What is `GridSearchCV`?

`GridSearchCV` is a function in scikit-learn (sklearn) that performs hyperparameter tuning for machine learning models using cross-validation. It searches for the best combination of hyperparameters that results in the best performance of the model.

Key arguments of `GridSearchCV`

Here are the key arguments of `GridSearchCV`:

- **`estimator`**: The machine learning model to tune. (e.g., `KNeighborsRegressor()`)
- **`param_grid`**: A dictionary with hyperparameter names as keys and lists of possible values as values. (e.g., `{'n_neighbors': [3, 5, 7], 'weights': ['uniform', 'distance']}`)
- **`cv`**: The number of folds for cross-validation. Can be an integer, a `KFold` object, or a `StratifiedKFold` object. (e.g., `5` for 5-fold cross-validation)
- **`scoring`**: The evaluation metric. Can be a string (e.g., `'neg_mean_squared_error'`, `'r2'`) or a callable function.
- **`n_jobs`**: The number of jobs to run in parallel. If `-1`, all CPUs are used.

Example usage of `GridSearchCV` with `KNeighborsRegressor`

Here's an example:

```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsRegressor
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split

# Load Boston housing dataset
boston = load_boston()
X, y = boston.data, boston.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create a K-Nearest Neighbors Regressor model
knn_model = KNeighborsRegressor()

# Define the hyperparameter grid
param_grid = {
    'n_neighbors': [3, 5, 7],
    'weights': ['uniform', 'distance']
}

# Perform grid search with 5-fold cross-validation
grid_search = GridSearchCV(knn_model, param_grid, cv=5,
scoring='neg_mean_squared_error', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Print the best parameters and the corresponding score
print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)
```

In this example, we:

- Load the Boston housing dataset.
- Split the data into training and testing sets using `train_test_split`.
- Create a `KNeighborsRegressor` model.
- Define a hyperparameter grid with two hyperparameters: `n_neighbors` and `weights`.
- Perform a grid search with 5-fold cross-validation (`cv=5`) using the negative mean squared error (`scoring='neg_mean_squared_error'`) as the evaluation metric.
- Run the grid search in parallel using all available CPUs (`n_jobs=-1`).

Output

```
Best parameters: {'n_neighbors': 5, 'weights': 'uniform'}
Best score: -10.5311
```

In this output, we see the best combination of hyperparameters and the corresponding score.

That's it! You've now used `GridSearchCV` to perform hyperparameter tuning for a `KNeighborsRegressor` model using cross-validation.