Introdução à Seleção de Modelos

Prof. Jefferson T. Oliva

Reconhecimento de Padrões (RC18EE)
Engenharia de Computação
Programa de Pós-Graduação em Engenharia Elétrica e de Computação (PPGEEC)
Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Pato Branco

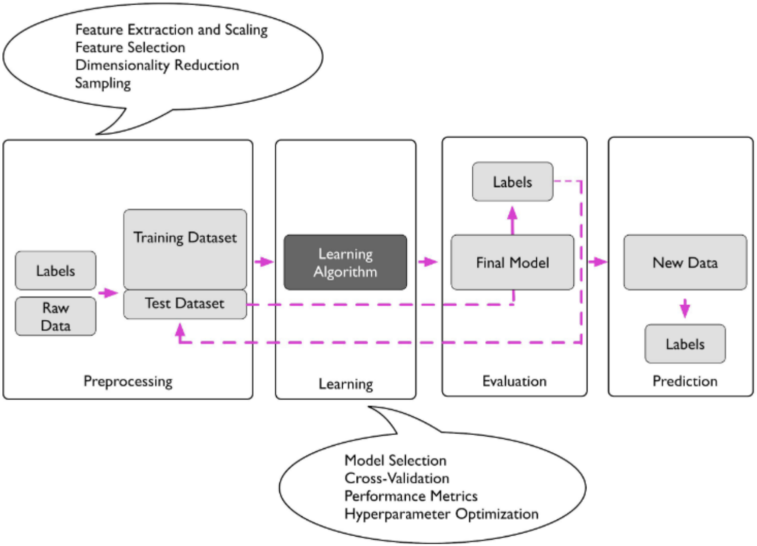
DAINF
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CÂMPUS PATO BRANCO

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Sumário

- Geração de Pipelines

Introdução

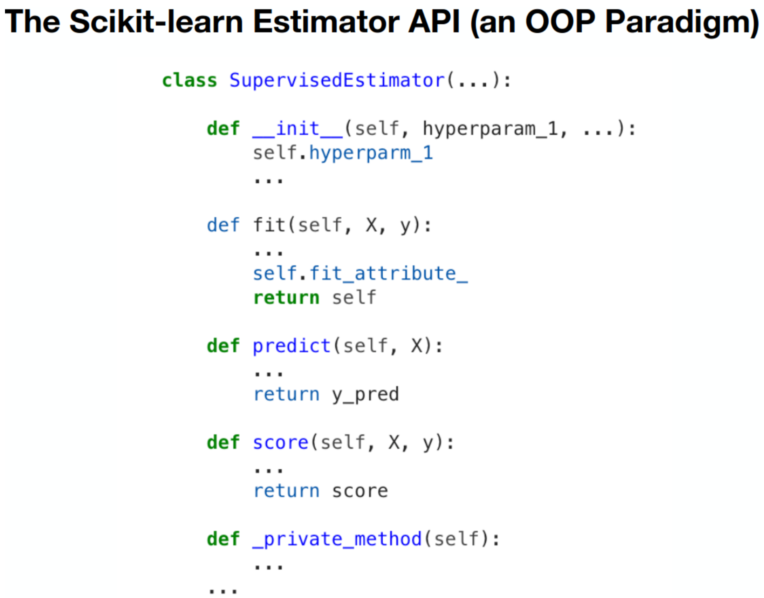


3

Sumário

Geração de Pipelines

Geração de Pipelines

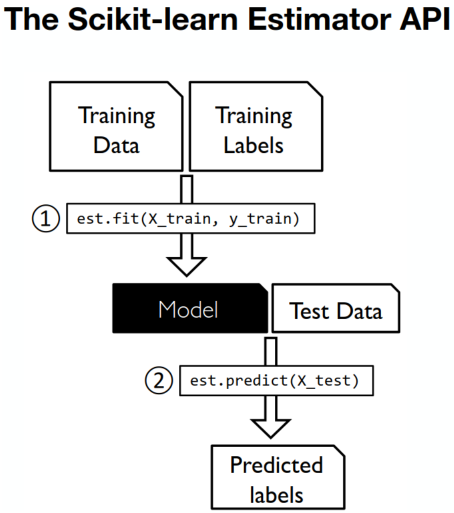


The Scikit-learn Estimator API (an OOP Paradigm)

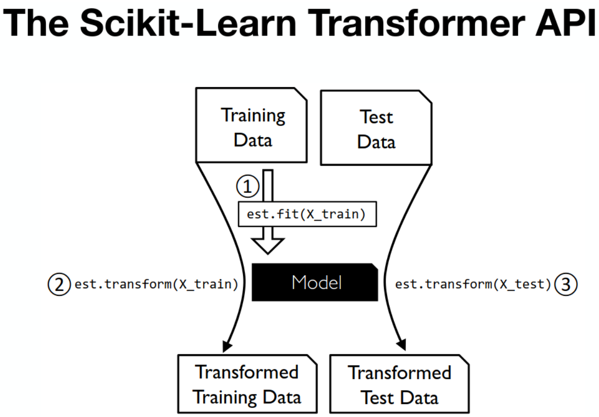
```
class SupervisedEstimator(...):

    def __init__(self, hyperparam_1, ...):
        self.hyperparm_1
        ...

    def fit(self, X, y):
        ...
        self.fit_attribute_
        return self

    def predict(self, X):
        ...
        return y_pred

    def score(self, X, y):
        ...
        return score

    def _private_method(self):
        ...
    ...
```

5

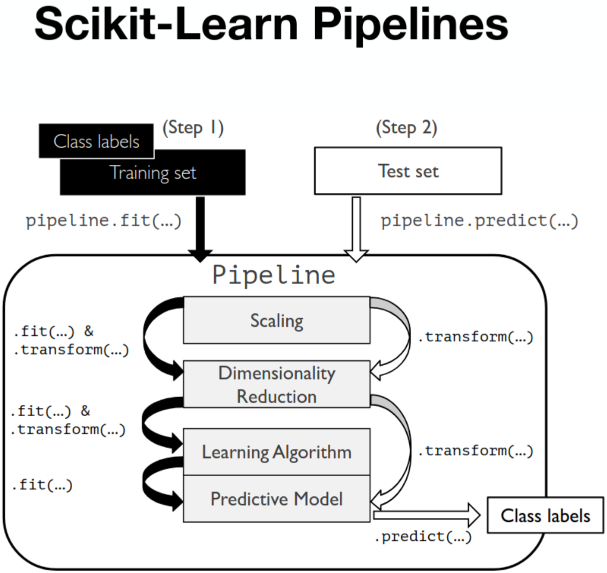
Geração de Pipelines



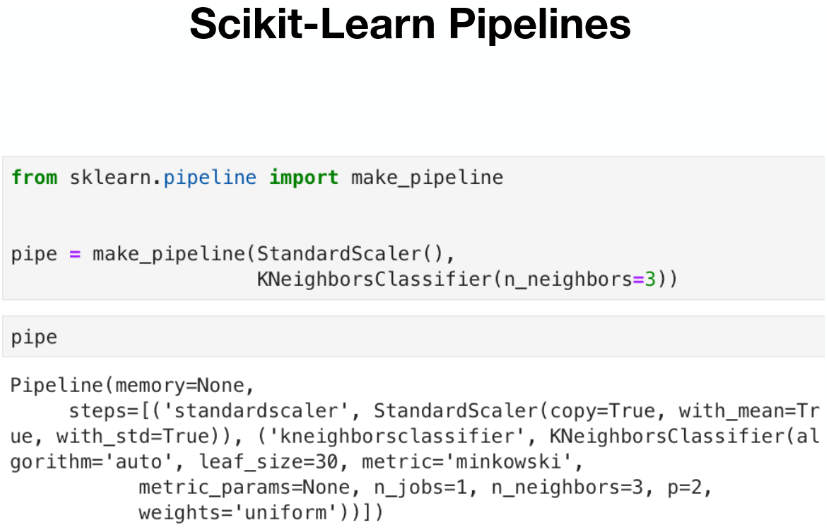
Geração de Pipelines



Geração de Pipelines



Geração de Pipelines



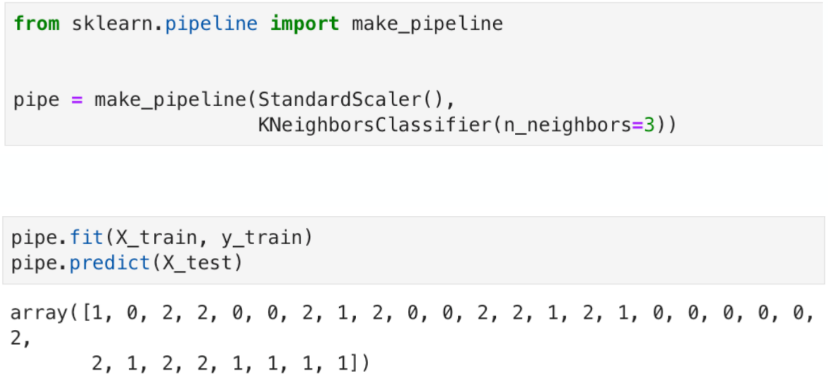
Scikit-Learn Pipelines

```python
from sklearn.pipeline import make_pipeline


pipe = make_pipeline(StandardScaler(),
                     KNeighborsClassifier(n_neighbors=3))
```

```python
pipe
```

```
Pipeline(memory=None,
     steps=[('standardscaler', StandardScaler(copy=True, with_mean=Tr
ue, with_std=True)), ('kneighborsclassifier', KNeighborsClassifier(al
gorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=3, p=2,
           weights='uniform'))])
```

Geração de Pipelines

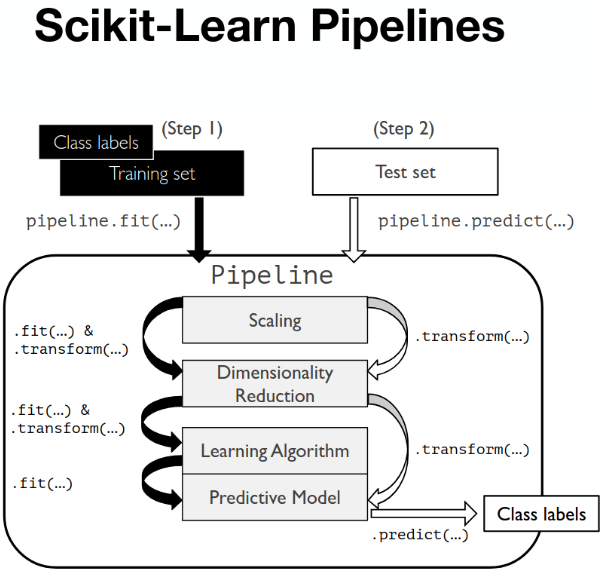


```python
from sklearn.pipeline import make_pipeline


pipe = make_pipeline(StandardScaler(),
                     KNeighborsClassifier(n_neighbors=3))
```

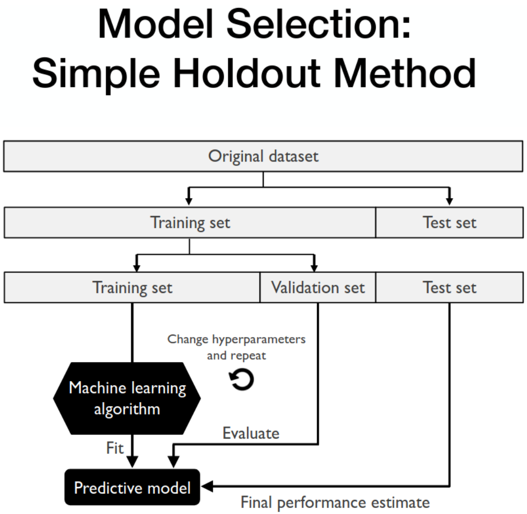
```python
pipe.fit(X_train, y_train)
pipe.predict(X_test)
```

```
array([1, 0, 2, 2, 0, 0, 2, 1, 2, 0, 0, 2, 2, 1, 2, 1, 0, 0, 0, 0, 0,
2,
       2, 1, 2, 2, 1, 1, 1, 1])
```

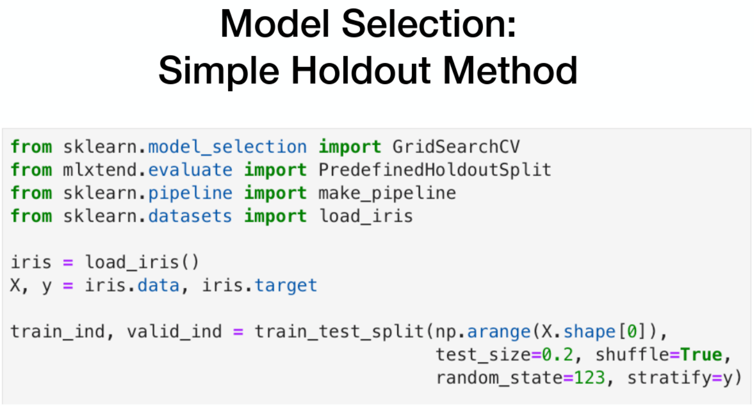
Geração de Pipelines



Geração de Pipelines



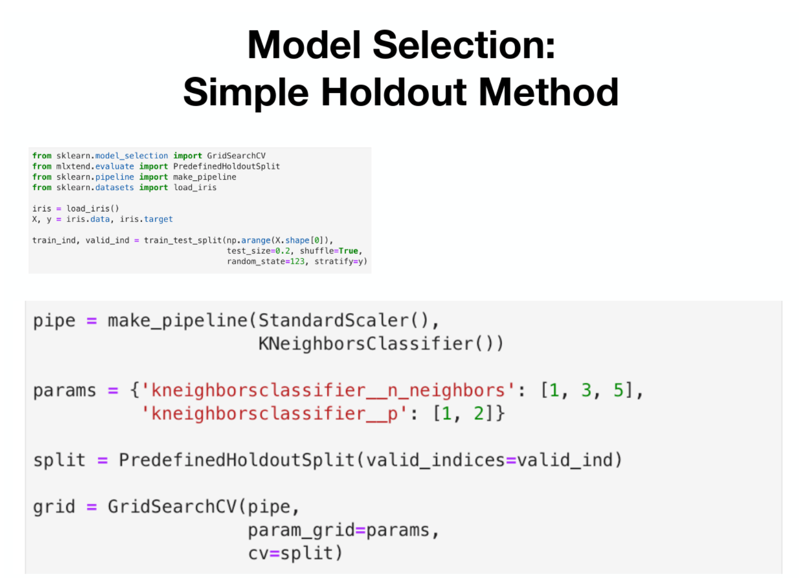
Geração de Pipelines



Model Selection: Simple Holdout Method

```python
from sklearn.model_selection import GridSearchCV
from mlxtend.evaluate import PredefinedHoldoutSplit
from sklearn.pipeline import make_pipeline
from sklearn.datasets import load_iris

iris = load_iris()
X, y = iris.data, iris.target

train_ind, valid_ind = train_test_split(np.arange(X.shape[0]),
                                        test_size=0.2, shuffle=True,
                                        random_state=123, stratify=y)
```

Geração de Pipelines



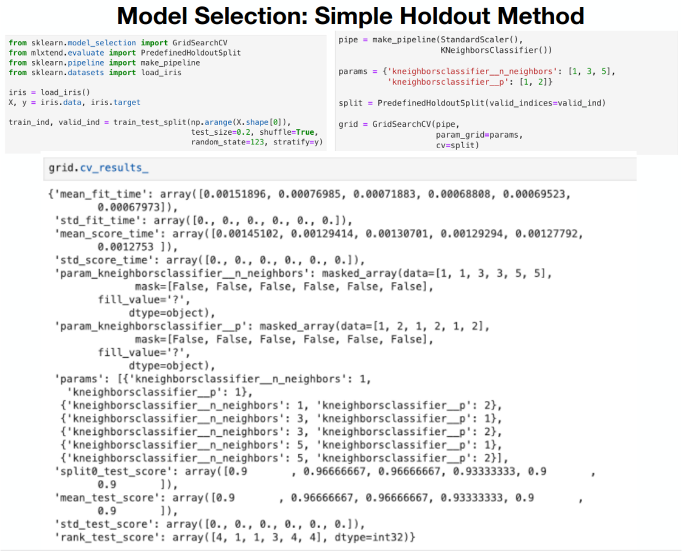
Model Selection: Simple Holdout Method

```python
from sklearn.model_selection import GridSearchCV
from mlxtend.evaluate import PredefinedHoldoutSplit
from sklearn.pipeline import make_pipeline
from sklearn.datasets import load_iris

iris = load_iris()
X, y = iris.data, iris.target

train_ind, valid_ind = train_test_split(np.arange(X.shape[0]),
                                        test_size=0.2, shuffle=True,
                                        random_state=123, stratify=y)
```

```python
pipe = make_pipeline(StandardScaler(),
                     KNeighborsClassifier())

params = {'kneighborsclassifier__n_neighbors': [1, 3, 5],
          'kneighborsclassifier__p': [1, 2]}

split = PredefinedHoldoutSplit(valid_indices=valid_ind)

grid = GridSearchCV(pipe,
                    param_grid=params,
                    cv=split)
```

14

Geração de Pipelines



Model Selection: Simple Holdout Method

```
from sklearn.model_selection import GridSearchCV
from mlxtend.evaluate import PredefinedHoldoutSplit
from sklearn.pipeline import make_pipeline
from sklearn.datasets import load_iris

iris = load_iris()
X, y = iris.data, iris.target

train_ind, valid_ind = train_test_split(np.arange(X.shape[0]),
                                        test_size=0.2, shuffle=True,
                                        random_state=123, stratify=y)
```

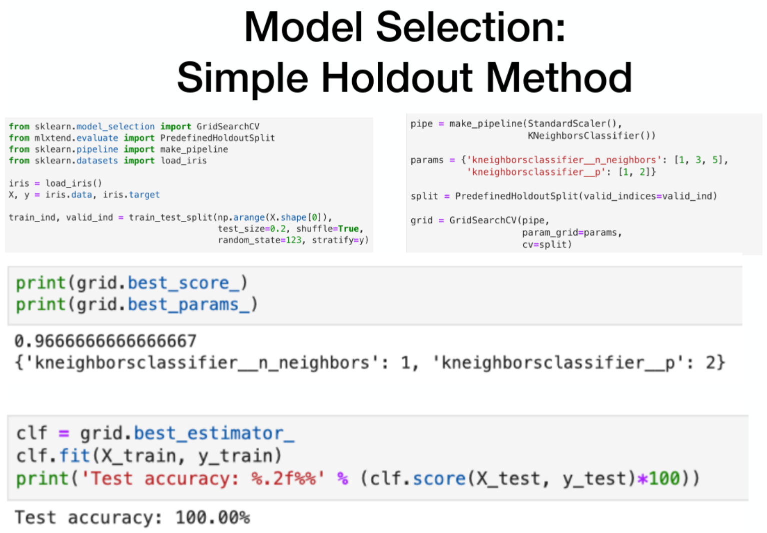
```
pipe = make_pipeline(StandardScaler(),
                     KNeighborsClassifier())

params = {'kneighborsclassifier__n_neighbors': [1, 3, 5],
          'kneighborsclassifier__p': [1, 2]}

split = PredefinedHoldoutSplit(valid_indices=valid_ind)

grid = GridSearchCV(pipe,
                    param_grid=params,
                    cv=split)
```

```
grid.cv_results_
```

```
{'mean_fit_time': array([0.00151896, 0.00076985, 0.00071883, 0.00068808, 0.00069523,
        0.00067973]),
 'std_fit_time': array([0., 0., 0., 0., 0., 0.]),
 'mean_score_time': array([0.00145102, 0.00129414, 0.00130701, 0.00129294, 0.00127792,
        0.0012753 ]),
 'std_score_time': array([0., 0., 0., 0., 0., 0.]),
 'param_kneighborsclassifier__n_neighbors': masked_array(data=[1, 1, 3, 3, 5, 5],
              mask=[False, False, False, False, False, False],
        fill_value='?',
             dtype=object),
 'param_kneighborsclassifier__p': masked_array(data=[1, 2, 1, 2, 1, 2],
              mask=[False, False, False, False, False, False],
        fill_value='?',
             dtype=object),
 'params': [{'kneighborsclassifier__n_neighbors': 1,
   'kneighborsclassifier__p': 1},
  {'kneighborsclassifier__n_neighbors': 1, 'kneighborsclassifier__p': 2},
  {'kneighborsclassifier__n_neighbors': 3, 'kneighborsclassifier__p': 1},
  {'kneighborsclassifier__n_neighbors': 3, 'kneighborsclassifier__p': 2},
  {'kneighborsclassifier__n_neighbors': 5, 'kneighborsclassifier__p': 1},
  {'kneighborsclassifier__n_neighbors': 5, 'kneighborsclassifier__p': 2}],
 'split0_test_score': array([0.9       , 0.96666667, 0.96666667, 0.93333333, 0.9       ,
        0.9       ]),
 'mean_test_score': array([0.9       , 0.96666667, 0.96666667, 0.93333333, 0.9       ,
        0.9       ]),
 'std_test_score': array([0., 0., 0., 0., 0., 0.]),
 'rank_test_score': array([4, 1, 1, 3, 4, 4], dtype=int32)}
```

15

Geração de Pipelines



Model Selection: Simple Holdout Method

```python
from sklearn.model_selection import GridSearchCV
from mlxtend.evaluate import PredefinedHoldoutSplit
from sklearn.pipeline import make_pipeline
from sklearn.datasets import load_iris

iris = load_iris()
X, y = iris.data, iris.target

train_ind, valid_ind = train_test_split(np.arange(X.shape[0]),
                                        test_size=0.2, shuffle=True,
                                        random_state=123, stratify=y)
```

```python
pipe = make_pipeline(StandardScaler(),
                     KNeighborsClassifier())

params = {'kneighborsclassifier__n_neighbors': [1, 3, 5],
          'kneighborsclassifier__p': [1, 2]}

split = PredefinedHoldoutSplit(valid_indices=valid_ind)

grid = GridSearchCV(pipe,
                    param_grid=params,
                    cv=split)
```

```python
print(grid.best_score_)
print(grid.best_params_)
```

```
0.9666666666666667
{'kneighborsclassifier__n_neighbors': 1, 'kneighborsclassifier__p': 2}
```

```python
clf = grid.best_estimator_
clf.fit(X_train, y_train)
print('Test accuracy: %.2f%%' % (clf.score(X_test, y_test)*100))
```

```
Test accuracy: 100.00%
```

Referências I

CASANOVA, D.
Intro to model selection. Aprendizado de Máquina.
Slides. Engenharia de Computação. Dainf/UTFPR, 2020.