CIS 419/519 Introduction to Machine Learning
Homework 1

Due: February 5, 2020 11:59pm

Instructions

Read all instructions in this section thoroughly.

Collaboration: Make certain that you understand the collaboration policy described on the course website.

You should feel free to talk to other members of the class in doing the homework. I am more concerned that you learn how to solve the problem than that you demonstrate that you solved it entirely on your own. You may *discuss* the homework to understand the problems and the mathematics behind the various learning algorithms, but you must **write down your solutions yourself**. In particular, **you are not allowed to share problem solutions or your code with any other students.**

We take plagiarism and cheating very seriously, and will be using automatic checking software to detect academic dishonesty, so please don't do it.

You are also prohibited from posting any part of your solution to the internet, even after the course is complete. Similarly, please don't post this PDF file or the homework skeleton code to the internet.

Formatting: This assignment consists of two parts: a problem set and program exercises.

For the problem set, you must write up your solutions electronically and submit it as a single PDF document. We will not accept handwritten or paper copies of the homework. Your problem set solutions must use proper mathematical formatting.

All solutions must be typeset using LaTeX; handwritten solutions of any part of the assignment (including figures or drawings) are not allowed.

Your solutions to the programming exercises must be implemented in python, following the precise instructions included in Part 2. Portions of the programing exercise will be graded automatically, so it is imperative that your code follows the specified API. A few parts of the programming exercise asks you to create plots or describe results; these should be included in the same PDF document that you create for the problem set.

Homework Template and Files to Get You Started: Here are some resources you will need for this assignment

- A Latex template: https://www.seas.upenn.edu/~cis519/spring2020/homework/hw1_template.tex
- A Jupyter Notebook template, which contains the function definitions that you must implement: https://www.seas.upenn.edu/~cis519/spring2020/homework/hw1_skeleton.ipynb
- The `hw1-NHANES-diabetes-train.csv` training dataset: https://www.seas.upenn.edu/~cis519/spring2020/homework/hw1-NHANES-diabetes-train.csv
- The `hw1-NHANES-diabetes-hw-test-unlabeled.csv` test data: https://www.seas.upenn.edu/~cis519/spring2020/homework/hw1-NHANES-diabetes-test-unlabeled.csv
- The t-table [519 only]: https://www.seas.upenn.edu/~cis519/spring2020/homework/t-table.pdf

Citing Your Sources: Any sources of help that you consult while completing this assignment (other students, textbooks, websites, etc.) ***MUST*** be noted in the your PDF file. This includes anyone you briefly discussed the homework with. If you received help from the following sources, you do not need to cite it: course instructor, course teaching assistants, course lecture notes, course textbooks or other readings.

Submitting Your Solution: You will submit this assignment through Gradescope. There will be two parts that you need to submit: one for the written problem set (a PDF), and another for the programming exercise (python code).

When you submit your code, we will run some automatic tests on it. You can resubmit the python code as many times as you need to, so don't worry if you don't get it on the first try.

CIS 519 ONLY Problems: Some parts of the assignment will be marked as "[CIS 519 ONLY]". Only students enrolled in CIS 519 are required to complete these problems. However, we do encourage students in CIS 419 to read through these problems, although you are not required to complete them.

All homeworks will receive a percentage grade, but CIS 519 homeworks will be graded out of a different total number of points than CIS 419 homeworks. Students in CIS 419 choosing to complete CIS 519 ONLY exercises will not receive any credit for answers to these questions (i.e., they will not count as extra credit nor will they compensate for points lost on other problems).

PART I: PROBLEM SET

Your solutions to the problems will be submitted as a single PDF document. Be certain that your problems are well-numbered and that it is clear what your answers are.

1 Decision Tree Learning (15 pts)

The following table gives a data set for deciding whether or not to hospitalize a patient.

PainLocation	Temperature (F)	Sp02 (%)	Tachycardic	Class
head	98	96	false	Outpatient
head	97	98	false	Outpatient
head	98	95	true	Admit
head	98	92	true	Admit
head	97	80	true	Admit
extremities	98	80	true	Admit
extremities	102	98	true	Outpatient
extremities	101	95	false	Outpatient
extremities	98	97	false	Outpatient
extremities	98	90	false	Admit
chest	97	70	true	Admit
chest	104	98	false	Admit
chest	98	98	true	Admit
chest	100	90	false	Admit

(a) (4 pts.) At the root node for a decision tree in this domain, what are the information gains associated with the PainLocation and Temperature features? (Use a threshold of 99 for temperature (i.e., assume a binary split: temperature < 99 / temperature ≥ 99). Be sure to show your computations.

(b) (4 pts.) Again at the root node, what are the gain ratios associated with the PainLocation and Temperature features (using the same threshold as in (a))? Be sure to show your computations.

(c) (4 pts.) Draw the complete (unpruned) decision tree, showing the class predictions at the leaves. Use the threshold from (a) for the Temperature, but you should determine your own threshold(s) for Sp02. You may very neatly hand draw the tree or draw it using a graphics program.

(d) (3 pts.) Does ID3 guarantee a globally optimal decision tree? By optimality, we mean a decision tree that perfectly fits the training data and also has a minimal depth. Justify your answer briefly.

2 [CIS 519 ONLY] Decision Trees & Linear Discriminants (10 pts)

Describe **in detail** how to modify a classic decision tree algorithm (ID3 / C4.5) to obtain oblique splits (i.e, splits that are *not* necessarily parallel to an axis).

PART II: PROGRAMMING EXERCISES

We highly recommend that you use Google Colab for these programming exercises.

But, if you decide to work on your own computer, you will need to install the following software:

- python 3.7.x or later (https://www.python.org/downloads/)
- numpy (http://www.numpy.org/)
- scipy (http://www.scipy.org/)
- scikit-learn (http://scikit-learn.org/stable/)

The instructions for installing scikit-learn already include the instructions for installing numpy and scipy, so we recommend that you start there.

Please note that we will provide assistance with colab, but cannot provide any system support or installation help if you decide to work on your own computer.

3 Improving Diabetes Classification

The decision tree for diabetes classification that we learned in class relied heavily on the "Glycohemoglobin" feature. In fact, this feature is the value of the specialized A1C blood test that is commonly used to determine whether or not a patient has diabetes – it represents a sort of average blood sugar of the patient over time. A1C works well for diagnosing whether or not a patient actually has diabetes if that disease is suspected. However, what about *before* diabetes is suspected and that specialized test is ordered by a physician?

In this problem, you will try to determine a set of alternative features that could be used to diagnose diabetes. The resulting decision tree could be useful, for example, for primary care physicians to screen for diabetes or in locations without immediate access to the A1C blood test.

We will first build two useful utility functions, and then use these functions to solve the problem.

3.1 Implementing Cross-Validation (15 pts)

In order to evaluate the performance of the DT classifier, you will use cross-validation. The purpose of cross-validation is to estimate how well a model will generalize on unseen data.

Write a function to calculate the mean cross-validated accuracy. Your function will take in a `DecisionTreeClassifier` (that has it's own set of parameters), the number of trials `num_trials` and the number of folds `num_folds` for the cross-validation. You should measure the accuracy of each trial and fold combination, and return the mean accuracy of the whole cross-validation experiment.

Your function must use the following prototype:

```
cross_validated_accuracy(DecisionTreeClassifier, X, y, num_folds, num_trials, random_seed)
```

and return the mean cross-validated accuracy of the classifier.

3.2 Automatic Decision Tree Pruning (15 pts)

The sklearn `DecisionTreeClassifier` uses a different form of pruning from what we covered in class, called Minimal Cost-Complexity Pruning. We won't cover the specifics, but you can learn more at https://scikit-learn.org/stable/modules/tree.html#minimal-cost-complexity-pruning if you wish.

But, you don't need to learn the details in order to use it effectively. The amount of pruning is entirely dependent on the value of the `ccp_alpha` parameter, which we can tune using cross-validation, as described in the lecture slides.

Implement a function to do automatic tuning of the `ccp_alpha` parameter. Your function will take in a `DecisionTreeClassifier` (which might have other parameters set), and will vary the value of the `ccp_alpha`

parameter. It should return the value for `ccp_alpha` with the highest cross-validated accuracy over the given dataset.

Your function must use the following prototype:

`automatic_dt_pruning(DecisionTreeClassifier, X, y, num_folds, num_trials, random_seed)`

and return the best value for `ccp_alpha`.

3.3 Training the New Decision Tree (15 pts)

Now that we have built these two utility functions, use them to solve the original problem of finding an alternative diabetes decision tree that doesn't rely on the Glycohemoglobin A1C feature.

The diabetes dataset that we are using is from the 2013-2014 National Health and Nutrition Examination Survey (NHANES). Originally, we reduced the dataset to only 14 features (one of which was the Glycohemoglobin A1C) for diabetes prediction, but our original dataset had over 1,800 features.

Without using the Glycohemoglobin A1C feature, your goal is to find an alternative set of features that we could use to build the decision tree. Of course, you want the performance of your decision tree to be as high as possible, and ideally close to the performance of including the Glycohemoglobin A1C feature that we obtained in lecture.

The data for this problem is available in the `hw1-NHANES-diabetes-train.csv` file at the URL listed in the instructions. Note that it does not contain the Glycohemoglobin A1C feature used in the lecture examples, but it does contain around 1,800 additional features. Many of these feature values are missing throughout the dataset.

Write an iPython notebook to solve this problem, following the given template. Your iPy notebook should:

- Load the dataset from the csv file
- Extract features that you have chosen. A good start might be to use the features we used in class. Look up your chosen features in the NHANES data dictionary: `https://wwwn.cdc.gov/nchs/nhanes/search/default.aspx` to really understand their values. Are these reasonable features to assume that (e.g.) a primary care physician would have access to?
- Preprocess the data as you so choose (e.g., binarize or one-hot-encode features, eliminate outliers, etc.). The preprocessing is entirely up to you, so experiment!
- Impute missing values with the mean of numeric features, or mode for categorical features
- Report the 10-fold cross-validated accuracy of your decision tree model, averaged over 10 trials

In addition, you should include a brief report (1/2 page) in your PDF writeup that

1. Describes your pre-processing and the best set of features you found
2. Describes your parameters for training the decision tree
3. Presents a performance table that compares the performance of three (3) potential sets of features, one of which is your chosen best set.

Please note that **all** of your dataset pre-processing must be done in the iPython notebook; you cannot download the CSV file and manually process it into an alternate version of the dataset.

3.4 Making Predictions (10 pts)

Our ultimate goal is prediction on new data, so use your best trained decision tree to output predictions on the provided `hw1-NHANES-diabetes-test-unlabeled.csv` data set. Be sure to pre-process the data exactly as you did before when training the model.

Output your predictions into the file `cis519_hw1_predictions.csv`, with one row per prediction. Be certain to keep your predictions in the same order as the test instances.

3.5 Computing Confidence Intervals [CIS 519 ONLY] (10 pts)

Even though you may have computed the average test accuracy across the held-out folds during cross validation, how confident can you be that the number you computed is the true average accuracy for that set of features? If you try rerunning your code with a different random seed, you may actually get a different accuracy score. But which one is right?

In order to answer this question, we will compute a confidence interval based on the Student's t-distribution, which will tell us with 99% confidence that the true mean is within a lower and upper bound. To compute the confidence interval, we need to compute the sample mean, $\bar{x}$, sample standard deviation, S, and the number of observations for each classifier, n. In our specific case, the number of observations should be 100 because we have 100 reported accuracies from cross-validation. The mean and standard deviation can be computed with `numpy` as follows:

```
import numpy as np
A = [...]  # your actual accuracies
x_bar = np.mean(A)
S = np.std(A)
n = len(A)
```

Then, the confidence interval is computed by

$$\bar{x} \pm t \cdot \frac{S}{\sqrt{n}}$$

Here, t is the critical value, which we can look up using the provided t-table. If we are looking for a 99% confidence interval, then the number in the 99% confidence column with degrees of freedom of $n - 1 = 4$ would be $t = 4.604$. Then, we can plug in all of the statistics into the confidence interval formula and get a range of values for which we are 99% confident that the true accuracy of the classifier falls between.

Includes these confidence intervals in the performance table in your PDF writeup.

What to Turn In

- Your PDF writeup containing the results of the written problems and a report of your results on the programming exercise
- `cis519_hw1_solution.ipynb` – your complete iPython notebook solution to HW 1
- `cis519_hw1_utilities.py` – Copy your implementations of the two functions `cross_validated_accuracy()` and `automatic_dt_pruning()` along with any required imports and helper functions you wrote into a separate `.py` file. This will allow us to provide automatic feedback on your solution. Be certain that you don't have any other code outside of these functions besides the import statements.
- `cis519_hw1_predictions.csv` – the csv file of your predictions on the test data, one prediction per row. Be certain that the order of these predicted labels is the same as the order of the test data.