Hyperparameter Tuning P2

ENSF 444

Review Cross validation

Discuss how to search for the best hyperparameters

Splitting a Dataset into Train and Test sets

There are 2 main approaches for splitting datasets

1. The first one is to use 'train-test split', this randomly divide the data into two subsets. One for training the model and one for **evaluating** its performance.
2. 'train-validation-test split' where you further the split the training set into two subsets: one for training the model and one for **tuning** its hyper parameters, then the test is only used for the final evaluation of the model.

```
Searching for optimal hyperparameters is crucial in machine learning for
several reasons:

1. **Improving Model Performance:** Hyperparameters control the behavior and
complexity of machine learning models. By tuning hyperparameters, you can
improve a model's performance and generalization ability, leading to better
predictive accuracy on unseen data.

2. **Avoiding Overfitting and Underfitting:** Selecting appropriate
hyperparameters helps strike a balance between bias and variance in a model.
Tuning hyperparameters can prevent overfitting (when the model learns the
training data too well but performs poorly on new data) or underfitting
(when the model is too simplistic to capture the underlying patterns in the
data).
```

```
3. **Tailoring Models to Specific Datasets:** Different datasets may require different hyperparameters to achieve optimal performance. By searching for the best hyperparameters, you can tailor your model to the characteristics of the dataset at hand, maximizing its effectiveness.

4. **Enhancing Computational Efficiency:** Optimized hyperparameters can lead to more efficient training processes. Tuning hyperparameters can help reduce the time and computational resources required for model training and evaluation by finding the optimal configuration that yields the best results with the fewest resources.

5. **Ensuring Robustness and Stability:** Hyperparameter tuning can make models more robust and stable across different datasets and tasks. By systematically exploring the hyperparameter space, you can identify configurations that result in consistent performance across various scenarios.


Overall, searching for optimal hyperparameters is essential for building high-performing, robust, and efficient machine learning models that generalize well to new data and effectively address the problem at hand....
```

This is what GPT says about looking for optimal hyperparamters

---

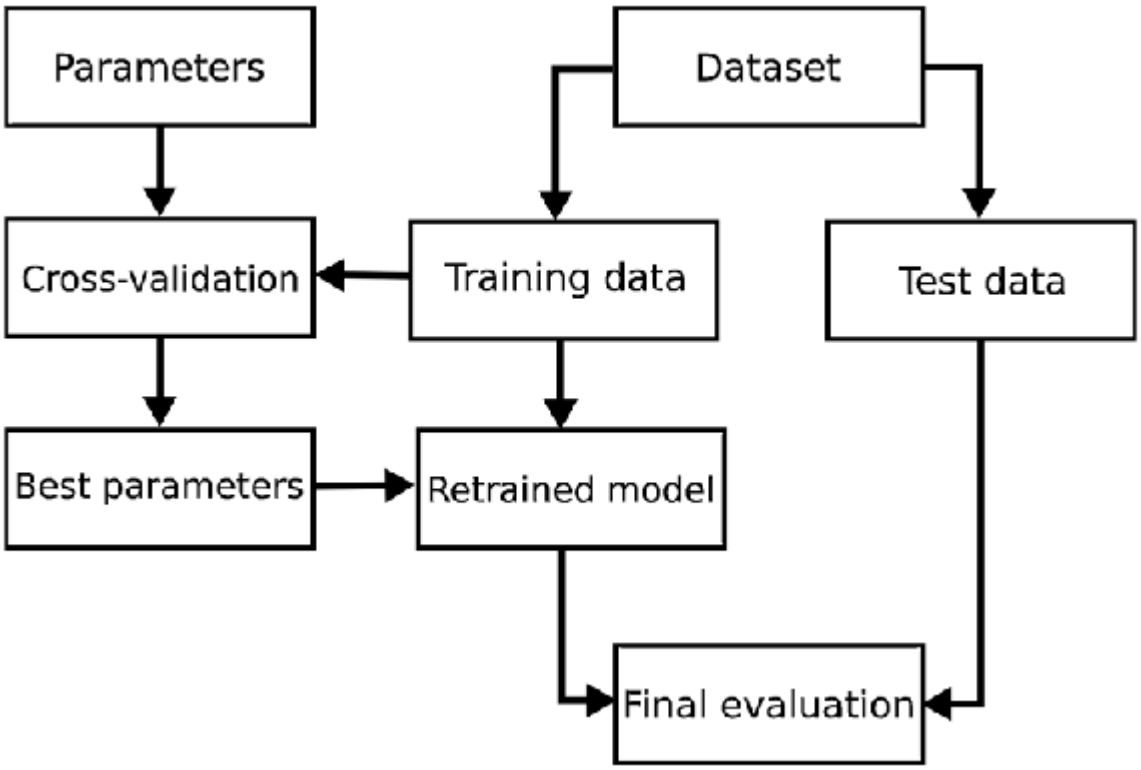
Cross Validated Grid-Search

Grid Search is a function that performs hyperparamter Optimization by training and evaluating a machine learning model using different combinations of hyperparameters.

The 2 advantages of GridSearchCV

1. Applies a grid search to an array of hyperparameters, which means it tries every possible combination of the values you specify for each hyperparameter
2. It cross validates your model using k fold cross validation

This is the flow chart



To use GridSearch

estimator: the machine learning model we want to tune, such as classifier or regressor

param_grid: a dictionary or list of dictionaries with hyperparameter's names as keys and the values to try as values

scoring: a string, a callable, a lis, or dictionary that defines the metric or metrics to use to evaluate the models performance.

n_jobs: an integer that specifies the number of parallel jobs to run -1 means using all processors.

refit: a boolean, a string, or a callable that determines whether to refit the model with the best-found parameters on the whole dataset and make it available as the best*estimator* attribute

Here is a comparison of the grid search

```
import pandas as pd
# convert to DataFrame
results = pd.DataFrame(grid_search.cv_results_).sort_values(by = 'rank_test_score').reset_index(drop = True)
display(results)
```

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
{'max_depth': 5, 'n_estimators': 100}	0.906667	0.946667	0.906667	0.940000	0.940000	0.928000	0.017588	1
{'max_depth': 5, 'n_estimators': 200}	0.906667	0.946667	0.906667	0.940000	0.933333	0.926667	0.016865	2
{'max_depth': 5, 'n_estimators': 150}	0.906667	0.940000	0.906667	0.940000	0.933333	0.925333	0.015434	3
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.

Halving the Grid Search

is a sci kit estimator that performs a grid search over specified parameter values with successive halving

Successive halving is a search strategy that starts evaluating all the candidates with a small amount of resources and iteratively selects the best candidates, using more and more resources.

This can be much faster than a regular grid search, especially when the number of candidates is large.