2018-09-18_sklearn_api

November 14, 2019

1 2018-09-18 - API de scikit-learn

Présentation de l'API de *scikit-learn* et implémentation d'un prédicteur fait maison. On utilise le jeu du Titanic qu'on peut récupérer sur opendatasoft ou awesome-public-datasets.

[1]:
```
import pandas
df = pandas.read_csv("titanic.csv/titanic.csv")
df.head(n=2)
```

[1]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1

   Parch     Ticket     Fare Cabin Embarked
0      0  A/5 21171   7.2500   NaN        S
1      0   PC 17599  71.2833   C85        C
```

[2]:
```
X, y = df[["Age", "Fare"]], df['Survived']
```

[3]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

[4]:
```
from sklearn.linear_model import LogisticRegression
cls = LogisticRegression()
try:
    cls.fit(X_train, y_train)
except Exception as e:
    print(e)
```

```
Input contains NaN, infinity or a value too large for dtype('float64').
```

[5]:
```
try:
    from sklearn.impute import SimpleImputer as Imputer
except ImportError:
    from sklearn.preprocessing import Imputer
imp = Imputer()
imp.fit(X_train)
```

```
X_train_nomiss = imp.transform(X_train)
```

[6]:
```
cls = LogisticRegression()
cls.fit(X_train_nomiss, y_train)
```

[6]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

[7]:
```
cls.score(imp.transform(X_test), y_test)
```

[7]:
```
0.6502242152466368
```

[8]:
```
from sklearn.pipeline import Pipeline
pipe = Pipeline([("imputer", Imputer()),
                 ("lr", LogisticRegression())])
pipe.fit(X_train, y_train)
```

[8]:
```
Pipeline(memory=None,
         steps=[('imputer',
                 SimpleImputer(add_indicator=False, copy=True, fill_value=None,
                               missing_values=nan, strategy='mean',
                               verbose=0)),
                ('lr',
                 LogisticRegression(C=1.0, class_weight=None, dual=False,
                                    fit_intercept=True, intercept_scaling=1,
                                    l1_ratio=None, max_iter=100,
                                    multi_class='auto', n_jobs=None,
                                    penalty='l2', random_state=None,
                                    solver='lbfgs', tol=0.0001, verbose=0,
                                    warm_start=False))],
         verbose=False)
```

[9]:
```
pipe.score(X_test, y_test)
```

[9]:
```
0.6502242152466368
```

[10]:
```
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(pipe, {"imputer__strategy": ['mean', 'most_frequent'],
                           "lr__max_iter": [5, 10, 50]})
grid.fit(X_train, y_train)
```

```
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
```

```
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
```

```
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\xavierdupre\__home_\github_fork\scikit-
learn\sklearn\linear_model\logistic.py:935: ConvergenceWarning: lbfgs failed to
converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
```

[10]:
```
GridSearchCV(cv=None, error_score=nan,
             estimator=Pipeline(memory=None,
                                steps=[('imputer',
                                        SimpleImputer(add_indicator=False,
                                                      copy=True,
                                                      fill_value=None,
                                                      missing_values=nan,
                                                      strategy='mean',
                                                      verbose=0)),
                                       ('lr',
                                        LogisticRegression(C=1.0,
                                                           class_weight=None,
                                                           dual=False,
                                                           fit_intercept=True,
                                                           intercept_scaling=1,
                                                           l1_ratio=None,
                                                           max_iter=100,
                                                           multi_class='auto',
                                                           n_jobs=None,
                                                           penalty='l2',
                                                           random_state=None,
                                                           solver='lbfgs',
                                                           tol=0.0001,
                                                           verbose=0,
                                                           warm_start=False))],
                                verbose=False),
             iid='deprecated', n_jobs=None,
             param_grid={'imputer__strategy': ['mean', 'most_frequent'],
                         'lr__max_iter': [5, 10, 50]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

[11]:
```
res = pandas.DataFrame(grid.cv_results_)
col = [_ for _ in res.columns if 'param_' in _ or "test_score" in _]
res[col].T
```

[11]:

	0	1	2	3	4	5
param_imputer__strategy	mean	mean	mean	most_frequent	most_frequent	most_frequent
param_lr__max_iter	5	10	50	5	10	50
split0_test_score	0.686567	0.69403	0.656716	0.686567	0.69403	0.656716
split1_test_score	0.619403	0.604478	0.597015	0.61194	0.626866	0.61194
split2_test_score	0.679104	0.679104	0.671642	0.664179	0.671642	0.656716
split3_test_score	0.706767	0.699248	0.684211	0.706767	0.714286	0.684211
split4_test_score	0.676692	0.699248	0.699248	0.676692	0.706767	0.691729
mean_test_score	0.673707	0.675222	0.661766	0.669229	0.682718	0.660263
std_test_score	0.0291387	0.0361333	0.0352828	0.0318525	0.0314484	0.0280138
rank_test_score	3	2	5	4	1	6

[12]:
```
from sklearn.base import BaseEstimator, ClassifierMixin
import numpy

class MeanPredictor(BaseEstimator, ClassifierMixin):
    def __init__(self, alpha=0.5):
        self.alpha = alpha

    def fit(self, X, y):
        self.mean_ = int(self.alpha + numpy.mean(y))

    def predict(self, X):
        return numpy.array(list(self.mean_ for k in range(X.shape[0])))
```

[13]:
```
pipe_mean = Pipeline([('imputer', Imputer()),
                      ('meanpredictor', MeanPredictor())])
pipe_mean.fit(X_train, y_train)
```

[13]:
```
Pipeline(memory=None,
         steps=[('imputer',
                 SimpleImputer(add_indicator=False, copy=True, fill_value=None,
                               missing_values=nan, strategy='mean',
                               verbose=0)),
                ('meanpredictor', MeanPredictor(alpha=0.5))],
         verbose=False)
```

[14]:
```
pipe_mean.score(X_test, y_test)
```

[14]:
```
0.6322869955156951
```

[15]:
```
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(pipe_mean, {"imputer__strategy": ['mean', 'most_frequent'],
                                "meanpredictor__alpha": [0.2, 0.5, 0.8]})
grid.fit(X_train, y_train)
```

[15]:
```
GridSearchCV(cv=None, error_score=nan,
             estimator=Pipeline(memory=None,
                                steps=[('imputer',
                                        SimpleImputer(add_indicator=False,
                                                      copy=True,
                                                      fill_value=None,
                                                      missing_values=nan,
                                                      strategy='mean',
                                                      verbose=0)),
                                       ('meanpredictor',
                                        MeanPredictor(alpha=0.5))],
                                verbose=False),
             iid='deprecated', n_jobs=None,
             param_grid={'imputer__strategy': ['mean', 'most_frequent'],
                         'meanpredictor__alpha': [0.2, 0.5, 0.8]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

[16]:
```
res = pandas.DataFrame(grid.cv_results_)
col = [_ for _ in res.columns if 'param_' in _ or "test_score" in _]
res[col].T
```

[16]:

	0	1	2	3	4	5
param_imputer__strategy	mean	mean	mean	most_frequent	most_frequent	most_frequent
param_meanpredictor__alpha	0.2	0.5	0.8	0.2	0.5	0.8
split0_test_score	0.61194	0.61194	0.38806	0.61194	0.61194	0.38806
split1_test_score	0.61194	0.61194	0.38806	0.61194	0.61194	0.38806
split2_test_score	0.61194	0.61194	0.38806	0.61194	0.61194	0.38806
split3_test_score	0.609023	0.609023	0.390977	0.609023	0.609023	0.390977
split4_test_score	0.609023	0.609023	0.390977	0.609023	0.609023	0.390977
mean_test_score	0.610773	0.610773	0.389227	0.610773	0.610773	0.389227
std_test_score	0.0014294	0.0014294	0.0014294	0.0014294	0.0014294	0.0014294
rank_test_score	1	1	5	1	1	5

[17]:
```python
best = grid.best_estimator_
```

[18]:
```python
import pickle
with open("model.pkl", "wb") as f:
    pickle.dump(best, f)
```

[19]:
```python
with open("model.pkl", "rb") as f:
    model = pickle.load(f)
```

[20]:
```python
model.predict(X_test) == best.predict(X_test)
```

[20]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True])
```

[21]:
```python

```