

### 1) Grid Search CV

Exhaustive search over specified parameter values for an estimator. The parameters of the estimator used to apply these methods are optimized by cross-validated grid-search over a parameter grid. Exhaustive search means it tries each and every possible combination and selects the best combination

#### Code:-

```
sklearn.model_selection.GridSearchCV(estimator, param_grid, *, scoring=None, n_jobs=None,
refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', error_score=nan, return
_train_score=False)
```

#### Important Parameters are:-

##### estimator

This is assumed to implement the scikit-learn estimator interface. Either estimator needs to provide a `score` function, or `scoring` must be passed.

##### param\_grid

Dictionary with parameters names (`str`) as keys and lists of parameter settings to try as values, or a list of such dictionaries, in which case the grids spanned by each dictionary in the list are explored. This enables searching over any sequence of parameter settings.

##### scoring

Strategy to evaluate the performance of the cross-validated model on the test set.

##### cv

Determines the cross-validation splitting strategy.

#### Important Attributes are:-

##### best\_estimator

Estimator that was chosen by the search, i.e. estimator which gave highest score(or smallest loss if specified) on the left out data. Not available if `refit=False`.

See `refit` parameter for more information on allowed values.

### **best\_score**

Mean cross-validated score of the best\_estimator

For multi-metric evaluation, this is present only if `refit` is specified. This attribute is not available if `refit` is a function.

### **best\_params**

Parameter setting that gave the best results on the hold out data. For multi-metric evaluation, this is present only if `refit` is specified.

### **Application:-**

```
>>> from sklearn import svm, datasets
>>> from sklearn.model_selection import GridSearchCV
>>> iris = datasets.load_iris()
>>> parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
>>> svc = svm.SVC()
>>> clf = GridSearchCV(svc, parameters)
>>> clf.fit(iris.data, iris.target)
>>> GridSearchCV(estimator=SVC(),
    param_grid={'C': [1, 10], 'kernel': ('linear', 'rbf')})
```

### **Important Methods :-**

`Fit(X, y)`- fit the linear model. `Predict(X)`-

predict using linear model.

`Score(X,y)`-returns the coefficient of determination  $R^2$  of the prediction.

