In the Name of God
Machine Learning (Spring 2020)
Assignment #2
Ensemble Algorithms

Due Date: 5th of Khordad

In this assignment, you are to implement two ensemble algorithms, Bagging, and AdaBoost.M1 for imbalanced data. In the following, these methods are described.

Method I: Adaboost with UnderSampling

The motivation of this method is to keep the high efficiency of under-sampling but reduce the possibility of ignoring potentially useful information contained in the majority class examples. This method randomly generates multiple sample sets $\{N_1, N_2, \ldots, N_T\}$ from the majority class N. The size of each sample set is equal to the size of the minority class, i.e., $|N_i| = |P|$.
In each iteration, the union of pairs N_i and P is used to train the AdaBoost ensemble. The final ensemble is formed by combining all the base learners in all the AdaBoost ensembles. The algorithm is shown in Figure below.

Input: Training data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Minority class examples $P \subseteq D$;
Majority class examples $N \subseteq D$;
Number of subsets T to sample from N;
Number of iterations s_i to train an AdaBoost ensemble H_i.
Process:
1. **for** $i = 1$ to T:
2. Randomly sample a subset N_i from N with $|N_i| = |P|$;
3. Use P and N_i to learn an AdaBoost ensemble H_i, which is with s_i weak classifiers $h_{i,j}$ and corresponding weights $\alpha_{i,j}$:

$$H_i(\boldsymbol{x}) = \texttt{sign}\left(\sum_{j=1}^{s_i} \alpha_{i,j} h_{i,j}(\boldsymbol{x})\right).$$

4. **end**

Output: $H(\boldsymbol{x}) = \texttt{sign}\left(\sum_{i=1}^{T} \sum_{j=1}^{s_i} \alpha_{i,j} h_{i,j}(\boldsymbol{x})\right)$.

This algorithm generates **T balanced sub-problems**. The output of the ith sub-problem is AdaBoost classifier H_i, an ensemble with s_i weak classifiers $\{h_{i,j}\}$. Finally, instead of counting votes from $\{H_i\}_{i=1,\ldots,T}$, we collect all the $h_{i,j}$ $(i = 1, 2, \ldots, T;\ j = 1, 2, \ldots, s_i)$ and form an ensemble classifier out of them.

The output of the above algorithm is a single ensemble, but it looks like an "**ensemble of ensembles**".

Note 1: The number of iterations T, and the number of rounds s_i should be selected from the sets {10, 15} and {11, 31, 51, 101}, respectively. The classification results of all these parameters should be provided in your report.

Method II: Bagging with UnderSampling

For each iteration in Bagging, draw a bootstrap sample from the minority class. Randomly draw the same number of samples, with replacement, from the majority class. Repeat this step for the T number of times. Aggregate the predictions of base learners and make the final prediction.

Note 2: In the Bagging Algorithm, the number of learners or iterations T should be selected from the set {11, 31, 51, 101}, and the results should be reported accordingly.

Use the implemented decision tree as the base learner of your ensemble models. **max_depth** parameter of the base learners in the AdaBoost.M1 algorithm should be tuned **experimentally** so that the decision tree performs a little better than a random classifier. For the Bagging algorithm, use the default parameters of the base learner decision tree.

Split the data set into train and test parts. Use 70% of the data for the training phase and the remaining 30% for the testing phase. Run your codes for 10 individual runs and report the mean and standard deviation of 10 runs for each performance metric.

Since the data set is imbalanced, you should use **Precision**, **Recall**, **F-measure**, and **AUC** measures to evaluate the performance of these algorithms. It is worth mentioning that these measures are one-class measures; in other words, you should compute these metrics just for the minority class. Moreover, **G-mean** should be reported.

$$\boldsymbol{True\ Positive\ Rate}\ (\boldsymbol{Acc_+}) = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}}$$

$$\boldsymbol{True\ Negative\ Rate}\ (\boldsymbol{Acc_-}) = \frac{\boldsymbol{TN}}{\boldsymbol{TN + FP}}$$

$$\boldsymbol{G - mean} = \sqrt{\boldsymbol{Acc_+ \times Acc_-}}$$

Compare the performance of your ensemble methods with the following classifiers: Naïve Bayes, One-nearest-neighbor (OneNN), linear SVM, and kernel SVM (use RBF kernel). For each classifier, use the balanced version of the data by the oversampling and undersampling techniques. Under-sampling deletes majority examples from the dataset so that the number of examples between different classes becomes balanced. Over-sampling is to add minority examples to the dataset to achieve a balance, in which the existing minority examples are replicated, or artificial minority examples are generated.

Note 3: Report the performance of each classifier for **RandomOverSampling**, **SMOTE**, and **RandomUnderSampling**. Feel free to use built-in classifiers, Undersampling, and Oversampling techniques.

Note 4: Pay extra attention to the type of the features in the given data set. Type of features should be considered in your implementation.

Questions:

- Why should we set max_depth parameter in AdaBoost.M1 so that the base classifiers become a little better than random guess?
- What do we mean by stable, unstable, and weak classifier?
- What kind of classifiers should be used in Bagging? How about AdaBoost.M1? Why?

Important Notes:

- Pay extra attention to the due date. It is fixed and **will not be extended**.
- Be advised that all submissions after the deadline **would not be graded**.
- Prepare complete report and answer the questions in you report.
- The name of the uploading file should be your **Lastname_Firstname.**

Good Luck