***class* imbbag.NBBag(n_estimator=10, estimator=DecisionTreeClassifier(), n_neighbors=5, psi = None, metric="hvdm", sampling_method="undersampling")**

Neighborhood Balanced Bagging (NBBag)

In the NBBag binary class bagging algorithm, bootstrap sampling is directed towards the more challenging sub-regions of the minority class distribution. This is achieved by adjusting the sampling process to increase the probabilities of drawing the less safe types of minority class examples, while simultaneously decreasing the probabilities of drawing majority class examples.

Source: Błaszczyński, J., & Stefanowski, J. (2015). Neighbourhood sampling in bagging for imbalanced data. Neurocomputing, 150, 529-542.

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.

n_neighbors: ***int (default= 5)***

The number of nearest neighbors used to create kNN classifier .

psi: ***real (default= None)***

The scaling factor is responsible for amplifying the weights of minority class examples in NBBag bootstrap sampling. In the code, the value of *psi* is set according to the original paper: *psi* = 2 for over-sampling and *psi* = 0.5 for under-sampling. However, other values can also be set and used in the current implementation.

metric: ***string (default= "hvdm")***

The metric used for building the kNN classifier can take the values either "hvdm" or "hvdmstd". These values refer to the Heterogeneous Value Difference Metric (HVDM) and the HVDM that uses standard deviation to normalize the distance, respectively.

sampling_method: ***string (default= "undersampling")***

The algorithm applies a sampling method, which can be either undersampling or oversampling, by setting the values "undersampling" and "oversampling" to this parameter, respectively.

Examples:

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

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 = NBBag(n_estimator = 50, estimator =
DecisionTreeClassifier(), sampling_method=”unersampling”)
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.