# AI534 — Implementation Assignment 4 —

## General instructions.

1. Please use Python 3 (preferably version 3.6+). You may use packages: Numpy, Pandas, and matplotlib, along with any from the standard library (such as 'math', 'os', or 'random' - for example).

2. You should complete this assignment alone. Please do not share code with other students, or copy program files/structure from any outside sources like Github. Your work should be your own.

3. Submit your report on Canvas and your code on TEACH following this link:
`https://teach.engr.oregonstate.edu/teach.php?type=assignment`.

4. Please follow the submission instructions for file organization (located at the end of the assignment description).

5. Please run your code before submission on the EECS servers (i.e. babylon). You can make your own virtual environment with the packages we've listed in either your user directory or on the scratch directory.

6. Be sure to answer all the questions in your report. You will be graded based on your code as well as the report. The report should be clear and concise, with figures and tables clearly labeled with necessary legend and captions. The quality of the report and is worth 10 pts. The report should be a PDF document.

7. In your report, the **results should always be accompanied by discussions** of the results. Do the results follow your expectation? Any surprises? What kind of explanation can you provide?

# Decision Tree & Ensembles for Mushroom Classification (total points: 90 pts + 10 report pts + bonus)

In this assignment we will work on the Mushroom Classification task to classify between edible and poisonous based on a set of discrete features related to each species of mushroom. The goal in this assignment is to develop variations of the **decision tree** algorithm and ensemble methods.

**Data.** The data for this assignment is from the UCI Machine Learning Repository. Each sample describes the characteristics of a mushroom with 22 categorical features, followed by the class (0 for poisonous or 1 for edible). For this assignment, all of the categorical features have been processed and turned into one-hot encoding, resulting in 117 binary features. We have broken down the data into the training set (mushroom-train.csv, 4500 examples) and validation set (mushroom-val.csv, 1600 examples). There is also a test set (2023 examples) that is reserved for the kaggle competition.

**Part 1 (50 pts) : Decision Tree.** For this part you will implement the decision tree learning algorithm discussed in class. Specifically, your algorithm will:

- Use information gain to build the tree with binary splits.
- Build the tree until it reaches a specified maximum depth ($dmax$) of the tree. That is, you can stop growing the tree if the maximum depth is reached, even if the leaf nodes are not yet pure. For this assignment, the depth of the tree refers to the length of the longest path of the tree. For example, Fig. 1 shows a tree of depth = 2.

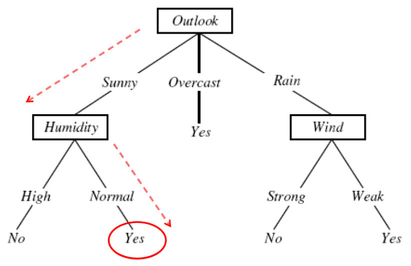

Figure 1: A depth 2 decision tree.

For this part of the assignment, you will experiment with your implemented algorithm with different maximum depth and observe its performance on training and validation data. Specifically, apply your implemented algorithm to learn from the training data with $dmax \in [1, 10]$. In your report, please answer the following questions.

(a) What are the first three splits selected by your algorithm? This is for the root, and the two splits immediately beneath the root. What are their respective information gains?

(b) Evaluate and plot the training and validation accuracies of your trees as a function of $dmax$ ranging from 1 to 10. At which depth does the train accuracy reaches to 100%? If your tree could not get to 100% before the depth of 10, keep on extending the tree in depth until it reaches 100% for the train accuracy. Do you observe any overfitting?

**Part 2 (35 pts). Random forest** In the second part, you will implement the random forest algorithm. Specifically, your algorithm will take the following inputs:

- $T$: the number of trees to include in your random forest.

- $m$: the number of features to sub-sample in each test selection step. You will need to modify the decision tree code to consider only $m$ randomly selected features when making the greedy selection.
- $dmax$: the maximum depth of the trees in your random forest.

For this part of the experiments, you will test your random forest algorithm with different $dmax$ values: 1, 2, 5. With each $dmax$ value, you will run experiments with different $m$ values, where $m = 5, 10, 25, 50$, grow your random forest with increasing size $T$ ranging from 10 to 50 in increments of 10 (note that for each combination of $m$ and $dmax$, you can do just one run to build 50 trees and then use subset of these trees for your experiments with different $T$ values).

For your report, please complete the following:

(a) For each $dmax$ value, create two figures, one for training accuracy and one for validation accuracy. The training accuracy figure should contain four curves, each showing the train accuracy (y-axis) of your random forest with a particular $m$ value as a function of $T$ (x-axis). Be sure to use different colors/lines to indicate which curve corresponds to which $m$ value, and include a clear legend to help the readability. Repeat the same process for validation accuracy. Compare your training curves with the validation curves, do you think your model is overfitting or underfitting for particular parameter combinations? And why?

(b) For each $dmax$ value, discuss what you believe is the dominating factor in the performance loss based on the concept of bias-variance decomposition. Can you suggest some alternative configurations of random forest that might lead to better performance for this data? Why do you believe so?

Be aware that these may take some time to run all experiments, as it requires the construction of many individual trees over all the experiments.

**Part 3 (5 pts) Kaggle competition.** We are hosting a in-class competition using the mushroom data. You must participate in single-person teams. The competition link will be posted on canvas. For this competition, we are focusing on ensemble methods. You can experiment with different hyperparamters, try out different variants for decision tree or the ensemble methods. The only limit is that the core algorithm must be your own implementation of the three ensemble algorithms of interest, namely, bagging, random forest and boosting. To get full points for this part, you need to list your kaggle team name, clearly describe the methods you used for your competition (code must be submitted for verification), and report the resulting performance you achieved on the private leaderboard. Top three teams will get 2 bonus points.

**Bonus Part (20 pts) : AdaBoost (Boosting).** For this bonus part we will implement and experiment with boosted decision trees. Specifically, your algorithm will take the following inputs:

- $T$: the number of trees to include in your ensemble
- $dmax$: the maximum depth of the tree in your ensemble

Note that your implementation will need to modify the decision tree learning algorithm to work with an additional input $D$, which is a vector of size $N$ (train set size) and specifies a weight for each training instance. (Hint: you will need to change how probabilities are estimated to incorporate $D$ during the tree construction, e.g., in computing information gain and in assigning the leaf labels.)

Apply your implemented boosted decision tree with three different $dmax$ values: 1, 2 and 5. With each $dmax$ value, you will build ensembles with different $T$ values, ranging from 10 to 50 in increments of 10 (note that for each value of $dmax$, you can do just one run to build 50 trees and then use subset of these trees for different $T$ values)
For your report, you need to do the following:

(a) For each $dmax$ value, create a figure showing two curves, showing the accuracy (y-axis) on train and validation of your ensemble as a function of $T$ (x-axis). Be sure to use different colors/lines for train and validation and include a clear legend to help the readability. Compare your training curves with the validation curves, do you think your model is overfitting or underfitting for particular parameter combinations? And why?

(b) For different *dmax* values, discuss what you believe is the dominating factor in the performance loss based on the concept of bias-variance decomposition. Can you suggest some alternative configurations of ensemble that might lead to better generalization performance for this data?

**Submission instructions.** Your submission should include the following:
1) Your source codes with a readme file to specify the required packages, zipped in a single file submitted on TEACH. We require **One file for each Part**.
2) You do not need to generate plots in the submission code, please remember to include those in your report. You need to print out your code result of each question in the console since it's easier to check the correctness of your code. Generally speaking, we check the code by console outputs and plots by report.
3) Please do **not** upload your python virtual environment. You can include the **data** in the same directory of the code (unless you think the data file is too big to upload), which can be tested with "python your_code.py" directly.
4) Your report (see general instruction items 6 and 7 on page 1 of the assignment), which should begin with a general introduction section, followed by one section for each part of the assignment; The report should be in PDF, submitted on Canvas.
5) Please always put your name on the first page of the report. All graphs must be properly labeled, i.e. Graphs must contain: Title, labels along both axis. It's a good habit to number pages, sections of the report.