Analysis of Grid Search Methods for Decision Tree Classifier

Overview

This document summarizes the results of comparing four different grid search strategies for hyperparameter tuning of a `DecisionTreeClassifier`. The goal is to evaluate the trade-off between computational time and model performance (accuracy).

The analysis is based on the `try_it_14.1_required_starter.ipynb` notebook using the Credit Card Fraud dataset.

Methods Compared

Strategies evaluated using `scikit-learn`:

1. **GridSearchCV**: Exhaustive search over specified parameter values.
2. **RandomizedSearchCV**: Randomized search on hyper parameters.
3. **HalvingGridSearchCV**: Searching over specified parameter values with successive halving.
4. **HalvingRandomSearchCV**: Randomized search on hyper parameters with successive halving.

Experimental Setup

- **Model**: `DecisionTreeClassifier`
- **Dataset**: Credit Card Fraud Detection (10k subset)
- **Source**: `miadul/credit-card-fraud-detection-dataset` (Kaggle)
- **Scoring Metric**: Accuracy
- **Cross-Validation**: 5-fold

Data & Preprocessing

- **Target Variable**: `is_fraud`
  Features Used:
  - `amount`
  - `transaction_hour`
  - `merchant_category`
  - `foreign_transaction`
  - `location_mismatch`
  - `device_trust_score`
  - `velocity_last_24h`
  - `cardholder_age`

  Encoders & Scalers:
  - `LabelEncoder`: Used to encode the `merchant_category` column.
  - `StandardScaler`: Applied to all features to scale the data for training.

Parameter Grid

The following parameter grid was explored: * `max_depth`: [None, 2, 5, 10, 15, 20] * `min_samples_split`: [2, 10, 20] * `min_samples_leaf`: [1, 5, 10] * `criterion`: ['gini', 'entropy']

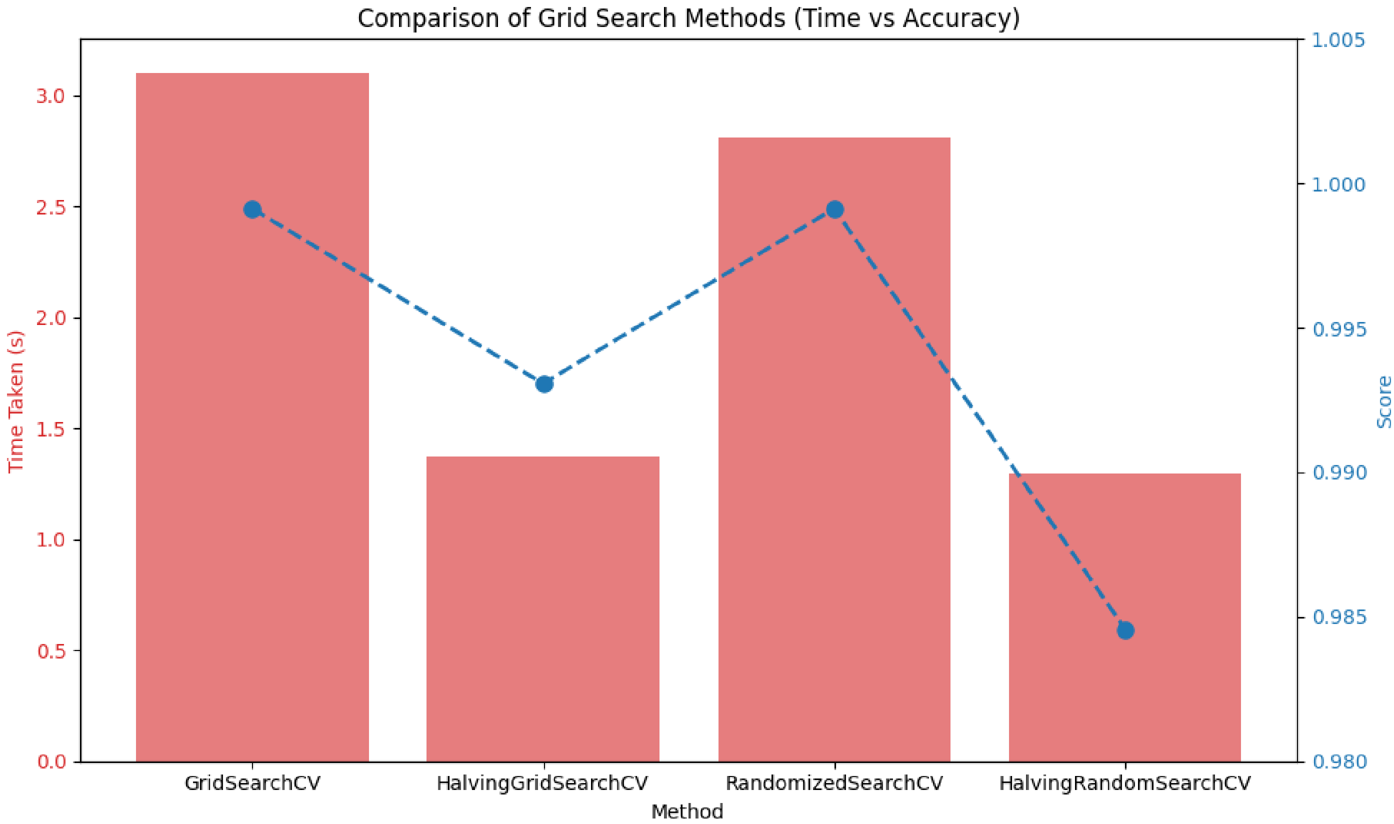
Results

The following table shows the Best Score (Accuracy), the Best Parameters found, and the Time Taken for each method:

Method	Best Score	Time Taken (s)	Best Parameters
GridSearchCV	0.999125	3.10	`{'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 10}`
RandomizedSearchCV	0.999125	2.81	`{'min_samples_split': 20, 'min_samples_leaf': 1, 'max_depth': 10, 'criterion': 'entropy}`
HalvingGridSearchCV	0.993069	1.37	`{'criterion': 'entropy', 'max_depth': 20, 'min_samples_leaf': 10, 'min_samples_split': 10}`
HalvingRandomSearchCV	0.984568	1.29	`{'min_samples_split': 10, 'min_samples_leaf': 10, 'max_depth': 20, 'criterion': 'entropy}`

Visualization

The chart below visualizes the trade-off between execution time (Red bars) and model accuracy (Blue line).



Discussion & Conclusion

Performance vs. Time:

- **GridSearchCV** and **RandomizedSearchCV** achieved the highest accuracy (**99.91%**). However, `GridSearchCV` was the slowest method (~3.1s), while `RandomizedSearchCV` offered a slight speed improvement (~2.8s) with identical performance.
- **Halving Strategies** were significantly faster. `HalvingGridSearchCV` was **2x faster** than standard grid search (~1.37s vs 3.10s) but with a slight drop in accuracy (~99.3%). `HalvingRandomSearchCV` was the fastest (~1.29s) but had the lowest accuracy (~98.4%).

Recommendation:

- For this specific dataset and model, **RandomizedSearchCV** appears to be a strong candidate as it matched the exhaustive search's accuracy while saving some time.
- If computational resources are very limited or the parameter space is huge, **HalvingGridSearchCV** offers a very good compromise, retaining high accuracy (only ~0.6% drop) while cutting runtime by more than half.