autosklearn-fail analysis

Mohammad Irfan Uddin

cv(folds)	time_left_for_this_task	per_run_time_limit	Accuracy
3	300	100	0.685
3	1200	100	0.685
3	1800	300	0.67
5	300	None	0.6375
5	1200	300	0.67
5	200	None	0.6375
5	200	1200	0.6725
8	300	None	0.64

In addition to the above combinations, I have also experimented with modifying the values of parameters such as initial_configurations_via_metalearning, ensemble_size, ensemble_class, ensemble_nbest, and resampling_strategy. However, these adjustments did not yield any improvement in accuracy.

AutoML Training Accuracy without CV: 0.89
Test Accuracy without CV: 0.61

AutoML Training Accuracy with CV: 0.603
AutoML Test Accuracy with CV: 0.65

From the above, we can see that the accuracy is much higher for the training set without cross-validation compared to the training set and this discrepancy suggests overfitting. However, including cross-validation significantly reduces overfitting.

In the original code without specifying these parameters, AutoSklearn uses its default resampling strategy and the AutoML accuracy is 0.58. However, changing resampling_strategy to "cv" from the default "holdout" increases upto 0.6875. Using cross-validation can help to reduce overfitting and improve the generalization performance of the model. It provides a better evaluation of the model's performance on unseen data, which is crucial for preventing overfitting. AutoSklearn performs hyperparameter tuning during its training process. The cross-validation strategy affects how this tuning is performed, and different strategies may result in different sets of hyperparameters being selected which can improve the performance of the models.

Another interesting finding was that limiting per run time improved accuracy significantly. AutoSklearn divides the total time among different steps, including data preprocessing, feature engineering, and model training. Allocating more time to a specific step doesn't necessarily guarantee better results. It's possible that the first configuration distributed time more effectively across different components, leading to better overall performance.