Grid Search CV

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

```
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')})
```

```
sorted(clf.cv_results_.keys())
```

```
['mean_fit_time',
 'mean_score_time',
 'mean_test_score',
 'param_C',
 'param_kernel',
 'params',
 'rank_test_score',
 'split0_test_score',
 'split1_test_score',
 'split2_test_score',
 'split3_test_score',
 'split4_test_score',
 'std_fit_time',
 'std_score_time',
 'std_test_score']
```

Parameters:

- **Estimator : estimator object -** *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 : dict or list of dictionaries -** *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.*
- **Scoring : str, callable, list, tuple or dict, default=None -** *strategy to evaluate the performance of the cross-validated model on the test set.*
- **n_jobs : int, default=None -** *Number of jobs to run in parallel. None means 1 unless in a joblib.parallel_backend context. -1 means using all processors.*
- **Refit : bool, str, or callable, default=True -** *Refit an estimator using the best found parameters on the whole dataset.*
- **Cv : int, cross-validation generator or an iterable, default=None -** *Determines the cross-validation splitting strategy.*
- **Verbose : int -** *Controls the verbosity: the higher, the more messages.*
- **pre_dispatch : int, or str, default=n_jobs -** *Controls the number of jobs that get dispatched during parallel execution. Reducing this number can be useful to avoid an explosion of memory consumption when more jobs get dispatched than CPUs can process.*
- **error_score : 'raise' or numeric, default=np.nan -** *Value to assign to the score if an error occurs in estimator fitting.*
- **return_train_score : bool, default=False -** *If False, the cv_results_ attribute will not include training scores.*

Attributes:

- **cv_results_ : dict of numpy (masked) ndarrays -** A dict with keys as column headers and values as columns, that can be imported into a pandas DataFrame.
- **best_estimator_ : 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.**
- **best_score_ : float -** Mean cross-validated score of the best_estimator
- **best_params_ : dict -** Parameter setting that gave the best results on the hold out data.
- **best_index_ : int - The index (of the cv_results_ arrays) which corresponds to the best candidate parameter setting.**
- **scorer_ : function or a dict -** Scorer function used on the held out data to choose the best parameters for the model.
- **n_splits_ : int -** The number of cross-validation splits (folds/iterations).
- **refit_time_ : float -** Seconds used for refitting the best model on the whole dataset.
- **multimetric_ : bool -** Whether or not the scorers compute several metrics.

Grid Search CV:

- GridSearchCV is the process of performing hyperparameter tuning in order todetermine the optimal values for a given model. GridSearchCV is a function that comes in Scikit-learn's model selection package.
- This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.
  We pass predefined values for hyperparameters to the GridSearchCV function.
  This is done by defining a dictionary in which we mention a particular hyperparameter along with the values it can take.
  GridSearchCV tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method.
- Hence after using this function, we get accuracy/loss for every combination of hyperparameters and we can choose the one with the best performance.