Hyperparameters for `LogisticRegression` (Scikit-Learn)

Logistic Regression has several hyperparameters that control its behavior, regularization, and optimization. Below is a **comprehensive list** of key hyperparameters, their purposes, and typical values.

---

1. Core Hyperparameters

Hyperparameter	Description	Common Values
`penalty`	Type of regularization (`'l1'`, `'l2'`, `'elasticnet'`, `'none'`).	`'l2'` (default)
`C`	Inverse of regularization strength (smaller = stronger regularization).	`1.0` (default), `0.01` – `10`
`solver`	Optimization algorithm (see solver compatibility).	`'lbfgs'` (default)
`max_iter`	Maximum number of iterations for solver convergence.	`100` (default), `100` – `1000`
`class_weight`	Weights for classes (handles imbalance). Use `'balanced'` or custom dict.	`None` (default), `'balanced'`

---

2. Regularization-Specific

Hyperparameter	Description	Notes
`l1_ratio`	Mixing parameter for `'elasticnet'` penalty (0 = L2, 1 = L1).	Only if `penalty='elasticnet'`
`dual`	Dual or primal formulation (for `penalty='l2'` and `solver='liblinear'`).	Rarely used (`False` default)

---

3. Solver-Specific Options

Solver (`solver`)	Supported Penalties	Use Cases
`'lbfgs'` (default)	`'l2'`, `'none'`	Small-to-medium datasets, multiclass.
`'liblinear'`	`'l1'`, `'l2'`	Small datasets, binary classification.

Solver (`solver`)	Supported Penalties	Use Cases
`'newton-cg'`	`'l2'`, `'none'`	Medium datasets, multiclass.
`'sag'` / `'saga'`	`'l1'`, `'l2'`, `'elasticnet'`	Large datasets (`'saga'` supports L1).

4. Multiclass Settings

Hyperparameter	Description	Options
`multi_class`	Strategy for multiclass problems (`'auto'`, `'ovr'`, `'multinomial'`).	`'auto'` (default)
`intercept_scaling`	Scales the intercept (useful for `solver='liblinear'` with `penalty='l2'`).	`1.0` (default)

5. Advanced Tuning

Hyperparameter	Description	Example Values
`tol`	Tolerance for stopping criteria (optimization precision).	`1e-4` (default), `1e-3`
`fit_intercept`	Whether to fit an intercept term.	`True` (default)
`verbose`	Log output during training (useful for debugging).	`0` (silent), `1` (verbose)
`warm_start`	Reuse previous solution for incremental training.	`False` (default)

Example: Hyperparameter Tuning with GridSearchCV

python Copy

```python
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import load_iris

# Load data
X, y = load_iris(return_X_y=True)

# Define hyperparameter grid
param_grid = {
    'penalty': ['l1', 'l2'],
    'C': [0.01, 0.1, 1.0, 10],
    'solver': ['liblinear', 'saga'],
```

```
    'max_iter': [100, 200]
}

# Search best hyperparameters
model = LogisticRegression()
grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(X, y)

print("Best hyperparameters:", grid_search.best_params_)
```

Key Takeaways

1. **Regularization**:
   - Use `penalty='l1'` for feature selection (sparse solutions).
   - Use `penalty='l2'` for general-purpose regularization.
2. **Solver Choice**:
   - `'liblinear'` for small datasets, `'saga'` for large datasets or L1.
3. **Class Imbalance**:
   - Set `class_weight='balanced'` if classes are uneven.
4. **Multiclass**:
   - Prefer `multi_class='multinomial'` for better multiclass accuracy.

For production, log hyperparameters in MLflow:

python Copy

```
import mlflow

with mlflow.start_run():
    mlflow.log_params(grid_search.best_params_)
    mlflow.sklearn.log_model(grid_search.best_estimator_, "model")
```