parameter_tuning_nested

May 29, 2021

1 Cross-validation and hyperparameter tuning

In the previous notebooks, we saw two approaches to tune hyperparameters: via grid-search and randomized-search.

In this notebook, we will show how to combine such hyperparameters search with a cross-validation.

1.1 Our predictive model

Let us reload the dataset as we did previously:

[1]:
```
from sklearn import set_config

set_config(display="diagram")
```

[2]:
```
import pandas as pd

adult_census = pd.read_csv("../datasets/adult-census.csv")
```

We extract the column containing the target.

[3]:
```
target_name = "class"
target = adult_census[target_name]
target
```

[3]:
```
0         <=50K
1         <=50K
2          >50K
3          >50K
4         <=50K
          …
48837     <=50K
48838      >50K
48839     <=50K
48840     <=50K
48841      >50K
Name: class, Length: 48842, dtype: object
```

We drop from our data the target and the `"education-num"` column which duplicates the information from the `"education"` column.

[4]:
```
data = adult_census.drop(columns=[target_name, "education-num"])
data.head()
```

[4]:

	age	workclass	education	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
0	25	Private	11th	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States
1	38	Private	HS-grad	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States
2	28	Local-gov	Assoc-acdm	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States
3	44	Private	Some-college	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States
4	18	?	Some-college	Never-married	?	Own-child	White	Female	0	0	30	United-States

Once the dataset is loaded, we split it into a training and testing sets.

[5]:
```
from sklearn.model_selection import train_test_split

data_train, data_test, target_train, target_test = train_test_split(
    data, target, random_state=42)
```

We will create the same predictive pipeline as seen in the grid-search section.

[6]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OrdinalEncoder
from sklearn.compose import make_column_selector as selector

categorical_columns_selector = selector(dtype_include=object)
categorical_columns = categorical_columns_selector(data)

categorical_preprocessor = OrdinalEncoder(handle_unknown="use_encoded_value",
                                          unknown_value=-1)
preprocessor = ColumnTransformer([
    ('cat-preprocessor', categorical_preprocessor, categorical_columns)],
    remainder='passthrough', sparse_threshold=0)
```

[7]:
```
# for the moment this line is required to import HistGradientBoostingClassifier
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.pipeline import Pipeline

model = Pipeline([
    ("preprocessor", preprocessor),
    ("classifier",
     HistGradientBoostingClassifier(random_state=42, max_leaf_nodes=4))])
model
```

[7]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(remainder='passthrough', sparse_threshold=0,
                                   transformers=[('cat-preprocessor',
OrdinalEncoder(handle_unknown='use_encoded_value',
unknown_value=-1),
                                                  ['workclass', 'education',
                                                   'marital-status',
                                                   'occupation', 'relationship',
                                                   'race', 'sex',
                                                   'native-country'])])),
                ('classifier',
                 HistGradientBoostingClassifier(max_leaf_nodes=4,
                                                random_state=42))])
```

1.2 Include a hyperparameter search within a cross-validation

As mentioned earlier, using a single train-test split during the grid-search does not give any information regarding the different sources of variations: variations in terms of test score or hyperparameters values.

To get reliable information, the hyperparameters search need to be nested within a cross-validation.

Note

To limit the computational cost, we affect cv to a low integer. In practice, the number of fold should be much higher.

[8]:
```
from sklearn.model_selection import cross_validate
from sklearn.model_selection import GridSearchCV

param_grid = {
    'classifier__learning_rate': (0.05, 0.1),
    'classifier__max_leaf_nodes': (30, 40)}
model_grid_search = GridSearchCV(model, param_grid=param_grid,
                                 n_jobs=4, cv=2)

cv_results = cross_validate(
    model_grid_search, data, target, cv=3, return_estimator=True)
```

Running the above cross-validation will give us an estimate of the testing score.

[9]:
```
scores = cv_results["test_score"]
print(f"Accuracy score by cross-validation combined with hyperparameters "
      f"search:\n{scores.mean():.3f} +/- {scores.std():.3f}")
```

```
Accuracy score by cross-validation combined with hyperparameters search:
0.872 +/- 0.002
```

The hyperparameters on each fold are potentially different since we nested the grid-search in the cross-validation. Thus, checking the variation of the hyperparameters across folds should also be analyzed.

[10]:
```
for fold_idx, estimator in enumerate(cv_results["estimator"]):
    print(f"Best parameter found on fold #{fold_idx + 1}")
    print(f"{estimator.best_params_}")
```

```
Best parameter found on fold #1
{'classifier__learning_rate': 0.1, 'classifier__max_leaf_nodes': 40}
Best parameter found on fold #2
{'classifier__learning_rate': 0.1, 'classifier__max_leaf_nodes': 30}
Best parameter found on fold #3
{'classifier__learning_rate': 0.05, 'classifier__max_leaf_nodes': 30}
```

Obtaining models with unstable hyperparameters would be an issue in practice. Indeed, it would become difficult to set them.

In this notebook, we have seen how to combine hyperparameters search with cross-validation.