```
# best set of hyper-parameter values
print("Best n_estimators: %d \nBest max_depth: %d \nBest min_samples_leaf:
%d " %  \
            (grid_search.best_estimator_.n_estimators, \
            grid_search.best_estimator_.max_depth, \
            grid_search.best_estimator_.min_samples_leaf))
'Output':
Best n_estimators: 14
Best max_depth: 8
Best min_samples_leaf: 1
```

Randomized Search

As opposed to grid search, not all the provided hyper-parameter values are evaluated, but rather a determined number of hyper-parameter values are sampled from a random uniform distribution. The number of hyper-parameter values that can be evaluated is determined by the **n_iter** attribute of the **RandomizedSearchCV** module.

In this example, we will use the same scenario as in the grid search case.

```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn import datasets

# load dataset
data = datasets.load_boston()

# separate features and target
X = data.data
y = data.target

# construct grid search parameters in a dictionary
parameters = {
    'n_estimators': [2, 4, 6, 8, 10, 12, 14, 16],
    'max_depth': [2, 4, 6, 8],
    'min_samples_leaf': [1,2,3,4,5]
    }
```

```
# create the model
rf_model = RandomForestRegressor()

# run the grid search
randomized_search = RandomizedSearchCV(estimator=rf_model, param_
distributions=parameters, n_iter=10)
# fit the model
randomized_search.fit(X,y)
'Output':
RandomizedSearchCV(cv=None, error_score='raise',
          estimator=RandomForestRegressor(bootstrap=True, criterion='mse',
          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=10, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_
           start=False),
          fit_params=None, iid=True, n_iter=10, n_jobs=1,
          param_distributions={'n_estimators': [2, 4, 6, 8, 10, 12, 14, 16],
          'max_depth': [2, 4, 6, 8], 'min_samples_leaf': [1, 2, 3, 4, 5]},
          pre_dispatch='2*n_jobs', random_state=None, refit=True,
          return_train_score='warn', scoring=None, verbose=0)

# evaluate the model performance
print("Best Accuracy: %.3f%%" %  (randomized_search.best_score_*100.0))
'Output':
Best Accuracy: 57.856%

# best set of hyper-parameter values
print("Best n_estimators: %d \nBest max_depth: %d \nBest min_samples_leaf:
%d " %  \
            (randomized_search.best_estimator_.n_estimators, \
            randomized_search.best_estimator_.max_depth, \
            randomized_search.best_estimator_.min_samples_leaf))
```

```
'Output':
Best n_estimators: 12
Best max_depth: 6
Best min_samples_leaf: 5
```

This chapter further explored using Scikit-learn to incorporate other machine learning techniques such as feature selection and resampling methods to develop a more robust machine learning method. In the next chapter, we will examine our first unsupervised machine learning method, clustering, and its implementation with Scikit-learn.

CHAPTER 25

Clustering

Clustering is an unsupervised machine learning technique for grouping homogeneous data points into partitions called clusters. In the example dataset illustrated in Figure 25-1, suppose we have a set of *n* points and 2 features. A clustering algorithm can be applied to determine the number of distinct subclasses or groups among the data samples.

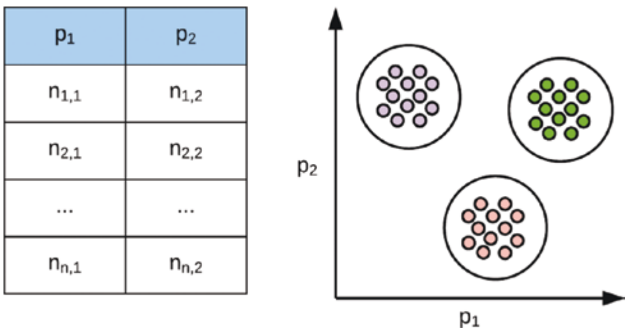

Figure 25-1. *An illustration of clustering in a 2-D space*

Clustering a 2-D dataset as seen in Figure 25-1 is relatively trivial. The real challenge arises when we have to perform clustering in higher-dimensional spaces. The question now is how do we ascertain or find out if a set of points are similar or if a set of points should be in the same group? In this section, we would cover two essential types of clustering algorithms known as k-means clustering and hierarchical clustering.

K-means clustering is used when the number of anticipated distinct classes or sub-groups is known in advance. In hierarchical clustering, the exact number of clusters is not known, and the algorithm is tasked to find the optimal number of heterogeneous sub-groups in the dataset.

E. Bisong, *Building Machine Learning and Deep Learning Models on Google Cloud Platform*,
https://doi.org/10.1007/978-1-4842-4470-8_25