regression_cv.py

Summary

This code defines a RegressionCV class for automated regression tree modeling with pluggable gradient-boosting backends using Bayesian optimization.

Dependencies

Standard Library

- typing
- multiprocessing
- functools

Other

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

Description

The `regression_cv.py` file implements a `RegressionCV` class, which is an automated regression tree model based on Bayesian optimization. This class combines several machine learning techniques to create a powerful and flexible regression model.

The implementation includes:

1. `RegressionCVConfig`: A Pydantic dataclass that defines configuration options for the regressor, 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. `quantile_alpha`: Optional parameter for quantile regression (specifies the quantile to predict)
8. Pre-configured settings:
9. `default_regression`: A standard configuration using all hyperparameters
10. `balanced_regression`: A configuration focused on regularization parameters
11. `RegressionCV`: The main class that inherits from `BaseAutoCV`, `RegressorMixin`, and `ExplainerMixIn`, providing:
12. Automated hyperparameter tuning via Bayesian optimization
13. Backend selection through `backend` (`"xgboost"` or `"catboost"`)
14. Support for quantile regression when specified
15. Model explanation using SHAP values (when available)
16. Feature importance calculation
17. Parallel processing support

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

The class supports various regression 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 and ensure model robustness.

Overall, this module provides a comprehensive solution for regression tasks, offering automated model selection, support for advanced regression techniques like quantile regression, and model interpretability through SHAP explanations.