CS 412: Fall'21
Introduction To Data Mining

Assignment 4

(Due Friday, Dec 3, 11:59 pm)

General Instructions

- Feel free to talk to other students of the class while doing the homework. We are more concerned that you learn how to solve the problems than that you demonstrate that you solved it entirely on your own. You should, however, write down your solution yourself. Please try to keep the solution brief and clear.
- The assignment is due at 11:59 pm on the due date. We will be using Gradescope for collecting the homework assignments. You should have been added to the Gradescope page for our class – if not, please email us. Contact the TAs if you are having technical difficulties in submitting the assignment. We will NOT accept late submissions!
- Please use Slack or Canvas if you have questions about the homework. You can also send us e-mails, and/or come to our office (zoom) hours. If you are sending us emails with questions on the homework, please start subject with "CS 412 Fall'21: " and send the email to *all of us* (Arindam, Yikun, Dongqi, Zhe, and Hang) for faster response.

Programming Assignment Instructions

- All programming needs to be in Python 3.
- A file containing starter code is on Canvas.
- You *cannot* use external libraries for k-fold cross-validation. You can use `scikit-learn` classifiers. You can use numpy.
- The homework will be graded using Gradescope. You will be able to submit your code as many times as you want.
- For this assignment, you will be required to test two different splitting strategies for classification. Each will contribute to half of the possible points on this problem.

Assignment. The assignment will focus on developing your own code for:

- k-fold cross-validation, and
- 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`

1. (50 points) Develop code for
   (a) (30 points) `get_splits`(n, k), which returns k lists of indices corresponding to folds, and
   (b) (20 points) `my_cross_val`(`method`,X,$\mathbf{y}$,k), which performs k-fold cross-validation on $(X, \mathbf{y})$ using `method`, and returns the error rate in each fold.

   Use `my_cross_val` to return the error rates in each fold for the five methods: `LinearSVC`, `SVC`, `LogisticRegression`, `RandomForestClassifier`, and `XGBClassifier` applied to the `Digits` dataset.

   You will have to develop **code** for the following two functions:

(a) `get_splits`(n, k), which returns k (almost) equal sized lists of disjoint indices from the set of all indices $\{0, \ldots, n-1\}$. These k list of indices correspond to the k folds over which cross-validation will be done. The function will have the following **output**: a list containing k lists, arrays, or sets. Each of these sublists should be disjoint, each of size roughly $\frac{n}{k}$, and contain elements from 0 to $n-1$. The union of all the k sublists should include all elements in $\{0, \ldots, n-1\}$. For example, `get_splits`(4, 2) might return [[0,2], [1,3]]; `get_splits`(7, 2) might return [[0,2,4,6], [1,3,5]]; `get_splits`(11, 3) might return [[0,3,6,9], [1,4,7,10], [2,5,8]]. You can use this method in `my_cross_val` to get your splits. You should make sure that the splits are randomized.

(b) `my_cross_val`(`method`,X,$\mathbf{y}$,k), which runs k-fold cross-validation for `method` on the dataset $(X, \mathbf{y})$. The **input parameters** are: (1) `method`, which specifies the (class) name of one of the five classification methods under consideration, (2) X,$\mathbf{y}$ which is the data for the classification problem, and (3) k, the number of folds for cross-validation. The function will have the following **output**: (1) the test set error rates for each of the k folds. The error should be measured as $\frac{\texttt{\# of wrong predictions}}{\texttt{\# of total predictions}}$. The mean and standard deviation of your errors should fall within three standard deviations of our solution in Gradescope.

2. (50 points) 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:

   (a) `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**: (1) A list of the test set error rates for each of the k splits. Error should be calculated as in 1(b), i.e., $\frac{\texttt{\# of wrong predictions}}{\texttt{\# of total predictions}}$. The grader will compare the mean and standard deviation of your list with our solution; it must be within three standard deviations.