1 Introduction

In this assignment, you will make practice on how data mining tasks (e.g., classification, regression, and etc.) can be handled by ensemble, and lazy learning algorithms.

Ensemble learning is a framework that is used by data scientist to produce a more generalized prediction on a given test data. It relies on a number of base learners then combines them for a more robustness over a single base learner. There are sorts of application types with ensemble learning. These can be outlined as follows:

- Different learning algorithms on same problem,
- Same algorithms with different parameters on same problem,
- Different training sets (bagging, pasting, and boosting),
- Different representations of the same input.

k-NN, however, is a lazy learning classification algorithm (also referred as instance-based learning algorithm). Contrary to eager learning approaches, k-NN like lazy learners do not train data to fit a model; but instead, they produce local approximation according to the given test data. Therefore, they take more time on testing but less time on training. The main advantage of k-NN is its adaptability on dynamic problems where the distribution can change over time.

The side goal of this assignment is to make you gained an experience on how a model can be evaluated on different types of approaches. These are *holdout* and *cross-validation* evaluation approaches. Actually you have driven *holdout* approach in the first two assignments. This approach relies on a simple strategy which splits the dataset with a given percentage (commonly adopted as 80% for training, 20% for testing). This approach is preferred mostly when the class distribution is balanced. Cross-validation, however, partitions data into k disjoint subsets (called as k fold cross-validation) then learning approach is trained on *k-1* partitions and tested on remaining partition. It is called *leave-one-out cross validation* when number of k is equal to the number of object is dataset (for a demonstration refer to Fig. 1).

2 Ensemble Learning

In this part, you are expected to apply bagging and boosting classifiers with some base classifiers that fit corresponding subset of data.

2.1 Different training sets

Here, you are to employ bagging &pasting and boosting algorithms for solving classification task. To do that follow the instruction in the following two subsections:

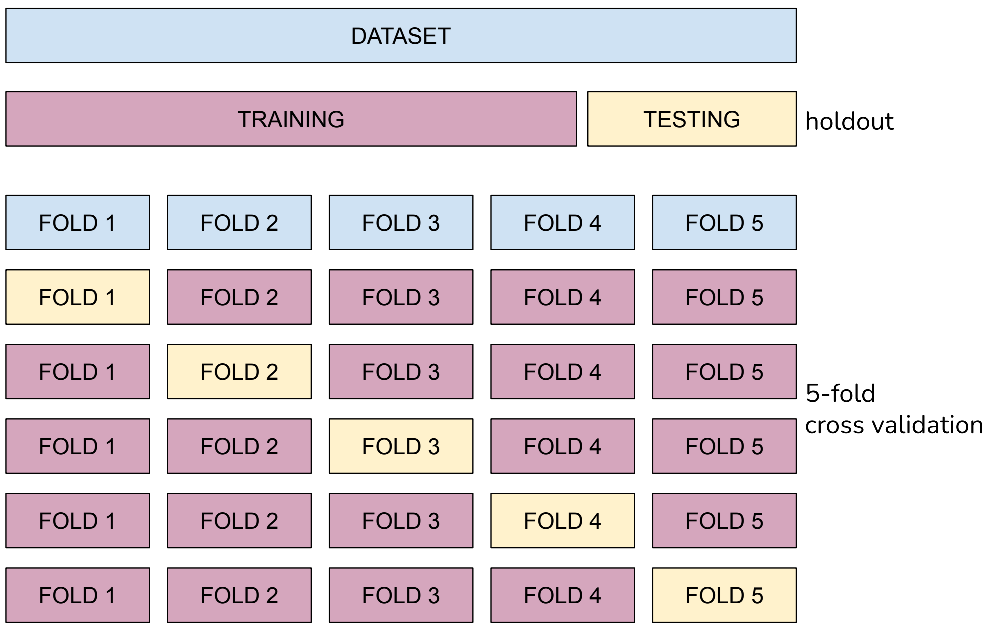

Figure 1: Holdout and cross-validation evaluation strategies.

2.1.1 Bagging & Pasting

1. Load *digit* dataset (D).
2. Split D such that randomly selected 70% tuples are used for training while 30% tuples are used for testing.
3. Create an instance of multi-layer perceptron network (four hidden layers with 16, 8, 4, and 2 neurons in order) with 1,000 or more iterations to handle one subset of D. 1 / 8 of D should be given to each base classifier.
4. Apply bagging classifier (with or without replacement) with eight base classifiers created at the previous step.
5. Calculate number of correctly classified test instance for each base classifier and finally for bagging classifier.
6. Print your findings in a format that exactly matches with the one below:

```
289 out of 540 instances are correctly classified by learner #1
291 out of 540 instances are correctly classified by learner #2
46 out of 540 instances are correctly classified by learner #3
124 out of 540 instances are correctly classified by learner #4
141 out of 540 instances are correctly classified by learner #5
260 out of 540 instances are correctly classified by learner #6
145 out of 540 instances are correctly classified by learner #7
158 out of 540 instances are correctly classified by learner #8
----------------------------------------------------------------
457 out of 540 instances are correctly classified by bagging
```

2.1.2 Boosting

1. Load *moon* dataset D with a tuple size greater than 100 (feel free to give any arbitrary number for every class).
2. Give Gaussian noise to D with a deviation value of 0.2.
3. Split D such that randomly selected 70% tuples are used for training while 30% tuples are used for testing.
4. Create an instance of logistic regression algorithm with SGD solver (with 'log' loss function).
5. Apply AdaBoost classifier with four base classifiers created at the previous step.
6. Through a 1×4-axis figure, visualize testing samples as well as a decision boundary that are learned by each base algorithm. An example outcome to this step is given in Figure 2. Once you output this figure save it as 'BaseLearnerVisualization.pdf'; then **close**[2] figure window.

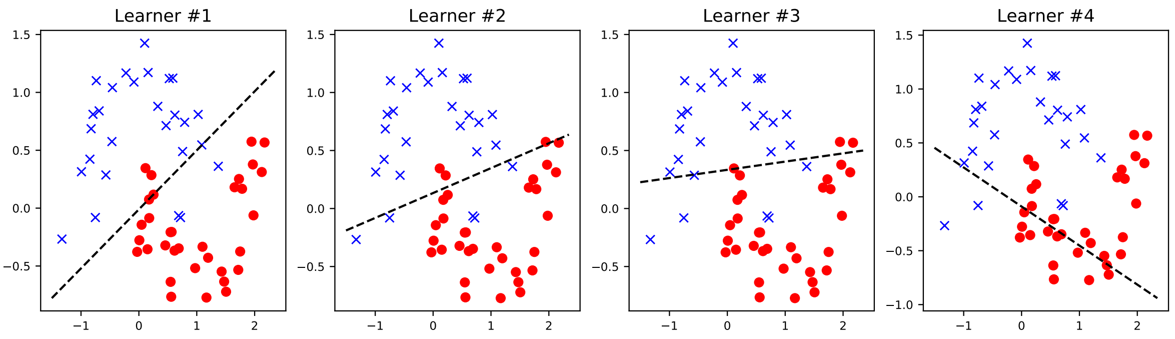

Figure 2: Decision boundary on binary-class classification problem.

2.2 Different learning algorithm

1. Load *breast-cancer* dataset D .
2. Create instances of three classification algorithms (feel free to choose estimator algorithms).
3. Apply Ensemble learning framework that produce final estimation with respect to the majority of votes returned by base classifiers created at the previous step. To do that, 5-fold cross validation strategy should be used.
4. Calculate accuracy of every base classifier and ensemble classifier.
5. Print your findings in a format that exactly matches with the one below:

[2]**never forget to close your window**; otherwise your work will be penalized!

```
Accuracy obtained by learner #1 is: 0.8576
Accuracy obtained by learner #2 is: 0.7981
Accuracy obtained by learner #3 is: 0.8839
------------------------------------------------
Accuracy obtained by ensemble learner is: 0.9086
```

2.3 Different parameter setting

1. Load *breast-cancer* dataset (D).
2. Split D such that randomly selected 70% tuples are used for training while 30% tuples are used for testing.
3. Create 10 instances of multi-layer perceptron network with different hidden layer and neuron size.
4. For hth network ($1 \leq h \leq 10$) the number of neurons in hidden layers is $2^h, 2^{h-1}, \ldots, 2^1$ from the first to the last hidden layers.
5. Apply Ensemble learning framework that produce final estimation with respect to the majority of votes returned by base classifiers created at the previous step.
6. Print your findings in a format that exactly matches with the one below:

```
Parameter setting: l#1 Accuracy: 0.543859649122807
Parameter setting: l#2 Accuracy: 0.543859649122807
Parameter setting: l#3 Accuracy: 0.543859649122807
Parameter setting: l#4 Accuracy: 0.543859649122807
Parameter setting: l#5 Accuracy: 0.543859649122807
Parameter setting: l#6 Accuracy: 0.543859649122807
Parameter setting: l#7 Accuracy: 0.9064327485380117
Parameter setting: l#8 Accuracy: 0.543859649122807
Parameter setting: l#9 Accuracy: 0.543859649122807
Parameter setting: l#10 Accuracy: 0.9064327485380117
----------------------------------------------------
Ensemble Learning Accuracy: 0.543859649122807
```

3 k-Nearest Neighbors Classifier

In this part, you are expected to apply an instance-based classifiers (i.e., kNN algorithm) for some particular classification problems. To do that, follow the procedure below:

1. Load *moon* dataset D with a tuple size greater than 100 (feel free to give any arbitrary number for every class).
2. Give Gaussian noise to D with a deviation value of 0.3.
3. Split D such that only four tuples are used for testing while remaining tuples are used for training.
4. Apply k-NN classifier for each testing tuples (with k = 5 setting).

5. Through a 1×4-axis figure. For each axis, visualize training samples, corresponding testing sample (indicated with '+' marker) as well the nearest k neighbors of that sample (indicated with green border color). The title of each axis should state predicted class.
6. An example outcome to this step is given in Figure 3. Once you output this figure save it as 'kNN.pdf'; then **close**[3] figure window.

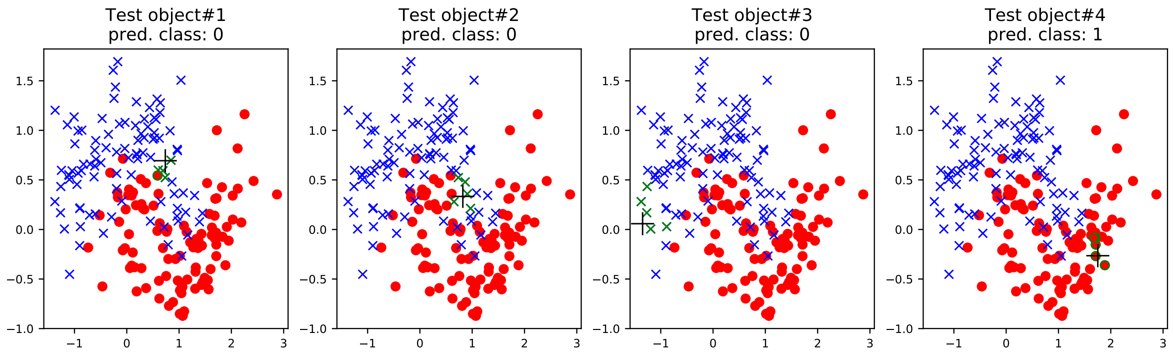

Figure 3: kNN demonstration on 2-D search space.

Notes

- Your source code should be designed as **easy-to-follow**. **Place comment** in it as much as possible. **Separate each task** through apparent patterns.
- Use LaTeX to prepare your reports. Include the observation tables here to your report. Once again, filled and signed declaration form should be first page of your report. **Reports must not exceed 5 pages in total**.
- **Do not miss** the deadline.
- **Save your work** until the end of this semester.
- The assignment must be **original**, **individual work**. **Duplicate or very similar assignments are both going to be considered as cheating.**
- You can ask your questions via **Piazza** (https://piazza.com/mu.edu.tr/fall2020/ceng3521) and you are supposed to be aware of everything discussed in Piazza.
- You will submit your work on CENG3521 course page at https://dys.mu.edu.tr with the file hierarchy as below[4]:

→ <student id>.zip
  → Assignment3.py
  → Report3.pdf

[3]**never forget to close your window**; otherwise your work will be penalized!
[4]do not place any file into a directory. Just compress all the files together.