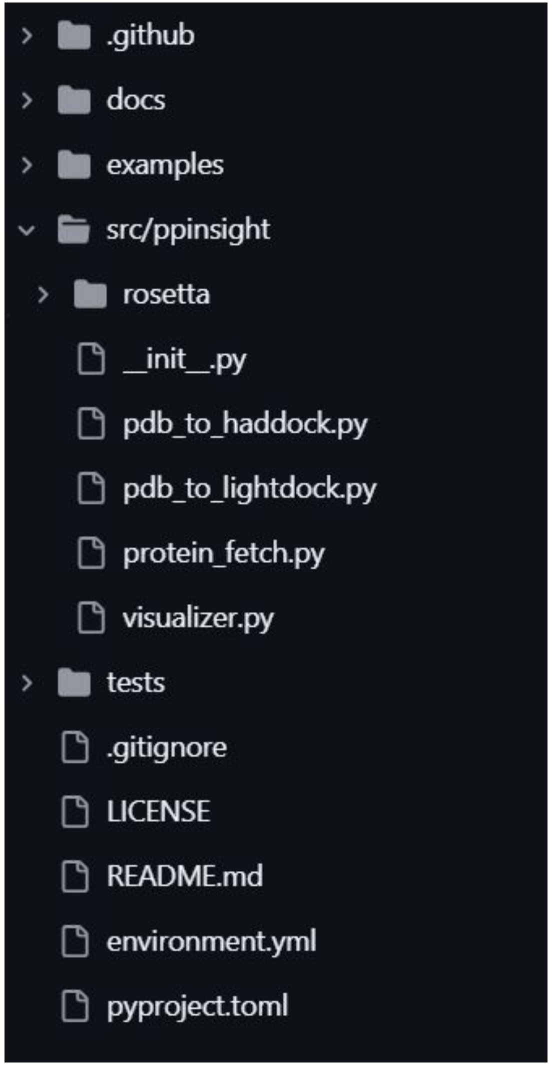
Motivating Use Case

Ana is a graduate student investigating whether two proteins interact under different cell stress conditions. She wants to quickly fetch sequence & structure data, run several PPI models, and compare their outputs in one place. Ana wants a workflow that reduces the manual steps for this process. She is comfortable with coding in Python.

Project Directory Structure

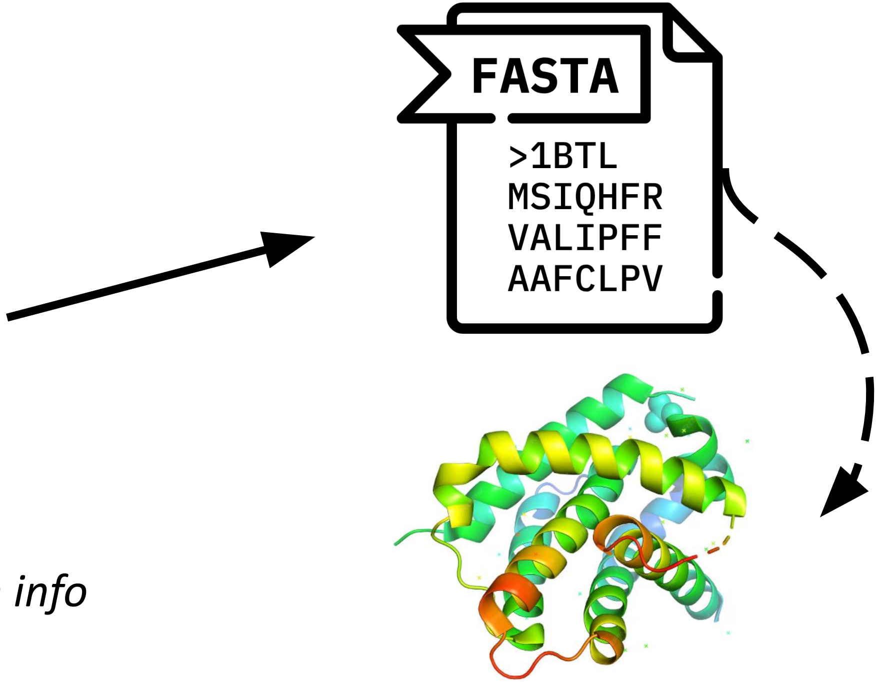
Name space management

```
>>> from ppinsight.rosetta import DockingPipeline

>>> from ppinsight.protein_fetch import get_uniprot_data

>>> from ppinsight.pdb_to_haddock import haddock_pipeline
```



```
> .github
> docs
> examples
v src/ppinsight
  > rosetta
    __init__.py
    pdb_to_haddock.py
    pdb_to_lightdock.py
    protein_fetch.py
    visualizer.py
> tests
  .gitignore
  LICENSE
  README.md
  environment.yml
  pyproject.toml
```

Data Used: Sequence & Structure Data

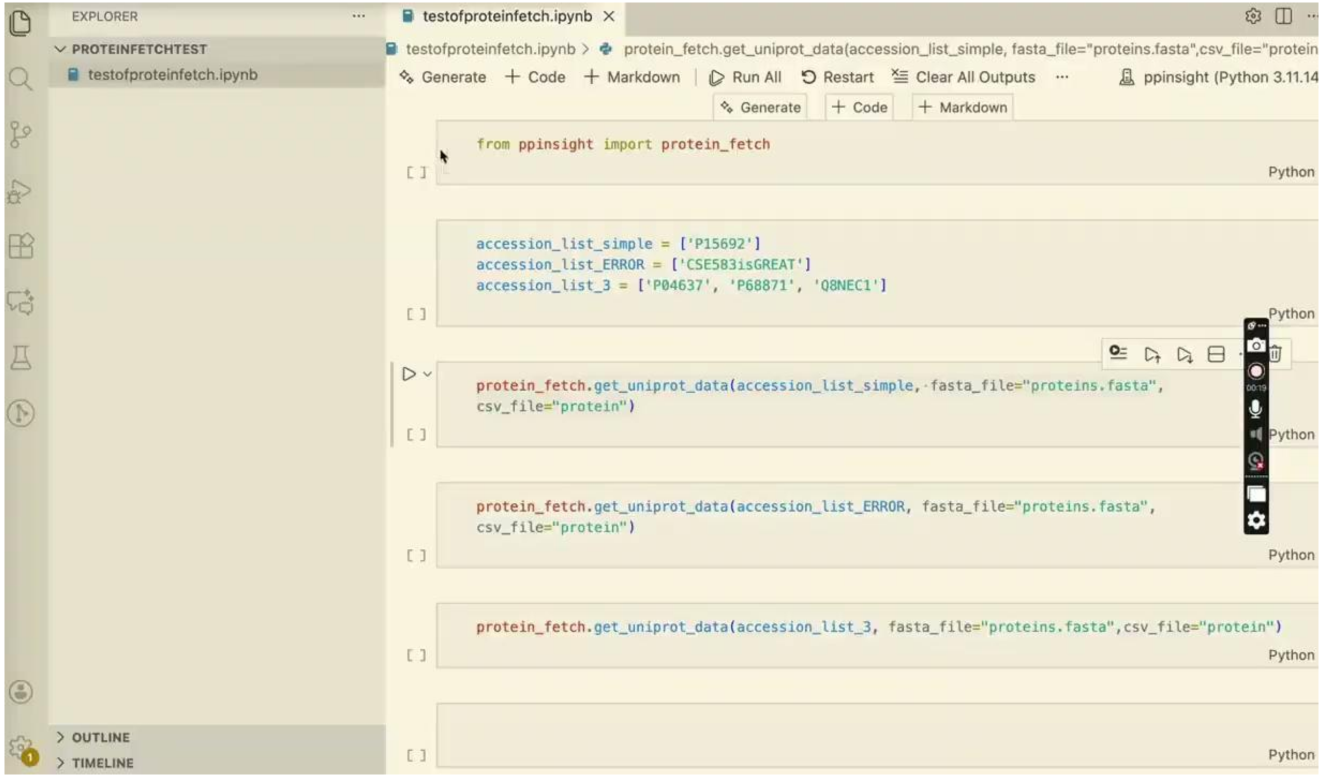


```
>1BTL
MSIQHFR
VALIPFF
AAFCLPV
```

UniProt: *Protein sequence and function info database*

PROTEIN DATA BANK

Stage 1: Fetching Protein Data



testofproteinfetch.ipynb > protein_fetch.get_uniprot_data(accession_list_simple, fasta_file="proteins.fasta",csv_file="protein

```
from ppinsight import protein_fetch
```

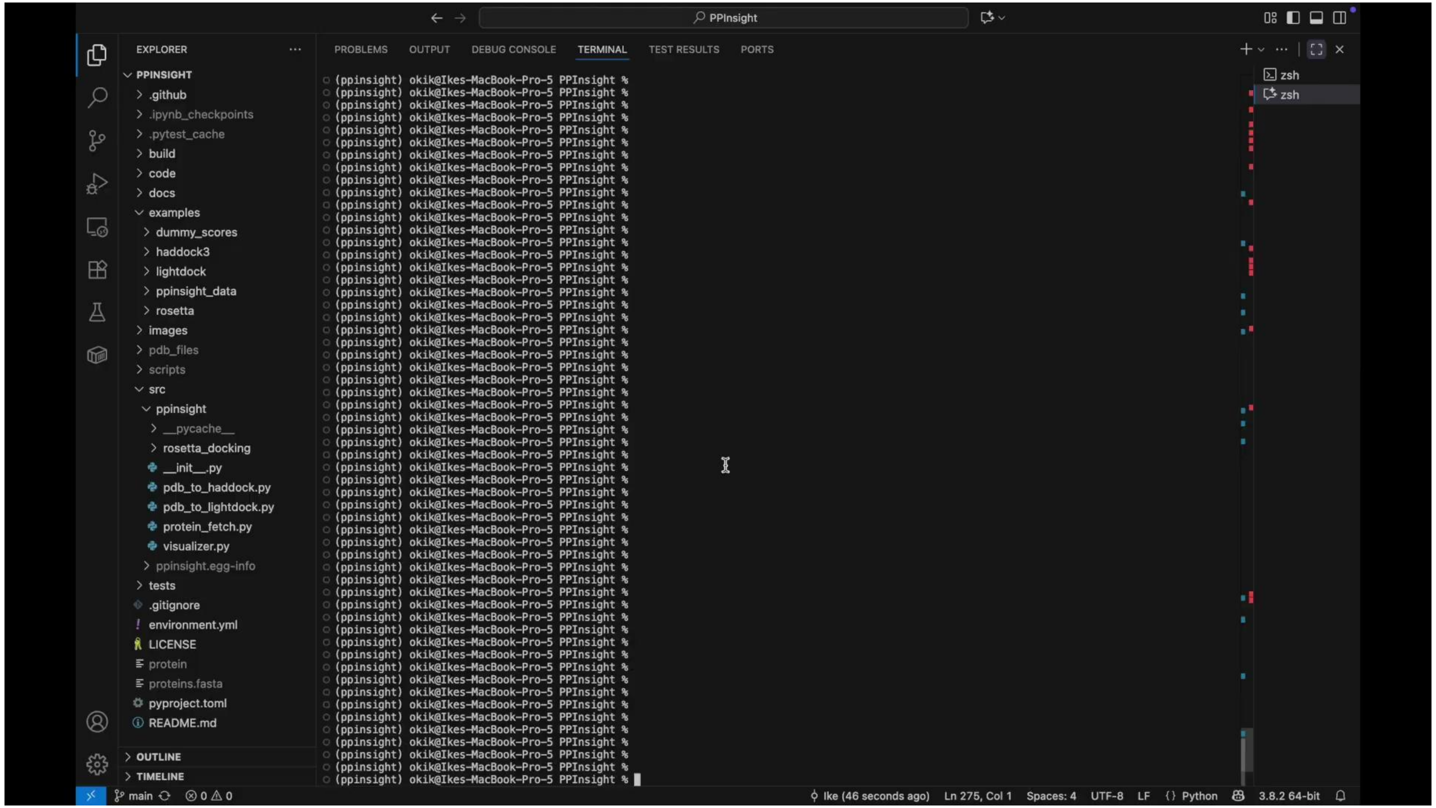
```
accession_list_simple = ['P15692']
accession_list_ERROR = ['CSE583isGREAT']
accession_list_3 = ['P04637', 'P68871', 'Q8NEC1']
```

```
protein_fetch.get_uniprot_data(accession_list_simple, fasta_file="proteins.fasta",
csv_file="protein")
```

```
protein_fetch.get_uniprot_data(accession_list_ERROR, fasta_file="proteins.fasta",
csv_file="protein")
```

```
protein_fetch.get_uniprot_data(accession_list_3, fasta_file="proteins.fasta",csv_file="protein")
```

Stage 2: Running PPI Predictors

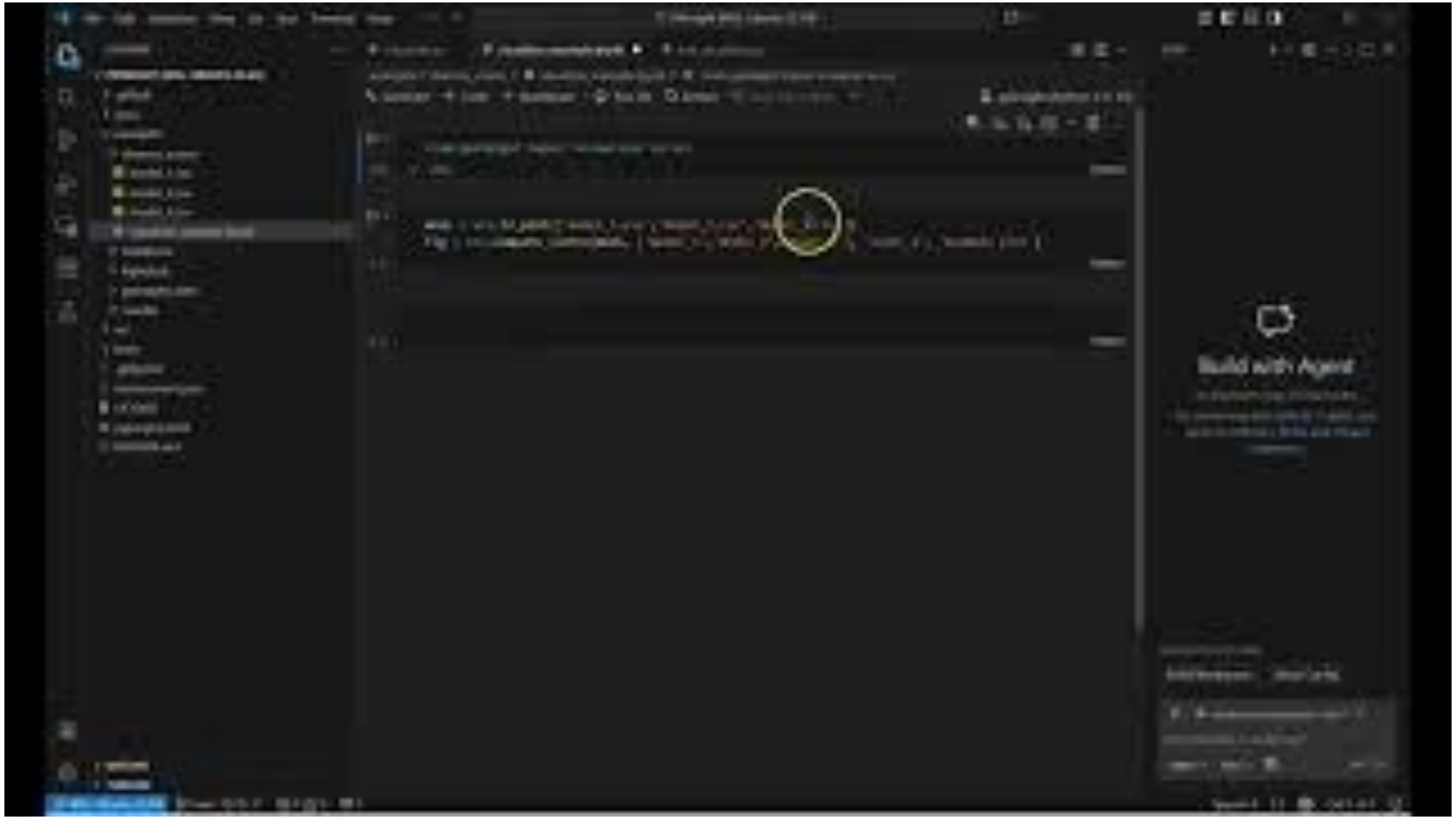


```
EXPLORER
PPINSIGHT
> .github
> .ipynb_checkpoints
> .pytest_cache
> build
> code
> docs
v examples
  > dummy_scores
  > haddock3
  > lightdock
  > ppinsight_data
  > rosetta
> images
> pdb_files
> scripts
v src
  v ppinsight
    > __pycache__
    > rosetta_docking
      __init__.py
      pdb_to_haddock.py
      pdb_to_lightdock.py
      protein_fetch.py
      visualizer.py
  > ppinsight.egg-info
> tests
  .gitignore
  environment.yml
  LICENSE
  protein
  proteins.fasta
  pyproject.toml
  README.md
```

```
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
(ppinsight) okik@Ikes-MacBook-Pro-5 PPInsight %
```

```
main  ⊗ 0 ⚠ 0    Ike (46 seconds ago)  Ln 275, Col 1  Spaces: 4  UTF-8  LF  {} Python  3.8.2 64-bit
```

Stage 3: Plotting PPI Confidence Metrics



Lessons Learned - SO MANY!

- Use of git(hub) to streamline communication
  - Code changes: pull requests
  - Task delegation and organization: git issue
- Project design
  - Needs → structure → delegation into parts
  - Parts integration: preemptive and continuous communication
  - Working with namespace to control user accessibility
  - Testing-first approach helps anticipating weaknesses
- Reading and working with official documentation & APIs
- Matching group members to tasks based on strengths and weaknesses

Lessons Learned - SO MANY!

- Use of git(hub) to streamline communication
  - Code changes: pull requests
  - Task delegation and organization: git issue
- Project design
  - Needs → structure → delegation into parts
  - Parts integration: preemptive and continuous communication
  - Working with namespace to control user accessibility
  - Testing-first approach helps anticipating weaknesses
- Reading and working with official documentation & APIs
- Matching group members to tasks based on strengths and weaknesses

Thank you Bryna, Dave, Elli and Ian for this wonderful course!

Future Work

- Parallel computation for Rosetta
- Build up connections between submodules
- Standardize output from different models
    - Directory and data structure
    - Naming conventions
- More interactive visualization