***class* imbbag.EUSBag (n_estimator=10, estimator=DecisionTreeClassifier(), population_size=50, num_generations = 25, alpha = 0.5, beta = 0.5)**

Evolutionary Under-sampling based Bagging (EUSBag)

EUSBag is a binary-class bagging algorithm that uses evolutionary under-sampling to identify the best subsamples from the majority class and then the balanced training set is used to train an individual classifier.

Source: Sun, B., Chen, H., Wang, J., & Xie, H. (2018). Evolutionary under-sampling based bagging ensemble method for imbalanced data classification. Frontiers of Computer Science, 12, 331-350.

Parameters : n_estimator : ***int (default=10)***

The number of nearest neighbors to search for.

estimator : ***object (default= DecisionTreeClassifier())***

An instance of a base classifier used in the ensemble

population_size: ***int (default=50)***

The size of population used to in the genetic algorithm.

num_generations: ***int (default=25)***

The number of iterations the genetic algorithm goes through to evolve and improve the population of solutions.

alpha : ***real (default= 0.5)***

Coefficients that determine the relative importance of the balance term to ensure a balanced dataset.
By adjusting the coefficients (alpha and beta), the authors can control the trade-off between achieving a balanced dataset and ensuring diversity among the classifiers

beta : ***real (default= 0.5)***

Coefficients that determine the relative importance of the diversity term in the fitness function.
By adjusting the coefficients (alpha and beta), the authors can control the trade-off between achieving a balanced dataset and ensuring diversity among the classifiers.

Examples:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn. model_selection import train_test_split
from ImbBag import EUSBag

dataframe = read_csv('dataset.csv')
data = dataframe.values
X = data[:,:-1]
Y = data[:,:-1]

# split the dataset into training and test sets
X_train ,X_test ,y_train ,y_test = train_test_split (X, y, test_size
=0.2)

# instantiate the imbalance bagging classifier, training, prediction
cls = EUSBag(n_estimator = 10,estimator= DecisionTreeClassifier(),
population_size=50, num_generations=25, alpha=0.5, beta=0.5)
clf.fit(X_train , y_train)
y_pred = clf.predict(X_test)
```

Methods

fit(self, X_train, y_train)	Fit the model.
predict(self, X)	Predict the class label for sample X
predict_proba(self, X)	Estimate the probability of X belonging to each class-labels.

fit(self, X_train,y_train)

Parameters X_train : ***numpy.ndarray of shape (n_samples, n_features)***

The features to train the model.

y_train : ***numpy.ndarray of shape (n_samples,)***

An array-like with the class labels of all samples in X_train.

Returns : self

predict(self, X):

Parameters X : ***numpy.ndarray of shape (n_samples, n_features)***

All the samples we want to predict the label for.

Returns : ***numpy.ndarray***

A 1D array of shape (, n_samples), containing the predicted class labels for all instances in X.

predict_proba(self, X):

Parameters X : ***numpy.ndarray of shape (n_samples, n_features)***

All the samples we want to predict the label for.

Returns : ***numpy.ndarray***

A 2D array of shape (n_samples, n_classes). Where each i-th row contains len(self.target_value) elements, representing the probability that the i-th sample of X belongs to a certain class label.