ensemble_ex_01

July 3, 2024

1 Exercise M6.01

The aim of this notebook is to investigate if we can tune the hyperparameters of a bagging regressor and evaluate the gain obtained.

We will load the California housing dataset and split it into a training and a testing set.

[4]:
```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split

data, target = fetch_california_housing(as_frame=True, return_X_y=True)
target *= 100  # rescale the target in k$
data_train, data_test, target_train, target_test = train_test_split(
    data, target, random_state=0, test_size=0.5
)
```

Note

If you want a deeper overview regarding this dataset, you can refer to the Appendix - Datasets description section at the end of this MOOC.

Create a `BaggingRegressor` and provide a `DecisionTreeRegressor` to its parameter `estimator`. Train the regressor and evaluate its generalization performance on the testing set using the mean absolute error.

[9]:
```
from sklearn.ensemble import BaggingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error

bag = BaggingRegressor(estimator=DecisionTreeRegressor())
bag.fit(data_train, target_train)
target_predicted = bag.predict(data_test)
score = mean_absolute_error(target_predicted, target_test)
score
```

[9]:
```
37.28007749031008
```

Now, create a `RandomizedSearchCV` instance using the previous model and tune the important parameters of the bagging regressor. Find the best parameters and check if you are able to find a set of parameters that improve the default regressor still using the mean absolute error as a metric.

Tip

You can list the bagging regressor's parameters using the get_params method.

[2]:
```
bag.get_params()
```

[2]:
```
{'base_estimator': 'deprecated',
 'bootstrap': True,
 'bootstrap_features': False,
 'estimator__ccp_alpha': 0.0,
 'estimator__criterion': 'squared_error',
 'estimator__max_depth': None,
 'estimator__max_features': None,
 'estimator__max_leaf_nodes': None,
 'estimator__min_impurity_decrease': 0.0,
 'estimator__min_samples_leaf': 1,
 'estimator__min_samples_split': 2,
 'estimator__min_weight_fraction_leaf': 0.0,
 'estimator__random_state': None,
 'estimator__splitter': 'best',
 'estimator': DecisionTreeRegressor(),
 'max_features': 1.0,
 'max_samples': 1.0,
 'n_estimators': 10,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

[11]:
```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

param_grid = {
    'estimator__max_depth': randint(3,10),
    'n_estimators':randint(10, 30),
    'max_features':[0.5, 0.8, 1.0],
    'max_samples':[0.5, 0.8, 1.0],
}
cv_results = RandomizedSearchCV(bag, param_grid, n_iter=20,␣
 ↪scoring="neg_mean_absolute_error")
cv_results.fit(data_train, target_train)
cv_results
```

[11]:
```
RandomizedSearchCV(estimator=BaggingRegressor(estimator=DecisionTreeRegressor())
,
                   n_iter=20,
                   param_distributions={'estimator__max_depth':
```

```
<scipy.stats._distn_infrastructure.rv_discrete_frozen object at 0x7f3ea3565f90>,
                                        'max_features': [0.5, 0.8, 1.0],
                                        'max_samples': [0.5, 0.8, 1.0],
                                        'n_estimators':
<scipy.stats._distn_infrastructure.rv_discrete_frozen object at
0x7f3ea39168d0>},
                   scoring='neg_mean_absolute_error')
```

[14]:
```
target_predicted = cv_results.predict(data_test)
print(
    "Mean absolute error after tuning of the bagging regressor:\n"
    f"{mean_absolute_error(target_test, target_predicted):.2f}"
)
```

```
Mean absolute error after tuning of the bagging regressor:
37.92
```

[18]:
```
randint(3, 10)
```

[18]:
```
<scipy.stats._distn_infrastructure.rv_discrete_frozen at 0x7f3eb0985090>
```

[ ]:
```
```