Hey, I’m Stefanie

- Open Source Developer at probabl since 2024
- Intern at scikit-learn in 2023
- Lecturer and TA at LeWagon in 2022-2023
- PhD in Contemporary History 2021

Connecting

https://www.linkedin.com/in/StefanieSenger
https://github.com/StefanieSenger

scikit learn

:probabl.

Scikit-learn's new
Metadata Routing API

Stefanie Senger

@StefanieSenger

PyLadies Berlin
January 14, 2025

:probabl.

What is metadata routing?

Introducing scikit-learn’s metadata routing API

Metadata

- Data to apply on top of tabular data
- Data a function can handle besides X and y

Use

- Fairness related, business logic, custom behaviour, etc.
- Benefit won’t be a better score, but a more realistic model

Routing

- Passing metadata between several components of a data science pipeline

What is metadata routing?

Introducing scikit-learn's metadata routing API

Prior to metadata routing API

- Restricted use of `sample_weight`
- No metadata passing in nested structures

With metadata routing API

- Nested `sample_weight` and `groups` passing
- Using metadata from other libraries' objects with scikit-learn estimators
- Custom definition of new metadata

Passing metadata without the routing API

In depth into the “groups” metadata in scikit-learn

Data

Sex	Age	Severity	Medication
Male	88	6	1
Female	100	4	0
Female	55	10	1
Female	70	7	1
Male	65	5	0
Male	68	0	1

Target

Recovery Time
33
9
10
25
70
29

Passing metadata without the routing API

In depth into the “groups” metadata in scikit-learn

Data

Sex	Age	Severity	Medication
Male	88	6	1
Female	100	4	0
Female	55	10	1
Female	70	7	1
Male	65	5	0
Male	68	0	1

groups

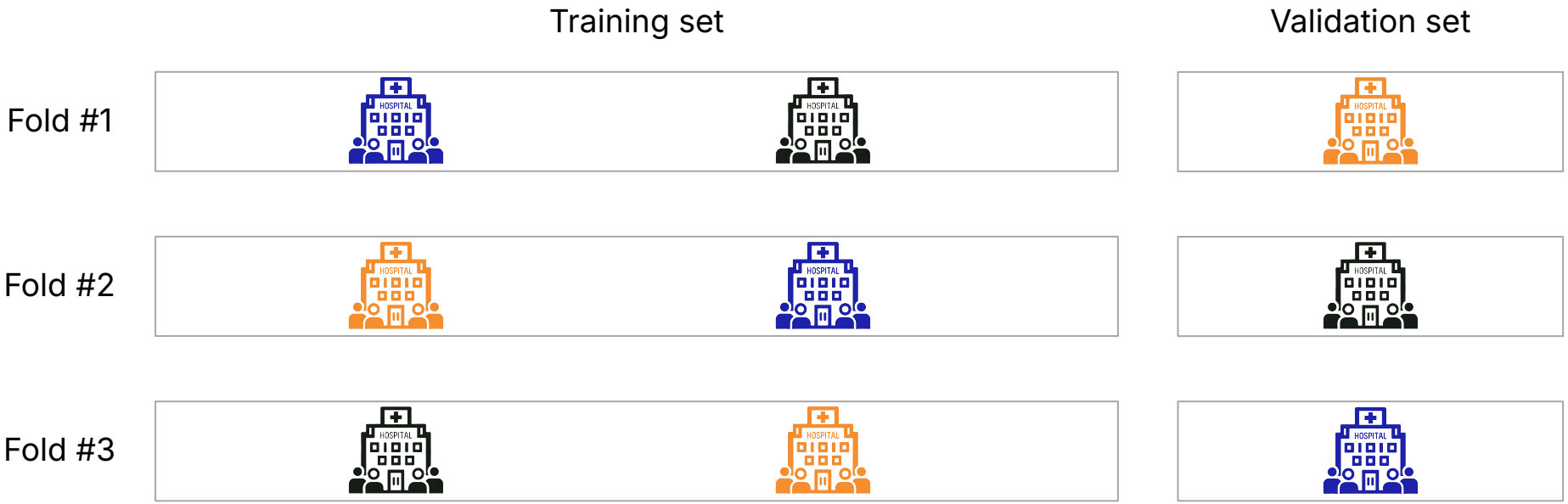
Hospital
Blue
Blue
Black
Black
Orange
Orange

Target

Recovery Time
33
9
10
25
70
29

Passing metadata without the routing API

Groups handling during cross-validation



Passing metadata without the routing API

In depth into the “groups” metadata in scikit-learn

```python
from sklearn.model_selection import GroupKFold, cross_validate
from sklearn.linear_model import Ridge

ridge = Ridge()

# GroupKFold considers groups while splitting:
cv = GroupKFold(n_splits=2)

# providing `groups` for GroupKFold splitter:
cross_validate(ridge, X, y, cv=cv, groups=groups)
```

```
{'fit_time': array([0.00135875, 0.00036836]),
 'score_time': array([0.00036907, 0.00023532]),
 'test_score': array([-0.2923336 , -0.12131859]}
```

Passing metadata without the routing API

Using the “sample_weight” metadata in scikit-learn

`sample_weight` draws a model’s attention to a sub-group of samples, in order to adjust their distribution (e.g. IPTW in medical settings)

- in distinction to a randomised trial, real-world observations often come with imbalanced and biased data

Sex	Age	Severity	Medication
Male	88	6	1
Female	100	4	0
Female	55	10	1

Recovery Time
33
9
10

Passing metadata without the routing API

Using the “sample_weight” metadata in scikit-learn

```
from sklearn.model_selection import cross_validate
from sklearn.linear_model import Ridge

# Ridge().fit() can consume `sample_weight`
ridge = Ridge()

cross_validate(ridge, X, y, cv=cv, sample_weight=sample_weight)
```

```
TypeError: got an unexpected keyword argument ‘sample_weight’
```

Passing metadata without the routing API

```
ridge = Ridge() # can consume sample_weight

cv=GroupKFold(n_splits=2) # consumes groups
scoring = get_scorer("neg_mean_squared_error") # can consume sample_weight


search = GridSearchCV(
  ridge, param_grid={"alpha": [0.1, 1, 10]}, cv=cv, scoring=scoring
)

cross_validate(
   search, X, y, cv=cv, scoring=scoring,
   sample_weight=sample_weight, groups=groups,
)
```

- Cross-validation on a `GridSearchCV` that searches `Ridge` hyperparameters
- Passing `sample_weight` and `groups` into it

```
TypeError: got an unexpected keyword argument 'sample_weight'
```

Using the metadata routing API

```
ridge = Ridge()

cv=GroupKFold(n_splits=2)
scoring = get_scorer("neg_mean_squared_error")


search = GridSearchCV(
  ridge, param_grid={"alpha": [0.1, 1, 10]}, cv=cv, scoring=scoring
)

cross_validate(
   search, X, y, cv=cv, scoring=scoring,
   params={"sample_weight": sample_weight, "groups": groups},
)
```

Using the metadata routing API

```
import sklearn
sklearn.set_config(enable_metadata_routing=True)
ridge = Ridge()

cv=GroupKFold(n_splits=2)
scoring = get_scorer("neg_mean_squared_error")



search = GridSearchCV(
  ridge, param_grid={"alpha": [0.1, 1, 10]}, cv=cv, scoring=scoring
)

cross_validate(
   search, X, y, cv=cv, scoring=scoring,
   params={"sample_weight": sample_weight, "groups": groups},
)
```

1. Enable metadata routing

Using the metadata routing API

```
import sklearn
sklearn.set_config(enable_metadata_routing=True)
ridge = Ridge()

cv=GroupKFold(n_splits=2)
scoring = get_scorer("neg_mean_squared_error")



search = GridSearchCV(
  ridge, param_grid={"alpha": [0.1, 1, 10]}, cv=cv, scoring=scoring
)

cross_validate(
   search, X, y, cv=cv, scoring=scoring,
   params={"sample_weight": sample_weight, "groups": groups},
)
```

1. Enable metadata routing
2. Pass metadata to top level tool

Using the metadata routing API

```
import sklearn
sklearn.set_config(enable_metadata_routing=True)
ridge = Ridge().set_fit_request(sample_weight=True)

cv=GroupKFold(n_splits=2)
scoring = get_scorer("neg_mean_squared_error")
scoring.set_score_request(sample_weight=True)

search = GridSearchCV(
  ridge, param_grid={"alpha": [0.1, 1, 10]}, cv=cv, scoring=scoring
)

cross_validate(
   search, X, y, cv=cv, scoring=scoring,
   params={"sample_weight": sample_weight, "groups": groups},
)
```

1. Enable metadata routing
2. Pass metadata to top level tool
3. Set requests where metadata is consumed

Voilà: Routing Success!

Metadata routing API

Ongoing work

- Expect compatibility for all scikit-learn estimators in 1.7 release
- Defining default settings to relieve users from setting all the requests
- Keep flexibility for custom scenarios

Metadata routing API

References and Links

Metadata routing

- User Guide on Metadata Routing
- Developer Guide on Metadata Routing
- Adrin Jalali's talk on the internal logic of metadata routing

Use cases using `sample_weight`

- :probabl. Whiteboard Series by Vincent Warmerdam: Improving models via subsets
- Blogpost by Florian Wilhelm on Inverse Probability of Treatment Weighting

Metadata routing API

Thank you for your attention!

Time for your questions

Connecting

https://www.linkedin.com/in/StefanieSenger
https://github.com/StefanieSenger