COMP307/AIML420 — Fundamentals of AI

# Assignment 1: Machine Learning Algorithms

*16% of Final Mark for COMP307, 15% of Final Mark for AIML420*

Due date: 11:59 PM - 27/03/2024 (Wednesday)

## Objectives

The goal of this assignment is to enhance your comprehension of fundamental concepts and algorithms in machine learning, develop computer programs to execute these algorithms, utilize these implementations for classification tasks, and analyze the outcomes to draw meaningful conclusions. The assignment covers the following topics:

- Basic machine learning concepts;
- Nearest neighbour method;
- Decision tree.

These topics are covered in week 1 (basic ML) and week 2 (kNN and DT). You can also refer to the textbook and online materials for additional resources. IMPORTANT: You are required to implement the learning algorithms in this assignment. You will not receive any marks if you use machine learning libraries such as scikit-learn or any other.

## General coding requirements

- **Programming language.** Preferably, you may write the code in Java or Python. However, you may use another programming language as long as it can be easily run on the ECS systems. **IMPORTANT:** If you choose a language other than Python or Java, ensure that you dedicate sufficient time to preparing your README. If your code cannot be executed or if there are unclear instructions, it will impact your grading.
- **Data**
  - **Reading and manipulating.** You can use libraries such as Pandas and Numpy to read and manipulate the data, or to generate random numbers (i.e. **numpy.random**). **IMPORTANT:** This doesn't mean you can use a machine learning library to implement the algorithms. If you utilize a machine learning library like scikit-learn to implement a learning algorithm in this assignment, you will receive **zero marks for that part of the assignment, including the analysis (PDF report)**.
  - **Datasets characteristics.** Your implementations should be flexible enough to handle other datasets with characteristics similar to the ones presented for each part of the assignment. More details below:
    - Part 1: *The first line of each file should contain a header with the names of the columns. The last column represents the class label.* **All features are numerical, and the class is nominal, but it is not considered a feature. This part does not need to handle nominal attributes.**
    - Part 2: *The first line of each file should contain a header with the names of the columns. The last column represents the class label.* **All features are nominal, represented as integers from 0 to $m-1$, where $m$ is the total number of possible nominal values for the feature. This means you can immediately read the nominal attribute values and use them; there's no need for mapping from string values to integers (you will learn to appreciate this while coding). These parts do not need to handle numerical attributes.**
  - Train and Predict. For each learning algorithm, ensure that the methods for training and prediction in your implementation are clearly identifiable. It should not be challenging for the marker to locate them. Ideally, consider separating files for each learning algorithm.
- **Metric**. The only metric that to be used in this assignment is **classification accuracy**. You don't need to use a machine learning library to calculate the accuracy, refer to the lecture slides for how to calculate accuracy.

## Part 1: k-Nearest Neighbour (40% marks)

### Data

The wine dataset was retrieved from the UCI Machine Learning Repository [1]. It comprises 178 instances divided into 3 classes, with 59, 71, and 48 instances, respectively. Each instance consists of 13 attributes: Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium, Phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins, Color intensity, Hue, OD (OD280/OD315 of diluted wines), and Proline. The dataset has already been split into **training** and **testing** sets.

### Requirements

You should implement the kNN algorithm such that it can be trained on the training data and used to make predictions on the test data. The class is identified as "class" and is the last column in the training and testing files. More details about the algorithm can be found on the lectures. **(AIML420 mandatory, optional for COMP307): Implement Min-Max normalization for each feature. When normalizing the training and test sets, it is imperative to ensure consistency and prevent data leakage.**

**INPUT:** Your program should take four arguments as command line arguments: file name for training data, file name for testing data, the file name for the output file, and k (the number of neighbors).

Example: **kNN.exe train.csv test.csv output_file.csv 3**.

**OUTPUT:** Print the accuracy on the second argument (i.e. test file) to the terminal and create a file (use the file name argument) containing the following information for each instance in the test set (second argument to your program): the original class label from the test data, the predicted class label according to your kNN implementation, and the distance between the test instance and each of the neighbors (if $k > 1$ then more columns will be needed in here). Example (assuming $k$ was 3, and only 5 rows in the test set):

| y | predicted_y | distance1 | distance2 | distance3 |
|---|---|---|---|---|
| 0 | 1 | 21.2 | 22.5 | 67.0 |
| 1 | 1 | 11.3 | 15.2 | 19.6 |
| 0 | 0 | 1.4 | 2.2 | 3.8 |
| 1 | 1 | 8.9 | 9.1 | 10.5 |
| 0 | 1 | 18.7 | 19.3 | 22.1 |

Table 1: Example output for kNN

You should submit the following files electronically:

1. **(25% marks) Program code** for your k-Nearest Neighbour Classifier (the source code as well as the executable program that runs on the ECS School machines [2])
2. **README** which describes how to run your program
3. **(15% marks) A report in .pdf format** and output files. The report should include the answers to the questions below, and the outputs should correspond to some of the questions:
   (a) Report the classification accuracy on the **test set** using **k=1** and **k=3**. Include the output files corresponding to each experiment (suggested names: knn1_test.csv and knn3_test.csv)
   (b) Report the classification accuracy on the **train set** using **k=1** and **k=3**. Include the output files corresponding to each experiment (suggested names: knn1_train.csv and knn3_train.csv).
   (c) When analyzing the results from the test and train sets using k=1 and k=3, consider the following questions:
      i. Does a specific k value maximize accuracy in the training set? If yes, does this optimal k value also improve accuracy in the test set?

[1] https://archive.ics.uci.edu/ml/datasets/wine
[2] If you use Python or other interpreted languages, the source and executable are the same

ii. Why does using this optimal k value (in the train data) work well or not work well for the test set?

(d) Discuss the consequences of increasing and decreasing k on the algorithm's performance. Additionally, consider scenarios where k is large, for example k equals the total size of the dataset. How does these extreme values of k influence the behavior and effectiveness of the kNN algorithm?

## Part 2: Decision Tree Learning Method (60% marks)

### Data

The datasets (rtg_A, rtg_B, and rtg_C) used for this section of the assignment were synthetically generated, lacking a specific domain background. The three datasets contains only nominal attributes (some are binary, some have more than 2 possible values) and 2 possible class labels. For this part of the assignment, only the training datasets are provided. The focus is not on evaluating accuracy on a test set but rather on analyzing the generated tree. **IMPORTANT:** rtg_C is only required for AIML420.

### Requirements

You are tasked with implementing a simplified version of the decision tree algorithm, reminiscent of the classical ID3 (Iterative Dichotomiser 3) [3] algorithm. More details about the algorithm can be found on the lectures.

**INPUT:** Your program should take two arguments as command line arguments: file name for training data [4], and file name for the output (tree).

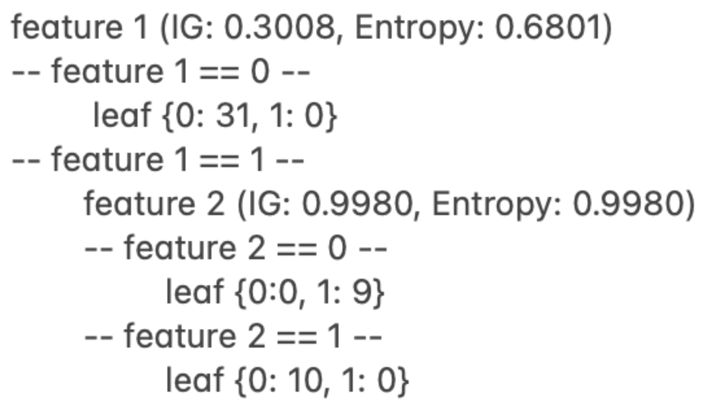
Example: **DecisionTree.exe train.csv output_tree.txt**

**OUTPUT:** Print the accuracy on the first argument (i.e. train file) and create a file (use the output file name) containing the decision tree generated for the training data. The decision tree output should include the following essential information:

- For split nodes: specify the features used for splitting, the Information Gain (IG), and the entropy.
- For leaves: specify the class counters associated with each leaf node.
- For edges: identify the values associated with the edges between nodes.

### Example tree output

Note that the most straightforward method for outputting the tree involves performing a depth-first traversal of the tree structure. Example [5] shown in Figure 1:



```
feature 1 (IG: 0.3008, Entropy: 0.6801)
-- feature 1 == 0 --
      leaf {0: 31, 1: 0}
-- feature 1 == 1 --
     feature 2 (IG: 0.9980, Entropy: 0.9980)
     -- feature 2 == 0 --
           leaf {0:0, 1: 9}
     -- feature 2 == 1 --
           leaf {0: 10, 1: 0}
```

Figure 1: Example output tree (text)

In this example, there are two split nodes and three leaves. Each split node must indicate the feature it is splitting on. In our case, the first split occurs on "feature 1," while the second split occurs on "feature 2." Additionally, split nodes should provide information on the information gain (IG) and entropy of that node.

[3] *Quinlan, J. Ross. "Induction of decision trees." Machine learning 1 (1986): 81-106.*

[4] Why isn't there a test set? This segment of the assignment directs your attention to comprehending the structure of the resulting tree, rather than conducting numerous experiments and verifying accuracy on test sets.

[5] This example is not related to any data file used in the assignment

The leaf nodes contain counters for the instances reaching them based on the class label. For instance, leaf 0: 31, 1: 0 indicates that 31 instances reaching that node belong to class 0, while no instances of class 1 reach that node.

It's essential for the edges to specify the value of the feature corresponding to that branch. For instance, "− feature 1 = 0 −" signifies that the node below is reached when feature 1 equals 0. Another view of the same example is shown in Figure 2, you don't need to provide these visualizations (the text version is enough).

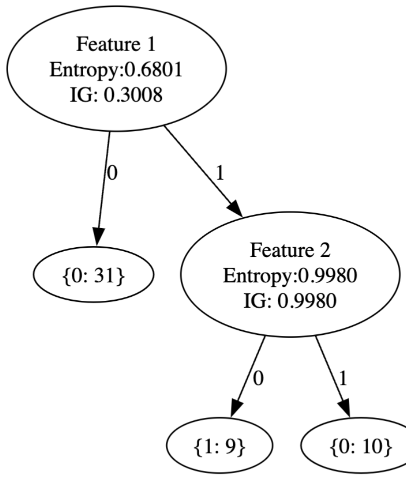


Figure 2: Example output tree (graph) - leaf counters for classes with 0 instances were omitted.

## Impurity measure and stopping criteria

You should use the Entropy impurity measure and Information Gain presented in the lectures. To determine when to stop growing the tree, use a fixed threshold of 0.00001 for the Information Gain (i.e. if the IG from a split is less than 0.00001, do not split). Notice that if there are no more instances to split on, no further splits will take place as well.

1. (40% marks) **Program code** for your decision tree classifier (the source code as well as the executable program that runs on the ECS School machines).
2. **README** describing how to run your program.
3. (20% marks) **A report in .pdf format** and output files. The report should include the answers to the questions below, and the outputs should correspond to some of the questions:
   (a) Report the classification accuracy for "rtg_A" and the decision tree output. Include the output file corresponding to the decision tree (suggested name: DT_A.csv).
   (b) Report the classification accuracy for "rtg_B" and the decision tree output. Include the output file corresponding to the decision tree (suggested name: DT_B.csv).
   (c) When analyzing the results for rtg_A and rtg_B, consider the following questions:
      i. Which feature results in the minimal entropy when considering the entire training dataset? How can this observation be made?
      ii. Were there any features that were not utilized in constructing the tree for both datasets? If so, what factors contributed to their exclusion from the tree-building process? Tip: you might want to observe the data to answer this question.
   (d) **(AIML420 mandatory)** Report the classification accuracy for "rtg_C' and the decision tree output.
   (e) **(AIML420 mandatory)** When analyzing the results for rtg_C, consider the following questions:

i. **(AIML420 mandatory)** Is the generated tree "good" for making predictions on unseen data? Justify your question based on the tree structure.

ii. **(AIML420 mandatory)** Were there any features that were not utilized in constructing the tree for both datasets? If so, what factors contributed to their exclusion from the tree-building process? Tip: you might want to observe the data to answer this question.

iii. **(AIML420 mandatory)** Are there any preprocessing steps that you would consider to potentially enhance the generalization ability of the generated tree? Note that while we lack a test set, you can draw conclusions based on the tree itself and how features were employed by the tree.

## Submission Guidelines

## Marking and helpdesks

We will endeavour to mark your work and return it to you as soon as possible, hopefully in 2 weeks. The tutors will run a number of help desks to provide guidance (but won't tell you the answers!).

### Submission Requirements

1. Programs for all individual parts. To avoid confusion, the programs for each part should be stored in separate directories **part1/** and **part2/**. In each directory, please provide a README file that specifies how to run (and compile) your programs on the ECS School machines. Output files should also be provided depending on each part of the assignment. If your programs cannot run properly, provide a buglist file which details what does and doesn't work.

2. **You can have a single PDF report for all parts or separate PDFs per part**. Regardless, of your choice, make sure to identify where are the answers for each part.

### Submission Method

The programs and the report should be submitted through the web submission system from the COMP307 or AIML420 course web site **by the due time**. Please make sure you submit to the course you are enrolled in.

Please check **again** that your programs can be run on the ECS machines easily according to your **README**. If the markers can't run your code, you will **lose marks!** Each marker has a limited amount of time to get your code running, so please don't ask them to use Pycharm, IntelliJ IDEA, Visual Studio, etc to run your code. All these IDEs support exporting runnable code.

### Late Penalties

The assignment must be submitted on time unless you have made a prior arrangement with the course **cordinator** or have a valid medical excuse. We use the ECS extension system for all extension requests. Please make a request there if you think you have a valid reason.

The penalty for assignments that are handed in late without prior arrangement is one grade reduction per day. Assignments that are more than one week late will not be marked.

### Plagiarism

Plagiarism in programming (copying someone else's code) is just as serious as written plagiarism, and is treated accordingly. Make sure you explicitly write down where you got code from (and how much of it) if you use any other resources asides from the course material. Relying heavily on borrowed code may lead to deductions in marks, whereas plagiarism warrants a zero mark and may entail disciplinary actions.