classifier_cv.py

Summary

This code defines a `ClassifierCV` class for automated classification using gradient boosting backends (XGBoost, CatBoost) with Bayesian optimization and SHAP explanations.

Dependencies

Standard Library

- typing
- multiprocessing

Other

- numpy
- pandas
- pydantic
- sklearn
- xgboost
- catboost
- shap (optional)

Description

The `classifier_cv.py` file implements an automated classification system using gradient-boosting backends as the base classifiers. The core components include:

1. `ClassifierCVConfig`: A Pydantic dataclass that defines configuration options for the classifier, including:

2. `monotone_constraints`: Dictionary specifying monotonicity direction for variables (0 for none, 1 for increasing, -1 for decreasing)
3. `interactions`: List of lists containing permitted feature interactions
4. `n_jobs`: Number of parallel jobs to use when fitting the model
5. `parameters`: Hyperparameter search space definition (OptimizerParams instance)
6. `return_train_score`: Whether to include training scores during optimization
7. Pre-configured settings:
8. `default_classifier`: A standard configuration using all hyperparameters
9. `balanced_classifier`: A configuration focused on regularization parameters
10. `ClassifierCV`: The main class that inherits from `BaseAutoCV`, `ClassifierMixin`, and `ExplainerMixIn`, providing:
11. Automated hyperparameter tuning via Bayesian optimization
12. Pluggable backends: `backend` can be `"xgboost"` (default) or `"catboost"`
13. Model explanation using SHAP values (when available)
14. Feature importance calculation
15. Parallel processing support

The implementation gracefully handles cases where the optional SHAP library isn't available, still allowing the core classification functionality to work while disabling the explanation features.

The class supports custom classification metrics through integration with scikit-learn's scoring system and Pydantic's validation mechanism, ensuring that only valid metrics are used. Cross-validation is employed during the optimization process to prevent overfitting.

Overall, this module provides a robust, automated approach to building classification models with XGBoost, combining the power of Bayesian optimization for parameter tuning with the interpretability offered by SHAP explanations.