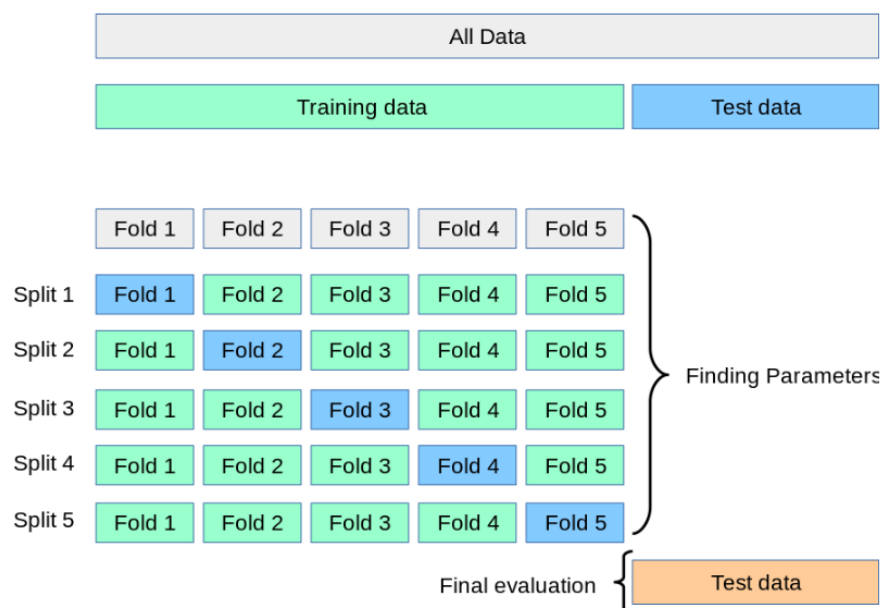**What is GridSearchCV ?**

GridSearchCV tests different hyperparameter values to find the best one for your model.

You should use GridSearchCV instead of KFold and cross_val_score when you're not only looking to evaluate a model but also want to tune hyperparameters to find the best combination for your model.

**How GridSearchCV Works?**

GridSearchCV splits the training data in k equal parts (folds).
Each fold is used as a validation set and the remaining is used as training



- train data using first fold in validation set and get a metric out of it like maybe the accuracy score and then the rest is the training data

Perform cross-validation(eg: with KFold) for each combination of hyper parameters.
Returns the best hyperparameters based on average performance across the cross validation folds
CV - cross validation done by model itself to check accuracy and if accuracy is not good then it tries to perform the iteration again.

```
grid_search_cv.best_params_   # best C, kernel, gamma
grid_search_cv.best_score_    # best cross-validation score
grid_search_cv.best_estimator_  # trained model with best params
```

| n_neighbors | Fold 1 Accuracy | Fold 2 Accuracy | Fold 3 Accuracy | Fold 4 Accuracy | Fold 5 Accuracy | Mean Accuracy | Best Parameter |
|---|---|---|---|---|---|---|---|
| 1 | 0.95 | 0.94 | 0.93 | 0.92 | 0.94 | 0.94 | |
| 3 | 0.96 | 0.95 | 0.95 | 0.96 | 0.95 | 0.95 | Best Parameter |
| 5 | 0.93 | 0.94 | 0.93 | 0.92 | 0.91 | 0.93 | |
| 7 | 0.92 | 0.93 | 0.92 | 0.93 | 0.92 | 0.92 | |
| 9 | 0.91 | 0.90 | 0.89 | 0.90 | 0.88 | 0.90 | |

.

---

## For SVM(hyperparameters)

```
from sklearn.model_selection import GridSearchCV, KFold
```

```
# create cross validator              creates a 5-fold cross-validation
k_fold = KFold(n_splits=5)            Your training data is split into 5 equal parts
                                      The model is trained 5 times(Your training data is split
                                      into 5 equal parts
```

The final score is the average across all 5 runs
This helps estimate how well the model generalizes to unseen data.

```
# which hyperparameters GridSearchCV should try.
parameters = {
    "C": np.arange(10) * 0.1,
    "kernel": ['linear', 'poly', 'rbf', 'sigmoid'],
    "gamma": np.arange(10) * 0.1
}
grid_search_cv = GridSearchCV(estimator=model, param_grid=parameters, cv=k_fold)
grid_search_cv.fit(x_train, y_train)
```

```
"C"  ==> [0.1, 0.2, ..., 0.9]
```
C controls the regularization strength in SVMs
Small C → more regularization (simpler model)
Large C → less regularization (more complex model)

C=0.0 is invalid for SVMs
Usually starts from something like 0.1 or 0.01.

"kernel"  ===>  ['linear', 'poly', 'rbf', 'sigmoid']

These are different SVM kernel functions:

linear: straight line / hyperplane
poly: polynomial decision boundary
rbf: radial basis function (most common)
sigmoid: neural-network-like behavior


"gamma" ===>gamma controls how far the influence of a single training
                        example reaches

Small gamma → smoother decision boundary
Large gamma → more complex boundary
gamma=0.0 is also invalid for svm


GridSearchCV object ===>grid_search_cv = GridSearchCV(
                                        estimator=model,
                                        param_grid=parameters,
                                        cv=k_fold)

This creates a grid search object that:
Tries every possible combination of:

C
kernel
gamma
Uses 5-fold cross-validation for each combination
Evaluates which combination performs best

grid_search_cv.fit(x_train, y_train)

            trains all those models
            Finds the best hyperparameter combination
            Stores the results


grid_search_cv.best_params_   # best C, kernel, gamma
grid_search_cv.best_score_    # best cross-validation score
grid_search_cv.best_estimator_  # trained model with best params