Pythonではじめる機械学習

6章　アルゴリズムチェーンとパイプライン

6.4　汎用パイプラインインターフェイス

6.4.1　make_pipelineによる簡便なパイプライン生成

6.4.2　ステップ属性へのアクセス

6.4.3　GridSearchCV内のパイプラインの属性へのアクセス

15T406IL　南濱　篤

6.4 汎用パイプラインインターフェイス

- Pipelineクラスの適用について、任意個数のEstimatorを連結可能。

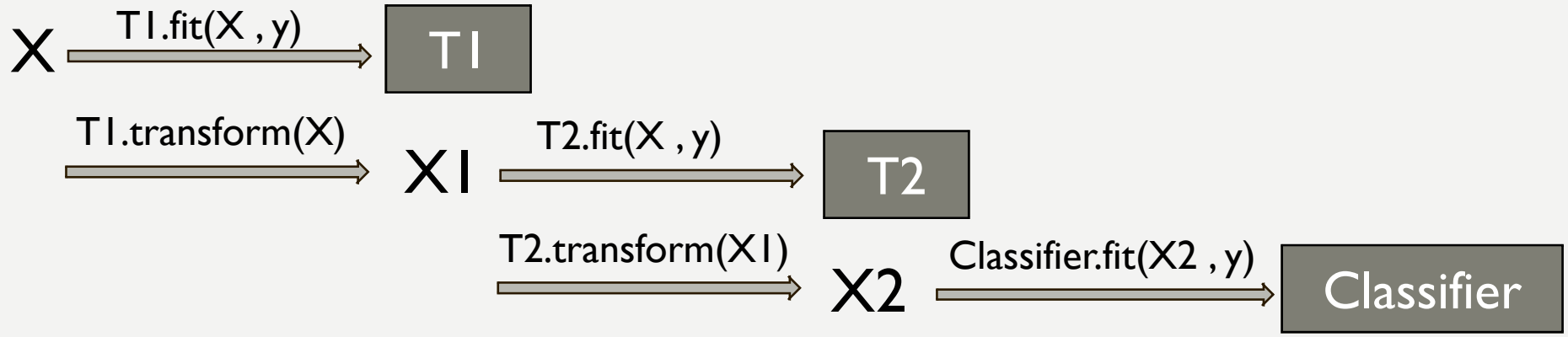
- Estimatorに関する制約
  最後以外のステップには次で使うデータの新しい表現を生成するために、transformメソッドが定義されている必要がある。

- 内部的に、直前のtransformメソッドの値の出力を入力とし、fitとtransformの順に呼び出し、これを繰り返す。
  最後は、fitメソッドだけが呼ばれる。

6.4 汎用パイプラインインターフェイス

```
def fit(self, X, y):
    X_transformed = X
    for name, estimator in self.steps[:-1]:
      X_transformed = estimator.fit_transform(X_transformed, y)
    self.steps[-1][1].fit(X_transformed, y)
    return self
```

pipe.fit(X , y)



6.4 汎用パイプラインインターフェイス

```
def predict(self, X):
    X_transformed = X
    for step in self.steps[:-1]:
      X_transformed = step[1].transform(X_transformed)
    return self.steps[-1][1].predict(X_transformed)
```

pipe.predict(X')

6.4.1 make_pipelineによる簡便なパイプライン生成

- make_pipeline
  各ステップに対してユーザが名前を与える必要がない場合、
  クラス名に基づいて個々のステップに自動的に名前付けを行う。

6.4.2 ステップ属性へのアクセス

- named_steps属性
  - パイプラインに含まれる各ステップの属性を見たい場合使用。
  - ステップ名とEstimatorのディクショナリ

6.4.3 GridSearchCV内のパイプラインの属性へのアクセス

1. LogisticRegressionクラス分類器を用いたグリッドサーチを行う。この際、Pipelineを用いてLogisticRegressionにデータを渡す前に、StandardScalerによるスケール変換を行う。
2. LogisticRegressionの正則化パラメータはCであり、make_pipeline関数を使用しているため、パラメータグリッドでlogisticregression_Cを指定する。

In:
```
from sklearn.linear_model import LogisticRegression
pipe = make_pipeline(StandardScaler(), LogisticRegression())
param_grid = {'logisticregression__C': [0.01, 0.1, 1, 10, 100]}
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, random_state=4)
grid = GridSearchCV(pipe, param_grid, cv=5)
grid.fit(X_train, y_train)
```

6.4.3 GridSearchCV内のパイプラインの属性へのアクセス

3. GridSearchCVで見つけた最良のLogisticRegressionモデルは、grid.best_estimator_に格納されているものを表示。

Out:

```
Best estimator:
Pipeline(memory=None,
     steps=[('standardscaler', StandardScaler(copy=True, with_mean=True,
with_std=True)), ('logisticregression', LogisticRegression(C=0.1, class_weight=None,
dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False))])
```

6.4.3 GridSearchCV内のパイプラインの属性へのアクセス

4. named_steps属性を利用して、logisticregression関して表示。

Out:
```
Logistic regression step:
LogisticRegression(C=0.1, class_weight=None, dual=False, fit_intercept=True,
        intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
        penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
        verbose=0, warm_start=False)
```

5. 個々の入力特徴量に対応する係数ついて表示。

Out:
```
Logistic regression coefficients:
[[-0.389 -0.375 -0.376 -0.396 -0.115  0.017 -0.355 -0.39  -0.058  0.209
  -0.495 -0.004 -0.371 -0.383 -0.045  0.198  0.004 -0.049  0.21   0.224
  -0.547 -0.525 -0.499 -0.515 -0.393 -0.123 -0.388 -0.417 -0.325 -0.139]]
```