In the Name of God

Machine Learning

Homework III

Decision Tree Algorithm Ensemble Learning

Assignment Date: **18 Ordibehesht** Submission Deadline: **30 Ordibehesht**

Contents

Precautions

- Any programming language is allowed, preferably python.
- A report is required with the following format:

  Ch.1 Problem Definition
  Ch.2 Solution methodology and implementation details.
  Ch.3 Results and discussion. (answer the homework questions here)
- Your submission will be a zip file containing your code files and your report. Naming format is “your names.zip” for example ”Achraf_Hakimi_Mohamed_Salah.zip”.
- Late submission strategy is: 70 percent score for one day delay and 50 percent for 2 days delay. Submissions after 2 days will not be graded.
- Feel free to ask your questions in the telegram channel.
- Do not copy other works, write your own code.Using other students’ codes or the codes available on the internet will lead to zero grades.
- You must implement all assignment from scratch.(don’t use any libraries for the main subjects)
- Pay extra attention to the due date. It will not be extended,Be advised that submissions after the deadline would not grade.

Thank you. Good Luck.

1 Decision Trees

Your task here is to predict survival of patients with heart failure, using decision tree classifiers. Follow the instructions on data preparation and modeling.

1.1 Data: Heart Failure and Covid19.csv

- Take a look at your dataset on its Kaggle homepage, which can be accessed from this **link**
- Once you opened the link, check out the data card tab. See what information you can find there about this dataset. What does each feature mean?
- Download the dataset.
- We usually recommend you preprocess and study the dataset before modeling. Data visualization, feature encoding, feature normalization and finding the feature correlations are up to you. (you could see what features are of more importance to get a heads up of how your decision tree is probably going to behave)

1.2 Model: Decision Tree

- Search for different decision tree algorithms, your task here is to implement two of them from scratch. (trees like ID3, C4.5, CART, CHAID, MARS etc.)
- It's best if you implement one python class of decision tree then add different fit functions for different tree type.
- In order to evaluate model performance, take the common binary classification metrics into account: Accuracy, Precision, Recall, and the Confusion matrix.
- Finally, you are required to establish a baseline on the current task and compare your results/metrics with that baseline. It's a common thing to do, especially when you are trying to prove that your proposed method is better than the previous works (hopefully in the future when writing your own paper).
- After you implement the two-class DT, you need to extend it to handle the multiclass data with OVO (One Versus One) and OVA (One Versus All) Use the "Covid19" datasets .Then evaluate the performance of both approaches with the mentioned metrics and report the results.
- Now what could that baseline be?
  - You can either use scikit-learn's API for decision trees as your baseline and perform classification using that API.
  - Or go to the code tab of your dataset homepage and pick one of the implementations there as your baseline, no matter what algorithm they used.

1.3 Questions

- Report your evaluation metrics per algorithm.
- Print the classifier's decision rules (or visualize them) on the complete data. Once again, interpret and discuss feature importance.
- Once you know what your baseline is, compare your results/metrics with that baseline. Did you get a better accuracy or not? And why do you think that is? (do this for all three trees you implemented)
- This part is voluntary. One way to prevent decision trees from overfitting, is pruning. Search for what pruning means and examine the effect of pruning on your model performance with different maximum tree heights.

2 Ensemble Learning

2.1 Data: Abalone

For this section you are going to use the Abalone dataset, which can be accessed from this **link**.

2.2 Model: Bagging and Boosting

In this section you will learn how to use Ensemble Learning on imbalanced data. Ensemble methods combine several base models to obtain better predictive performance or improve generalizability. Here we are interested in two main approaches, namely, bagging and boosting to obtain ensemble models.

When dataset is imbalanced, one class (named the minority or positive class) contains a significantly smaller number of samples than the other one (named the majority or negative class). In such cases the model will be biased towards the majority class and thus have a poor performance.

Here you are required to use the algorithm depicted in Fig.1 in order to first balance the dataset and then use ensemble method to measure Average AUROC and AUPR.

Algorithm 1 Synthetic sample generation procedure

```
1: Input: $p$ is the sampling probability distribution for the
   minority class; $N$ is the number of synthetic examples
   from minority class; $k$ is the number of nearest neigh-
   bors.
2: for $n = 1, \ldots, N$ do
3:     Randomly choose a minority class example, $\mathbf{x}$ from
       the sampling probability distribution $p$.
4:     Find $k$NN of $\mathbf{x}$ within the minority class,
       $NN_{k,intra}(\mathbf{x}_i)$.
5:     Randomly select a minority class sample, $\mathbf{x}_{nn}$ from
       $NN_{k,intra}(\mathbf{x}_i)$.
6:     Calculate the difference vector $\boldsymbol{\delta} = \mathbf{x}_{nn} - \mathbf{x}$.
7:     Create a synthetic sample using the following equa-
       tion.
       $\mathbf{x}_{syn} = \mathbf{x} + \lambda\boldsymbol{\delta}$ where $\lambda \in [0, 1]$
8: end for
9: Output: A temporary training dataset combining the
   original data with synthetic data
```

Figure 1

In order to implement the algorithm in Fig.1, follow the instructions:

- Measure dynamic local neighborhood with the equation depicted in Fig.2.

$$NN_d(\mathbf{x}; k) = \{\mathbf{x}_i | \parallel \mathbf{x} - \mathbf{x}_i \parallel \leq r, \mathbf{x}_i \neq \mathbf{x}\}$$

Figure 2: Where r is the distance to k-th nearest sample in the same class as x. Set k to 5.

- Now we define risks for synthetic samples using SMOTE located in the region of the majority class. Therefor use the first Risk formula in Fig.3.
- Then you should measure ration intra/extra nearest neighborhood (N2) which computes the ratio of the sum of distances between individual samples and their nearest neighbors from the same class as formulated in Fig.4 and Fig.5.

[illegible]

$$R1(i;k) = \frac{\sum_{j \in NN_d(\mathbf{x};k)} I(y_i \neq y_j)}{\| NN_d(\mathbf{x};k) \|}$$

Figure 3: Where $\|NN_d(x;k)\|$ is the number of samples in the dynamic local neighborhood.

$$intra\_extra = \frac{\sum_{i=1}^{n} \sum_{j \in NN_{1,intra}(\mathbf{x}_i)} d(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{i=1}^{n} \sum_{j \in NN_{1,extra}(\mathbf{x}_i)} d(\mathbf{x}_i, \mathbf{x}_j)}$$

Figure 4

$$N2 = 1 - \frac{1}{1 + intra\_extra} = \frac{intra\_extra}{1 + intra\_extra}$$

Figure 5: Intra_extra can take any positive real values; hence, $N2$, whose value is within a range of 0 to 1

- The Second risk formula is illustrated in Fig.6 and Fig.7.

$$R2(i;k) = \frac{intra\_extra_k(i)}{1 + intra\_extra_k(i)}$$

Figure 6

$$intra\_extra_k(i) = \frac{\sum_{j \in NN_{k,intra}(\mathbf{x}_i)} d(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{j \in NN_{k,extra}(\mathbf{x}_i)} d(\mathbf{x}_i, \mathbf{x}_j)}$$

Figure 7

- Note that: $R1$ can reflect the degree of overlapping among classes around each sample, and $R2$ can measure how close samples from another class are to another sample compared with samples from the same class.
- In this step we generate new samples from the sampling probability using equations in Fig.8 and Fig.9:

$$p(i) = \frac{S_c(i)(1 - S_r(i))}{\sum_{j \in C_+} S_c(j)(1 - S_r(j))}, \; i \in C_+$$

Figure 8: $C+$ represents the minority (positive) class.

$$S_r(i) = \begin{cases} \frac{R(i) - R_t}{1 - R_t} & \text{if } R(i) > R_t \\ 0 & \text{if } R(i) \leq R_t \end{cases}$$

Figure 9

- In order to update whenever a weak learner is added we use figures 10 to 12.
- After the previous step synthetic samples will be generate and the dataset will be balanced.

$$S_c(i;t) = \frac{\{(T-t-1)N1(i) + (t-1)D'_t(i)\}}{T},\ t = 1, 2, \ldots, T$$

Figure 10: where T is the number of iterations in boosting, t corresponds to each iteration in the boosting procedure.

$$N1(i;k) = \frac{\sum_{j \in NN_k(\mathbf{x}_i)} I(y_i \neq y_j)}{k}$$

Figure 11: $NN_k(x_i)$ is the set of k nearest neighbors of x_i and y_i represents the class label of samples x_i ; I is the indicator function, which returns a value of 1 if its argument is true and 0 otherwise.

$$D'_t(i) = \frac{D_t(i, y_i)}{\max_{j \in C_+} D_t(i, y_i)},\ i \in C_+$$

Figure 12: $D\prime_t(i)$ is the score obtained from the normalized weight distribution of the boosting procedure at iteration t.

- Now perform bagging and boosting on your balanced dataset.(just from scratch) For your base classifier you should select one of the classifiers you implemented in the first section (decision tree). Also note that, for boosting you should set the maximum depth to 1 in order to obtain a weak learner, and for bagging you should find the best depth to increase the diversity among base learners.
- Use Random forest as classifiers for bagging model and compare result with decision tree classifier
- Set the number of base classifiers for each ensemble model to 5. For comparison, five-fold cross validation is applied which will be repeated 5 times.
- Compute accuracy and average AUROC and average AUPR metrics.
- Plot the true positive rate against the false positive rate at various threshold settings of classifiers.

2.3 Bonus

To validate the effect of noise samples in sampling, modify the original dataset to create a noisy dataset by changing the labels of some majority class samples to positive. To increase noise samples in the minority class, first, calculate the proportion of positive samples in the 20 nearest neighbors for each majority class sample, then, select random samples as many as 10% of the minority class size among the majority class samples with a proportion of 0.3 or less. Lastly, change the labels of the selected majority class samples to positive.