sklearn_pipeline

February 24, 2019

In [1]:
```
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline, FeatureUnion

X, y = make_classification(n_samples = 100, n_features=100,
                           n_informative=5, n_redundant=2, random_state=101)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=

classifier = LogisticRegression(C=0.1, penalty='l1', random_state=101)
```

In [2]:
```
# define parallel steps
from sklearn.decomposition import PCA, KernelPCA
from sklearn.preprocessing import FunctionTransformer

def identity(x):
    return x

def inverse(x):
    return 1.0 / x

paraller = FeatureUnion(transformer_list=[
    ('pca',PCA()),
    ('kernelpca',KernelPCA()),
    ('inverse', FunctionTransformer(inverse,validate=True )),
    ('original', FunctionTransformer(identity,validate=True ))], n_jobs=-1)
```

In [3]:
```
# define date pipeline
from sklearn.preprocessing import RobustScaler
from sklearn.linear_model import RandomizedLogisticRegression
from sklearn.feature_selection import RFECV
selector = RandomizedLogisticRegression(n_resampling=300, random_state=101, n_jobs=-1)
pipeline = Pipeline(steps=[('paraller_transformation', paraller),
                           ('random_selection', selector),
                          ('logistic_reg', classifier)])
```

```
d:\python\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning: Class Randomiz
  warnings.warn(msg, category=DeprecationWarning)
```

```
In [6]: # find best combination of parameres
        import warnings
        warnings.filterwarnings("ignore")

        from sklearn.model_selection import GridSearchCV
        search_dict = {'logistic_reg__C':[100,10,1,0.1,0.01],
                       'logistic_reg__penalty':['l1','l2']}
        search_func = GridSearchCV(estimator = pipeline, param_grid=search_dict, scoring='accu
                                   iid=False, refit=True, cv=10)
        search_func.fit(X_train, y_train)
```

```
Out[6]: GridSearchCV(cv=10, error_score='raise-deprecating',
               estimator=Pipeline(memory=None,
                   steps=[('paraller_transformation', FeatureUnion(n_jobs=-1,
                     transformer_list=[('pca', PCA(copy=True, iterated_power='auto', n_components=No
                  svd_solver='auto', tol=0.0, whiten=False)), ('kernelpca', KernelPCA(alpha=1.0, coef0=
                        tol=0.0001, verbose=0, warm_start=False))]),
                     fit_params=None, iid=False, n_jobs=-1,
                     param_grid={'logistic_reg__C': [100, 10, 1, 0.1, 0.01], 'logistic_reg__penalty'
                     pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                     scoring='accuracy', verbose=0)
```

```
In [8]: print("BEST ESTIMATOR")
        print(search_func.best_estimator_)
        print("BEST SCORE")
        print(search_func.best_score_)
        print("BEST PARAMS")
        print(search_func.best_params_)
```

```
BEST ESTIMATOR
Pipeline(memory=None,
     steps=[('paraller_transformation', FeatureUnion(n_jobs=-1,
       transformer_list=[('pca', PCA(copy=True, iterated_power='auto', n_components=None, rando
  svd_solver='auto', tol=0.0, whiten=False)), ('kernelpca', KernelPCA(alpha=1.0, coef0=1, copy_
         tol=0.0001, verbose=0, warm_start=False))])
BEST SCORE
0.7369047619047618
BEST PARAMS
{'logistic_reg__C': 1, 'logistic_reg__penalty': 'l2'}
```

```
In [10]: # Generate prediction for test (and terin) data
         from sklearn.metrics import classification_report
         print("Result for test data:")
         print(classification_report(y_test, search_func.predict(X_test)))
```

```
print("Result for train data:")
print(classification_report(y_train, search_func.predict(X_train)))
```

```
Result for test data:
              precision    recall  f1-score   support

           0       0.82      0.75      0.78        12
           1       0.84      0.89      0.86        18

   micro avg       0.83      0.83      0.83        30
   macro avg       0.83      0.82      0.82        30
weighted avg       0.83      0.83      0.83        30

Result for train data:
              precision    recall  f1-score   support

           0       0.87      0.89      0.88        38
           1       0.87      0.84      0.86        32

   micro avg       0.87      0.87      0.87        70
   macro avg       0.87      0.87      0.87        70
weighted avg       0.87      0.87      0.87        70
```

In [ ]: