Chapter 9: Demo GridSearchCV & RandomSearch

In [1]:

```
# Import scikit-learn dataset library
from sklearn import datasets

# Load dataset
iris = datasets.load_iris()
```

In [2]:

```
type(iris)
```

Out[2]:

```
sklearn.utils.Bunch
```

In [3]:

```
# print the label species(setosa, versicolor,virginica)
print(iris.target_names)

# print the names of the four features
print(iris.feature_names)
```

```
['setosa' 'versicolor' 'virginica']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (c
m)']
```

In [4]:

```
# print the iris data (top 5 records)
print(iris.data[0:5])

# print the iris labels (0:setosa, 1:versicolor, 2:virginica)
print(iris.target[:5])
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
[0 0 0 0 0]
```

In [5]:

```
# Creating a DataFrame of given iris dataset.
import pandas as pd
data=pd.DataFrame({
    'sepal length':iris.data[:,0],
    'sepal width':iris.data[:,1],
    'petal length':iris.data[:,2],
    'petal width':iris.data[:,3],
    'species':iris.target
})
data.head()
```

Out[5]:

	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [6]:

```
X=data[['petal length', 'petal width']]
y=data['species']
```

In [7]:

```
from sklearn.model_selection import train_test_split
```

In [8]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                  test_size=0.3,
                                                  random_state = 42)
```

GridSearchCV

In [9]:

```
# Dùng Grid Search
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
```

In [10]:

```
param_grid = {
    'n_estimators': [30, 50, 100, 150, 200, 250, 300],
    'max_features': ['auto', 'sqrt', 'log2'],
    'bootstrap': [True, False],
    'criterion': ["gini", "entropy"]
}
```

In [11]:

```
from datetime import datetime
from datetime import timedelta
```

In [12]:

```
start_time = datetime.now()
```

In [13]:

```
CV_rfc = GridSearchCV(estimator=RandomForestClassifier(),
                      param_grid=param_grid, cv= 5)
```

In [14]:

```
CV_rfc.fit(X_train, y_train)
```

```
c:\program files\python36\lib\site-packages\sklearn\model_selection\_search.py:
814: DeprecationWarning: The default of the `iid` parameter will change from Tr
ue to False in version 0.22 and will be removed in 0.24. This will change numer
ic results when test-set sizes are unequal.
  DeprecationWarning)
```

Out[14]:

```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=RandomForestClassifier(bootstrap=True, class_weight=Non
e,
                                              criterion='gini', max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators='warn', n_jobs=None,
                                              oob_score=False,
                                              random_state=None, verbose=0,
                                              warm_start=False),
             iid='warn', n_jobs=None,
             param_grid={'bootstrap': [True, False],
                         'criterion': ['gini', 'entropy'],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [30, 50, 100, 150, 200, 250, 300]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

In [15]:

```
end_time = datetime.now()
```

In [16]:

```
dt = end_time - start_time
seconds_1 = (dt.days * 24 * 60 * 60 + dt.seconds)
print(seconds_1)
```

```
52
```

In [17]:

```
print(CV_rfc.best_params_)
```

```
{'bootstrap': True, 'criterion': 'gini', 'max_features': 'auto', 'n_estimator
s': 30}
```

In [18]:

```
# Dự đoán trên test dataset
y_pred_1=CV_rfc.predict(X_test)
```

In [19]:

```
from sklearn import metrics
```

In [23]:

```
# Độ chính xác lúc này
print("Accuracy:",metrics.accuracy_score(y_test, y_pred_1))
```

```
Accuracy: 1.0
```

In [24]:

```
# với petal length = 5, petal width = 2 => loại hoa gì?
CV_rfc.predict([[5, 2]])
```

Out[24]:

```
array([2])
```

Sử dụng Random Search

In [25]:

```
# dùng random search
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint as sp_randint
param_dist = {"n_estimators":[30, 50, 100, 150, 200, 250, 300],
              "max_features": ['auto', 'sqrt', 'log2'],
              "bootstrap": [True, False],
              "criterion": ["gini", "entropy"]}
```

In [26]:

```
start_time = datetime.now()
```

In [27]:

```
forest_random = RandomizedSearchCV(estimator=RandomForestClassifier(),
                                   param_distributions=param_dist,
                                   cv=5, random_state=1)
```

In [28]:
```
forest_random.fit(X_train,y_train)
```

```
c:\program files\python36\lib\site-packages\sklearn\model_selection\_search.py:
814: DeprecationWarning: The default of the `iid` parameter will change from Tr
ue to False in version 0.22 and will be removed in 0.24. This will change numer
ic results when test-set sizes are unequal.
  DeprecationWarning)
```

Out[28]:
```
RandomizedSearchCV(cv=5, error_score='raise-deprecating',
                   estimator=RandomForestClassifier(bootstrap=True,
                                                    class_weight=None,
                                                    criterion='gini',
                                                    max_depth=None,
                                                    max_features='auto',
                                                    max_leaf_nodes=None,
                                                    min_impurity_decrease=0.0,
                                                    min_impurity_split=None,
                                                    min_samples_leaf=1,
                                                    min_samples_split=2,
                                                    min_weight_fraction_leaf=0.
0,
                                                    n_estimators='warn',
                                                    n_jobs=None,
                                                    oob_score=False,
                                                    random_state=None,
                                                    verbose=0,
                                                    warm_start=False),
                   iid='warn', n_iter=10, n_jobs=None,
                   param_distributions={'bootstrap': [True, False],
                                        'criterion': ['gini', 'entropy'],
                                        'max_features': ['auto', 'sqrt',
                                                         'log2'],
                                        'n_estimators': [30, 50, 100, 150, 200,
                                                         250, 300]},
                   pre_dispatch='2*n_jobs', random_state=1, refit=True,
                   return_train_score=False, scoring=None, verbose=0)
```

In [29]:
```
end_time = datetime.now()
```

In [30]:
```
dt = end_time - start_time
seconds_2 = (dt.days * 24 * 60 * 60 + dt.seconds)
print(seconds_2)
```

```
7
```

In [36]:

```
forest_random_best = forest_random.best_estimator_
forest_random_best
```

Out[36]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='sqrt', max_leaf_nodes=Non
e,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=150,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [31]:

```
print("Best Model Parameter: ",forest_random.best_params_)
```

```
Best Model Parameter:  {'n_estimators': 150, 'max_features': 'sqrt', 'criterio
n': 'gini', 'bootstrap': True}
```

In [32]:

```
# Dự đoán trên test dataset
y_pred_1=forest_random.predict(X_test)
```

In [33]:

```
# Độ chính xác lúc này
print("Accuracy:",metrics.accuracy_score(y_test, y_pred_1))
```

```
Accuracy: 1.0
```

In [34]:

```
# với petal length = 5, petal width = 2 => loại hoa gì?
forest_random.predict([[5, 2]])
```

Out[34]:

```
array([2])
```