

# Mining Documentation to Extract Hyperparameter Schemas

Guillaume Baudart, Peter D. Kirchner, Martin Hirzel, Kiran Kate  
*IBM Research*

# Mining Documentation...

## `sklearn.linear_model.LogisticRegression`

```
class sklearn.linear_model.LogisticRegression(penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True,
intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0,
warm_start=False, n_jobs=None, l1_ratio=None)
```

[\[source\]](#)

Logistic Regression (aka logit, MaxEnt) classifier.

### Parameters:

**penalty** : {'l1', 'l2', 'elasticnet', 'none'}, default='l2'

Used to specify the norm used in the penalization. The 'newton-cg', 'sag' and 'lbfgs' solvers support only l2 penalties. 'elasticnet' is only supported by the 'saga' solver. If 'none' (not supported by the liblinear solver), no regularization is applied.

*New in version 0.19:* l1 penalty with SAGA solver (allowing 'multinomial' + L1)

**class\_weight** : dict or 'balanced', default=None

Weights associated with classes in the form {class\_label: weight}. If not given, all classes are supposed to have weight one.

## Challenges

- Variability: multiple formats, typos, ...
- Constraints are expressed in natural language

# Mining Documentation...

## `sklearn.linear_model.LogisticRegression`

```
class sklearn.linear_model.LogisticRegression(penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True,
intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0,
warm_start=False, n_jobs=None, l1_ratio=None)
```

[\[source\]](#)

Logistic Regression (aka logit, MaxEnt) classifier.

### Parameters:

**penalty** : `{'l1', 'l2', 'elasticnet', 'none'}`

Used to specify the norm used in the penalization. The 'newton-cg', 'sag' and 'lbfgs' solvers support only l2 penalties. 'elasticnet' is only supported by the 'saga' solver. If 'none' (not supported by the liblinear solver), no regularization is applied.

*New in version 0.19:* l1 penalty with SAGA solver (allowing 'multinomial' + L1)

**class\_weight** : `dict` *or* `'balanced'` `None`

Weights associated with classes in the form `{class_label: weight}`. If not given, all classes are supposed to have weight one.

## Challenges

- Variability: multiple formats, typos, ...
- Constraints are expressed in natural language

# Mining Documentation...

## `sklearn.linear_model.LogisticRegression`

```
class sklearn.linear_model.LogisticRegression(penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True,
intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0,
warm_start=False, n_jobs=None, l1_ratio=None)
```

[\[source\]](#)

Logistic Regression (aka logit, MaxEnt) classifier.

### Parameters:

**penalty :** `{'l1', 'l2', 'elasticnet', 'none'}`

Used to specify the norm used in the penalization. The 'newton-cg', 'saq' and 'lbfgs' solvers support only l2 penalties. 'elasticnet' is only supported by the 'saga' solver (supported by the liblinear solver), no regularization is applied.

*New in version 0.19:* l1 penalty with SAGA solver (allowing 'multinomial' + L1)

**class\_weight :** `dict` **or** `'balanced'` `None`

Weights associated with classes in the form `{class_label: weight}`. If not given, all classes are supposed to have weight one.

## Challenges

- Variability: multiple formats, typos, ...
- Constraints are expressed in natural language

# ... to Extract Hyperparameter Schemas

## Machine Readable JSON Schemas

- Can be compiled to search spaces for multiple AutoML tools
- Expressive enough for cross-parameter constraints

```
{ '$schema': 'http://json-schema.org/draft-04/schema#',  
  'description': 'Hyperparameter schema.',  
  'allOf': [  
    { 'type': 'object',  
      'additionalProperties': False,  
      'required': ['penalty', 'dual', 'tol', ... ],  
      'relevantToOptimizer': ['penalty', 'dual', 'tol', ... ],  
      'properties': {  
        'penalty': {  
          'description': 'Norm used in the penalization.',  
          'enum': ['l1', 'l2'],  
          'default': 'l2'},  
        'class_weight': {  
          'description': 'Weights associated with classes',  
          'anyOf': [  
            { 'description': 'Adjust weights by inverse frequency.',  
              'enum': ['balanced']},  
            { 'description': 'Dictionary mapping class labels to weights.',  
              'type': 'object',  
              'propertyNames': {'pattern': '^.+$', 'type': 'number'},  
              'forOptimizer': False}],  
          'default': None}    ]
```

# ... to Extract Hyperparameter Schemas

## Machine Readable JSON Schemas

- Can be compiled to search spaces for multiple AutoML tools
- Expressive enough for cross-parameter constraints

```
{ '$schema': 'http://json-schema.org/draft-04/schema#',  
  'description': 'Hyperparameter schema.',  
  'allOf': [  
    { 'type': 'object',  
      'additionalProperties': false,  
      'required': ['penalty'],  
      'relevantToOptimizer': 'solver',  
      'properties': {  
        'penalty': {  
          'description': 'The type of regularization to use.',  
          'enum': ['l1', 'l2'],  
          'default': 'l2',  
          'class_weight': 'balanced',  
          'description': 'The type of regularization to use.',  
          'anyOf': [  
            { 'description': 'Adjust weights by inverse frequency.',  
              'enum': ['balanced']},  
            { 'description': 'Dictionary mapping class labels to weights.',  
              'type': 'object',  
              'propertyNames': {'pattern': '^.+$', 'type': 'number'},  
              'forOptimizer': False}],  
          'default': None}
```

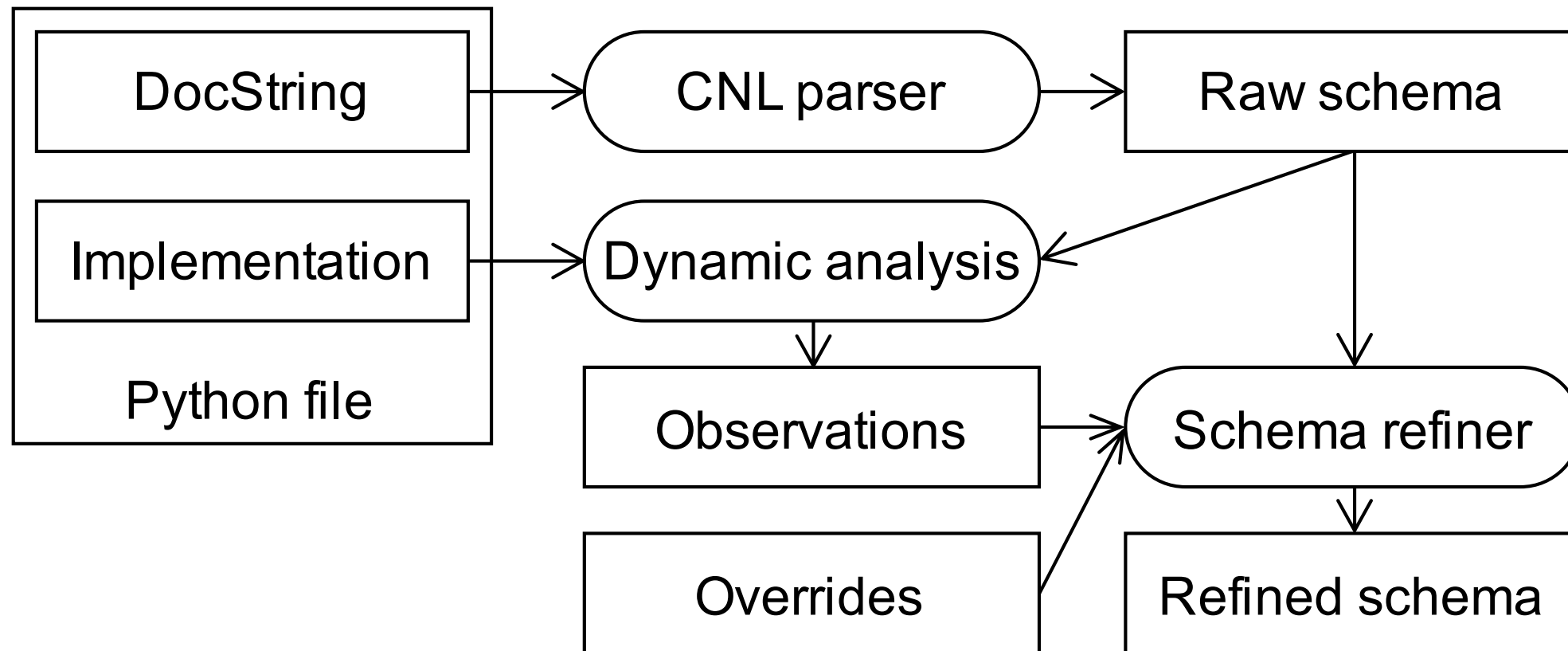
### Constraint example

```
{ 'description':  
  'The newton-cg, sag, and lbfgs solvers support only l2 penalties.',  
  'anyOf': [  
    { 'type': 'object',  
      'properties': {  
        'solver': {'not': {'enum': ['newton-cg', 'sag', 'lbfgs']}}},  
    { 'type': 'object',  
      'properties': {'penalty': {'enum': ['l2']}}}],
```

# Our Approach

Two sources of truth: documentation & source code

- Controlled natural language parser: Mine high-quality documentation
- Dynamic analysis: Analyze the code to refine the schema





# Evaluation

**Complete Dataset: 115 SKLearn, 2 XGBoost, 2 LightGBM**

- Types: 94% of 1,758 parameters
- Ranges: 50% of 790 parameters (numeric types, enums)
- Constraints: flagged 118, compiled 43.

**Curated Dataset: 38 SKLearn, 2 XGBoost, 2 LightGBM**

- Types: 81% of 452 parameters
- Ranges: 81% of 103 parameters (numeric types, enums)
- Constraints: flagged 50, compiled 20 of 65 constraints

**AutoML Pipeline: preprocess → features → classifier**

- Use our schemas to tune hyperparameters for 15 OpenML tasks
- Accuracies are comparable to Auto-SKLearn





<https://github.com/IBM/laie>