***class* imbbag.PTBag(n_elements=10, estimator=*DecisionTreeClassifier()*)**

Probability Threshold Bagging classifier

Probability Threshold Bagging (PTBag) integrates the concept of threshold-moving with bagging, as several studies have identified threshold-moving as an effective algorithm level method for addressing class imbalance issue.
PTBag utilizes bagging to initially achieve well-adjusted posterior probabilities and subsequently applies suitable thresholds based on the performance metric that needs to be optimized. It trains base classifiers using straightforward bootstrap duplicates of the original dataset, thereby maintaining the class distribution and avoids the side effects of the rebalancing methods, such as leading to miscalibrated posterior probability estimates, prior shift, and undesired biases.

Source: Collell, G., Prelec, D., & Patil, K. R. (2018). A simple plug-in bagging ensemble based on threshold-moving for classifying binary and multiclass imbalanced data. Neurocomputing, 275, 330-340.

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 PTBag

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 = PTBag(estimator = DecisionTreeClassifier())
```

```
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

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.