GridSearchCV API ASSIGNMENT

NAME: Nishant Parekh

ROLL NO: J046

**API:-**

Exhaustive search over specified parameter values for an estimator.

Important members are fit, predict.

GridSearchCV implements a “fit” and a “score” method. It also implements “score\_samples”, “predict”, “predict\_proba”, “decision\_function”, “transform” and “inverse\_transform” if they are implemented in the estimator used.

The parameters of the estimator used to apply these methods are optimized by cross-validated grid-search over a parameter grid.

**CODE:-**

class 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)

**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.

A) If scoring represents a single score, one can use:

i) a single string;

ii) a callable that returns a single value.

B) If scoring represents multiple scores, one can use:

i) a list or tuple of unique strings;

ii) a callable returning a dictionary where the keys are the metric names and the values are the metric scores;

iii) a dictionary with metric names as keys and callables a values.

* **n\_jobs: int, default=None**

Number of jobs to run in parallel.

* **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.

A) >1 : the computation time for each fold and parameter candidate is displayed;

B) >2 : the score is also displayed;

1. >3 : the fold and candidate parameter indexes are also displayed together with the starting time of the computation.

* **pre\_dispatch: int, or str, default=n\_jobs**

Controls thenumber of jobs that get dispatched during parallel execution.

* **error\_score‘raise’ or numeric, default=np.nan**

Value to assign to the score if an error occurs in estimator fitting. If set to ‘raise’, the error is raised.

* **return\_train\_scorebool, 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.