Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the ‘multi\_class’ option is set to ‘ovr’, and uses the cross-entropy loss if the ‘multi\_class’ option is set to ‘multinomial’. (Currently the ‘multinomial’ option is supported only by the ‘lbfgs’, ‘sag’, ‘saga’ and ‘newton-cg’ solvers.)

This class implements regularized logistic regression using the ‘liblinear’ library, ‘newton-cg’, ‘sag’, ‘saga’ and ‘lbfgs’ solvers. **Note that regularization is applied by default**. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied).

The ‘newton-cg’, ‘sag’, and ‘lbfgs’ solvers support only L2 regularization with primal formulation, or no regularization. The ‘liblinear’ solver supports both L1 and L2 regularization, with a dual formulation only for the L2 penalty. The Elastic-Net regularization is only supported by the ‘saga’ solver.

Program:-

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LogisticRegression

from sklearn import datasets

iris = datasets.load\_iris()

X = iris.data[:, :2]

Y = iris.target

logreg = LogisticRegression(C=1e5)

logreg.fit(X, Y)

x\_min, x\_max = X[:, 0].min() - .5, X[:, 0].max() + .5

y\_min, y\_max = X[:, 1].min() - .5, X[:, 1].max() + .5

h = .02 # step size in the mesh

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))

Z = logreg.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.figure(1, figsize=(4, 3))

plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)

plt.scatter(X[:, 0], X[:, 1], c=Y, edgecolors='k', cmap=plt.cm.Paired)

plt.xlabel('Sepal length')

plt.ylabel('Sepal width')

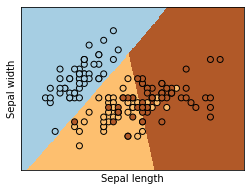
plt.xlim(xx.min(), xx.max())

plt.ylim(yy.min(), yy.max())

plt.xticks(())

plt.yticks(())

plt.show()

Output:-

Program:-

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LogisticRegression

from sklearn import datasets

iris = datasets.load\_iris()

X = iris.data[:, :2]

Y = iris.target

logreg = LogisticRegression(C=1e5)

logreg.fit(X, Y)

x\_min, x\_max = X[:, 0].min() - .5, X[:, 0].max() + .5

y\_min, y\_max = X[:, 1].min() - .5, X[:, 1].max() + .5

h = .02 # step size in the mesh

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))

Z = logreg.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.figure(1, figsize=(4, 3))

plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)

plt.scatter(X[:, 0], X[:, 1], c=Y, edgecolors='k', cmap=plt.cm.Paired)

plt.xlabel('Petal length')

plt.ylabel('Petal width')

plt.xlim(xx.min(), xx.max())

plt.ylim(yy.min(), yy.max())

plt.xticks(())

plt.yticks(())

plt.show()

Output:-

