

```
class imbbag.UnderBag(n_estimator=10, estimator=DecisionTreeClassifier())
```

### Under Bagging classifier.

The Under Bagging algorithm combines the principles of bagging with a focus on under-sampling the majority class to balance the dataset.

**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.

---

### Examples:

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

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 = UnderBag(n_estimator = 50, estimator =
DecisionTreeClassifier())
clf.fit(X_train , y_train)
y_pred = clf.predict(X_test)
```

---

### Methods

|                                    |   |
|------------------------------------|---|
| <b>fit(self, X_train, y_train)</b> | Fit the model.  |
| <b>predict(self, X)</b>            | Predict the class label for sample X                          |
| <b>predict_proba(self, X)</b>      | 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.