CS 412: Fall'22
Introduction To Data Mining

Assignment 4

(Due Friday, December 02, 11:59 pm)

- The homework is due at 11:59 pm on the due date. We will be using Gradescope for the homework assignments. You should join Gradescope using the entry code shared on Aug 3. Please do NOT email a copy of your solution. Contact the TAs if you are having technical difficulties in submitting the assignment. We will NOT accept late submissions!
- Please use Slack or Canvas first if you have questions about the homework. You can also come to our (zoom) office hours and/or send us e-mails. If you are sending us emails with questions on the homework, please start subject with "CS 412 Fall'22: " and send the email to *all of us* (Arindam, Mukesh, Chandni, Mayank, Hang, and Shiliang) for faster response.
- Please write your code entirely by yourself.

  Programming Assignment Instructions

  - All programming needs to be in Python 3.
  - The homework will be graded using Gradescope. You will be able to submit your code as many times as you want.
  - Two python files named `homework4_q1.py`, `homework4_q2.py` containing starter code are available on Canvas
  - For submitting on Gradescope, you need to upload both the files `homework4_q1.py` and `homework4_q2.py`.

1. (50 points) The assignment will focus on developing code for Bayes Theorem applied to estimating the type of candy bag we have based on candies drawn from the bag, following the example discussed in class.

   Suppose there are five types of bags of candies:

   - π_1 fraction are h_1: p_1 fraction "cherry" candies, $(1 - p_1)$ fraction "lime" candies
   - π_2 fraction are h_2: p_2 fraction "cherry" candies, $(1 - p_2)$ fraction "lime" candies
   - π_3 fraction are h_3: p_3 fraction "cherry" candies, $(1 - p_3)$ fraction "lime" candies
   - π_4 fraction are h_4: p_4 fraction "cherry" candies, $(1 - p_4)$ fraction "lime" candies
   - π_5 fraction are h_5: p_5 fraction "cherry" candies, $(1 - p_5)$ fraction "lime" candies

   For the specific example discussed in class:

$$\begin{aligned} \pi_1 = 0.1\ , \quad \pi_2 = 0.2\ , \quad \pi_3 = 0.4\ , \quad \pi_4 = 0.2\ , \quad \pi_5 = 0.1\ , \\ p_1 = 1\ , \quad p_2 = 0.75\ , \quad p_3 = 0.5\ , \quad p_4 = 0.25\ , \quad p_5 = 0\ . \end{aligned}$$

   A bag is given to us, but we do not know which type of bag $h \in \{h_1, h_2, h_3, h_4, h_5\}$ it is. We will be drawing a sequence of candies $c_1, c_2, \ldots, c_i \in \{\text{"cherry"}, \text{"lime"}\}$ from the given bag and, using Bayes rule, maintain posterior probabilities of the type of bag conditioned on the candies which have been drawn, i.e.,

$$\begin{aligned} &\text{After drawing } c_1 : && p(\pi_h|c_1)\ , \quad h = 1, \ldots, 5 \\ &\text{After drawing } c_1, c_2 : && p(\pi_h|c_1, c_2)\ , \quad h = 1, \ldots, 5 \\ &\ldots && \ldots \end{aligned}$$

   We will make the following assumptions regarding the setup:

   - The probabilities $p_h, h = 1, \ldots, k$ of drawing "cherry" candies from the bag do not change as candies are being drawn the bag. As a result, the probabilities $(1 - p_h), h = 1, \ldots, k$ of drawing "lime" candies from the bag also do not change as candies are being drawn the bag.
   - The joint probability of drawing different candies from a bag are conditionally independent given the bag, i.e.,

$$\begin{aligned} p(c_1, c_2|\pi_h) &= p(c_1|\pi_h)p(c_2|\pi_h)\ , \\ p(c_1, c_2, c_3|\pi_h) &= p(c_1|\pi_h)p(c_2|\pi_h)p(c_3|\pi_h)\ , \end{aligned}$$

     and so on.

   You will have to develop code for the following function:

   `my_Bayes_candy`(π, p, $c_{1:10}$), which uses prior probabilities π, conditional probabilities p, and sequence of 10 candy draws $c_{1:10}$ to compute posterior probabilities of each type of bag.

   The function will have the following **input**:

   - Prior probability of each type of bag: $\pi = [\pi_1, \pi_2, \pi_3, \pi_4, \pi_5]$

- Conditional probability of cherry candies in each type of bag: $p = [p_1, p_2, p_3, p_4, p_5]$
- Sequence of 10 candies drawn from the bag under consideration: $c_{1:10} = [c_1, c_2, \ldots, c_{10}]$

where $c_i \in 0, 1$ with "0" denoting cherry and "1" denoting lime candy. For the specific example discussed in class:

$$c = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1] \ .$$

The function will have the following **output**:

(a) A list of posterior probabilities of each type of bag $\pi_h, h = 1, \ldots, 5$ after every subsequence of candy draws, i.e., $c_{1:t} = [c_1, \ldots, c_t], t \in \{1, \ldots, 10\}$.

In particular, the output will be the following posterior probabilities:

$$\begin{array}{ll} \text{After drawing } c_1 : & p(\pi_h|c_1) \ , \quad h = 1, \ldots, 5 \\ \text{After drawing } c_1, c_2 : & p(\pi_h|c_1, c_2) \ , \quad h = 1, \ldots, 5 \\ \ldots & \ldots \\ \text{After drawing } c_1, \ldots, c_{10} : & p(\pi_h|c_1, \ldots, c_{10}) \ , \quad h = 1, \ldots, 5 \end{array}$$

The function `my_Bayes_candy` will return a two-dimensional list of size 10x5 containing the aforementioned posterior probabilities. Specifically, the returned list must contain probabilities in the format:

$$\begin{array}{c} [[p(\pi_1|c_1), \ p(\pi_2|c_1), \ ..., \ p(\pi_5|c_1)], \\ [p(\pi_1|c_1, c_2), \ ..., \ p(\pi_5|c_1, c_2)], \\ ..., \\ [p(\pi_1|c_1, .., c_{10}), \ ..., \ p(\pi_5|c_1, .., c_{10})]] \end{array}$$

Note that your code should work for any given $\pi, p, c_{1:10}$ and we will always consider 5 types of bags and 2 types of candies in each bag. The specific example discussed in class can serve as a test case.

2. (50 points) The assignment will focus on developing your own code for: `random train-test split validation`.

   Your code will be evaluated using five standard classification models applied to a multi-class classification dataset.

   Dataset: We will be using the following dataset for the assignment.

   `Digits`: The `Digits` dataset comes prepackaged with `scikit-learn` (sklearn.datasets.load_digits). The dataset has 1797 points, 64 features, and 10 classes corresponding to ten numbers $0, 1, \ldots, 9$. The dataset was (likely) created from the following dataset:
   `https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits`

   Classification Methods. We will consider five classification methods from `scikit-learn`:

- Linear support vector classifier: `LinearSVC`,
- Support vector classifier: `SVC`,
- Logistic Regression: `LogisticRegression`,
- Random Forest Classifier: `RandomForestClassifier`, and
- Gradient Boosting Classifier: `XGBClassifier`.

Use the following parameters for these methods:

- `LinearSVC: max_iter=2000`
- `SVC: gamma='scale', C=10`
- `LogisticRegression: penalty='l2', solver='lbfgs', multi_class='multinomial'`
- `RandomForestClassifier: max_depth=20, random_state=0, n_estimators=500`
- `XGBClassifier: max_depth=5`

Develop code for `my_train_test(method,`X`,`$\mathbf{y}$`,`π`,`k`)`, which performs random splits on the data $(X, \mathbf{y})$ so that $\pi \in [0, 1]$ fraction of the data is used for training using `method`, rest is used for testing, and the process is repeated k times, after which the code returns the error rate for each such train-test split. Your `my_train_test` will be tested with $\pi = 0.75$ and $k = 10$ on the five methods: `LinearSVC`, `SVC`, `LogisticRegression`, `RandomForestClassifier`, and `XGBClassifier` applied to the `Digits` dataset.

You will have to develop code for the following function:

`my_train_test(method,`X`,`$\mathbf{y}$`,`π`,`k`)`, which does random train-test split based evaluation of `method` with π fraction used for training for each split.

The function will have the following **input**:

(1) `method`, which specifies the (class) name of one of the five classification methods under consideration,

(2) X,$\mathbf{y}$ which is data for the classification problem,

(3) π, the fraction of data chosen randomly to be used for training,

(4) k, the number of times the train-test split will be repeated.

The function will have the following **output**:

(a) A list of the test set error rates for each of the k splits.

Error rate should be calculated as $\frac{\texttt{\# of wrong predictions}}{\texttt{\# of total predictions}}$. The (auto)grader will compare the mean and standard deviation of your list with our solution; it must be within three standard deviations.