CS 412: Fall'24
Introduction To Data Mining

Assignment 3

(Due Wednesday, November 13, 11:59 pm)

- The homework is due on November 13th at 11:59 p.m. We will be using Gradescope for homework assignments. Please do NOT email a copy of your solution. Contact the TAs if you are having technical difficulties in submitting the assignment. Unfortunately, We will NOT accept late submissions without a reasonable justification.
- Please use Slack or Canvas first if you have questions about the homework. You can also join office hours and/or send us emails. If you are sending us emails with questions on the problems, please start the subject with "CS 412 Fall'24: " and send the email to *all of us* (Arindam, Chandni, Grace, Junting, and Meghna) 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. There are no post deadline tests. The grade appearing on Gradescope right after your submission will be your final grade for this assignment.

  – Two python files named `homework3_q1.py` and `homework3_q2.py` containing starter code are available on Canvas. Do NOT change the filename.

  – Do NOT add any additional `import` lines to your code. Required libraries have already been included in the starter code.

  – For submitting on Gradescope, you would need to upload both files, `homework3_q1.py` and `homework3_q2.py`.

1. (50 points) The problem will focus on developing your own code for: K-`fold cross 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`

```
from sklearn.datasets import load_digits
digits = load_digits()
X, y = digits.data, digits.target
```

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

   – 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 (do not specify any additional parameters):

   – `LinearSVC: max_iter=2000, random_state=412`
   – `SVC: gamma='scale', C=10, random_state=412`
   – `LogisticRegression: penalty='l2', solver='lbfgs', random_state=412, multi_class='multinomial'`
   – `RandomForestClassifier: max_depth=20, n_estimators=500, random_state=412`
   – `XGBClassifier: max_depth=5, random_state=412`

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

   (a) `get_splits`(n, k, `seed`), which returns: randomized k 'almost equal' sized lists of unique disjoint indices from the set of all indices $\{0, \ldots, n-1\}$, where the randomization depends on the integer `seed`. By 'almost equal', we mean the cardinality of the lists can differ by at most 1. These k list of indices correspond to the k folds over which cross-validation will be performed. The function will have the following **output**:

      i. a python list containing exactly k lists. Each of these sublists should be disjoint, each of size roughly $\frac{n}{k}$, contain elements from the set $\{0, 1, ..., n-1\}$ and must not contain repeated elements. The union of all the k sublists should include all elements in $\{0, \ldots, n-1\}$.

   Input `seed` determines the randomization and the output should be the same every time we use the same `seed` for a given n, k, and should be (randomly) different for different values of the `seed`.

For example, `get_splits`(4, 2, 1) may return [[0,2], [1,3]], and the output must be same every time with the same input; `get_splits`(4, 2, 73) may return [[0,3], [1,2]]; `get_splits`(7, 2, 2) may return [[0,2,4,6], [1,3,5]]; `get_splits`(11, 3, 7) may return [[0,3,6,9], [1,4,7,10], [2,5,8]].

For our tests, n would be less than 1200. `get_splits` would have a time limit of 10 seconds.

(b) `my_cross_val(method`, X, $\mathbf{y}$, `splits`), which runs k-fold cross-validation for `method` on the dataset $(X, \mathbf{y})$. The **input parameters** are:

   i. `method`, which specifies the (class) name of one of the five classification methods under consideration,
   ii. X, $\mathbf{y}$ which is the data for the classification problem
   iii. `splits`: the output of the `get_splits method (len(splits) =` k`)`

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

   i. 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 (auto)grader will judge your solution as correct if the difference between the reported and the expected mean error rates is within 10^{-3}. `my_cross_val` would have a time limit of 2 minutes.

   Within `my_cross_val`, strictly use the *splits* encoded in the input parameter `splits`. Do not define your splits for K-fold cross validation within this method.

   Also, make sure that you are NOT inadvertently shuffling your training data during K-fold cross validation. The training examples (for any particular split) should be in the same order as the input X.

Use `my_cross_val` to return the error rates in each fold for k fold cross-validation for the specific `method` (one of `LinearSVC`, `SVC`, `LogisticRegression`, `RandomForestClassifier`, and `XGBClassifier`) with the parameters outlined above.

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. `my_train_test` would have a time limit of 2 minutes.