CS 412: Introduction to Data Mining	**Spring 2019**
Homework 3	
Handed Out: Mar 29, 2019 12:00 pm	*Due: Apr 14, 2019 11:59 pm*

1 General Instructions

- The programming assignment will be hosted on hackerrank (`https://www.hackerrank.com/`) as a programming contest. To participate in this contest, please open a hackerrank account with your illinois.edu email netid. If you already have a username in hackerrank that is different from your netid, please fill out your netid and username in this spreadheet (`https://docs.google.com/spreadsheets/d/1AHRYivBFDm77-Xigy4jP7MZumiotqkiDJ9ZZU-DyoHA/edit?usp=sharing`).
- It is OK to discuss the problems with your classmates. However, it is NOT OK to work together or share code. You need to write your code independently and the TAs will not do the code debugging. Plagiarism is an academic violation to copy, to include text from other sources, including online sources, without proper citation. To get a better idea of what constitutes plagiarism, consult the CS Honor code (`http://cs.illinois.edu/academics/honor-code`) on academic integrity violations, including examples, and recommended penalties. There is a zero tolerance policy on academic integrity violations. Any student found to be violating this code will be subject to disciplinary action.
- Please use Piazza if you have questions about the homework.
- **Late policy**: 10% off for one day, 20% off for two days, 40% off for three days.

2 Programming Assignment Instructions

In this programming assignment, you are required to implement a contiguous sequential pattern mining algorithm and apply it on text data to mine potential phrase candidates. Participate in the programming contest hosted at hackerrank: `www.hackerrank.com/cs412-sinha-hw3`.

- Please read the problem description carefully.
- The input will always be valid. We are mainly testing your understanding of frequent sequential pattern mining, not your coding skills.
- Please pay special attention to the output format. We will be using the hackerrank based autograder and it is extremely important that your generated output satisfies the requirement.
- We don't have specific constrains for this programming question. The only constrains are the standard environment constraints in hackerrank: `https://www.hackerrank.com/environment`.

- The grading will be based on if you pass the test cases. You are provided with one sample test case to debug your code. For the final grading, we will use additional test cases to test your code.

3 Programming Assignment

This assignment aims to familiarize you with the mechanism of two widely-used classification methods: decision tree (DT) and k-nearest neighbors (KNN).

Specifically, the problem in this assignment is a multi-class classification problem with continuous feature/attribute values. For the decision tree you are to implement, please always use binary split and a threshold to split data. That is, each decision node has the form

If attribute $X \leq$ threshold θ; then *left node*; else *right node*.

where the best X and θ are for you to determine. Please use Gini impurity to construct the decision tree.

For k-nearest neighbors, please use Euclidean distance to compute the distance between any pair of nodes.

Model specifications

For DT, let max_depth = 2 be the only stopping criterion. In other words, you should grow the tree as long as max_depth = 2 is not violated and not all training instances in the current node have the same label. In face of multiple choices of threshold θ with the same resulting Gini impurity, always choose the one with the largest margin. (We will illustrate this point with example at a later stage.)

For KNN, set $k = 3$.

[**The following design choice in implementation purely aims to ease auto-grading on HackerRank.**] We ensure each attribute is named by a **non-negative integer**, and each class label is named by a **positive integer**. Since we are to use HackerRank for grading, we have to eliminate additional randomness and generate the deterministic results. We therefore enforce the following rule in this assignment: In the event of ties, always choose the attribute or label with the smallest value. Namely,

- When training a DT, if splitting on either attribute X_1 or X_2 gives you the best Gini impurity, choose the smaller of X_1 and X_2.

- When determining the nearest neighbors, if at most one of two training instances can be included into the set of k nearest neighbors, but they have two distinct labels L_1 and L_2, include the one with smaller of L_1 and L_2.
- In prediction, if both label L_1 and L_2 have the same number of training instances at a leaf node of a DT or in the set of k nearest neighbors, predict the smaller of L_1 and L_2.

For all test cases in this assignment, we guarantee that deterministic results can be generated as long as the aforementioned requirements are satisfied.

Input Format and Sample

Each input dataset contains training instances followed by test instances in a format adapted from the *libsvm* format. Each line has the form

- [label] [attribute 1]:[value 1] [attribute 2]:[value 2] ...

which is space-separated. As mentioned above, the name of each attribute, e.g., [attribute 2], is a non-negative integer and the value of an attribute, e.g., [value 2], is a float number. A line stands for **a test instance if [label] is 0** and a training instance otherwise. The label of a training instance can be any positive integer number.

We provide the following toy input example alongside visualization in Figure 1.

```
1 0:1.0 2:1.0
1 0:1.0 2:2.0
1 0:2.0 2:1.0
3 0:2.0 2:2.0
1 0:3.0 2:1.0
3 0:3.0 2:2.0
3 0:3.0 2:3.0
3 0:4.5 2:3.0
0 0:1.0 2:2.2
0 0:4.5 2:1.0
```

Again, you may assume the test instances ([label] = 0) are at the end of the input file. Note that please do not assume the attribute names to start from 0 or to be consecutive integers, and please do not assume the class labels to start from 1 or to be consecutive integers.

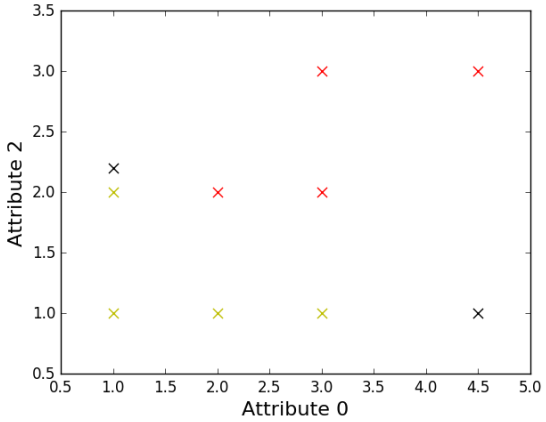

Figure 1: Visualization of the input of the toy example. The black dots represent test instances, and the others are training instances.

In this example the goal of the classification task is to predict the labels of the black dots ([label] = 0) to either label 1 (yellow) or label 3 (red) as illustrated in Figure 1. Note that when building the decision tree, the first split would have the form "If attribute 2 $\leq$ 1.5; then *left node*; else *right node*." Threshold θ is learned to be 1.5 because even though values like 1.3 or 1.6 can result in the same Gini impurity, 1.5 gives the largest margin to training instances on either side.

Output Format and Sample

The output is **the prediction on the test instances** made by your DT followed by that made by your KNN. An empty line is used to separate the results from the two models. For each method, print the prediction for each test instance per line, following the order in the input file.

As an example the output of the toy example is as follows.

```
1
1

1
3
```