Parameters:

Penalty:
{‘l1’, ‘l2’, ‘elasticnet’, ‘none’}, default=’l2’

Specify the norm of the penalty:

- 'none': no penalty is added;
- 'l2': add a L2 penalty term and it is the default choice;
- 'l1': add a L1 penalty term;
- 'elasticnet': both L1 and L2 penalty terms are added.

Warning

Some penalties may not work with some solvers. See the parameter solver below, to know the compatibility between the penalty and solver.

New in version 0.19: l1 penalty with SAGA solver (allowing ‘multinomial’ + L1)

Dual:
bool, default=False

Dual or primal formulation. Dual formulation is only implemented for l2 penalty with liblinear solver. Prefer dual=False when n_samples > n_features.

Tol:
float, default=1e-4

Tolerance for stopping criteria.

C:
float, default=1.0

Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

fit_intercept:
bool, default=True

Specifies if a constant (a.k.a. bias or intercept) should be added to the decision function.

intercept_scaling:
float, default=1

Useful only when the solver ‘liblinear’ is used and self.fit_intercept is set to True. In this case, x becomes [x, self.intercept_scaling], i.e. a “synthetic” feature with constant value equal to intercept_scaling is appended to the instance vector. The intercept becomes intercept_scaling * synthetic_feature_weight.

Note! the synthetic feature weight is subject to l1/l2 regularization as all other features. To lessen the effect of regularization on synthetic feature weight (and therefore on the intercept) intercept_scaling has to be increased.

class_weight:
dict or ‘balanced’, default=None

Weights associated with classes in the form {class_label: weight}. If not given, all classes are supposed to have weight one.

The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y)).

Note that these weights will be multiplied with sample_weight (passed through the fit method) if sample_weight is specified.

New in version 0.17: *class_weight=’balanced’*

random_state:
int, RandomState instance, default=None

Used when solver == ‘sag’, ‘saga’ or ‘liblinear’ to shuffle the data. See Glossary for details.

Solver:
{‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’}, default=’lbfgs’

Algorithm to use in the optimization problem. Default is ‘lbfgs’. To choose a solver, you might want to consider the following aspects:

- For small datasets, ‘liblinear’ is a good choice, whereas ‘sag’ and ‘saga’ are faster for large ones;
- For multiclass problems, only ‘newton-cg’, ‘sag’, ‘saga’ and ‘lbfgs’ handle multinomial loss;
- ‘liblinear’ is limited to one-versus-rest schemes.

Warning

The choice of the algorithm depends on the penalty chosen: Supported penalties by solver:

- ‘newton-cg’ - [‘l2’, ‘none’]
- ‘lbfgs’ - [‘l2’, ‘none’]
- ‘liblinear’ - [‘l1’, ‘l2’]
- ‘sag’ - [‘l2’, ‘none’]
- ‘saga’ - [‘elasticnet’, ‘l1’, ‘l2’, ‘none’]

Note

‘sag’ and ‘saga’ fast convergence is only guaranteed on features with approximately the same scale. You can preprocess the data with a scaler from sklearn.preprocessing.

See also

Refer to the User Guide for more information regarding LogisticRegression and more specifically the Table summarazing solver/penalty supports. <!– # noqa: E501 –>

New in version 0.17: Stochastic Average Gradient descent solver.

New in version 0.19: SAGA solver.

Changed in version 0.22: The default solver changed from ‘liblinear’ to ‘lbfgs’ in 0.22.

max_iter:
int, default=100

Maximum number of iterations taken for the solvers to converge.

multi_class:
{‘auto’, ‘ovr’, ‘multinomial’}, default=’auto’

If the option chosen is ‘ovr’, then a binary problem is fit for each label. For ‘multinomial’ the loss minimised is the

multinomial loss fit across the entire probability distribution, *even when the data is binary*. ‘multinomial’ is unavailable when solver=’liblinear’. ‘auto’ selects ‘ovr’ if the data is binary, or if solver=’liblinear’, and otherwise selects ‘multinomial’.

New in version 0.18: Stochastic Average Gradient descent solver for ‘multinomial’ case.

Changed in version 0.22: Default changed from ‘ovr’ to ‘auto’ in 0.22.

Verbose:
int, default=0

For the liblinear and lbfgs solvers set verbose to any positive number for verbosity.

warm_start:
bool, default=False

When set to True, reuse the solution of the previous call to fit as initialization, otherwise, just erase the previous solution. Useless for liblinear solver. See the Glossary.

New in version 0.17: *warm_start* to support *lbfgs*, *newton-cg*, *sag*, *saga* solvers.

n_jobs:
int, default=None

Number of CPU cores used when parallelizing over classes if multi_class=’ovr’”. This parameter is ignored when the solver is set to ‘liblinear’ regardless of whether ‘multi_class’ is specified or not. None means 1 unless in a joblib.parallel_backend context. -1 means using all processors. See Glossary for more details.

l1_ratio:
float, default=None

The Elastic-Net mixing parameter, with 0 <= l1_ratio <= 1. Only used if penalty='elasticnet'. Setting l1_ratio=0 is equivalent to using penalty='l2', while setting l1_ratio=1 is equivalent to using penalty='l1'. For 0 < l1_ratio <1, the penalty is a combination of L1 and L2.

Attributes:

classes_:
ndarray of shape (n_classes,)

A list of class labels known to the classifier.

coef_:
ndarray of shape (1, n_features) or (n_classes, n_features)

Coefficient of the features in the decision function.

coef_ is of shape (1, n_features) when the given problem is binary. In particular, when multi_class='multinomial', coef_ corresponds to outcome 1 (True) and -coef_ corresponds to outcome 0 (False).

intercept_:
ndarray of shape (1,) or (n_classes,)

Intercept (a.k.a. bias) added to the decision function.

If fit_intercept is set to False, the intercept is set to zero. intercept_ is of shape (1,) when the given problem is binary. In

particular, when multi_class='multinomial', intercept_ corresponds to outcome 1 (True) and -intercept_ corresponds to outcome 0 (False).

n_features_in_:
int

Number of features seen during fit.

New in version 0.24.

feature_names_in_:
ndarray of shape (n_features_in_,)

Names of features seen during fit. Defined only when X has feature names that are all strings.

New in version 1.0.

n_iter_:
ndarray of shape (n_classes,) or (1,)

Actual number of iterations for all classes. If binary or multinomial, it returns only 1 element. For liblinear solver, only the maximum number of iteration across all classes is given.

Changed in version 0.20: In SciPy <= 1.0.0 the number of lbfgs iterations may exceed max_iter. n_iter_ will now report at most max_iter.