CS534 — 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. Your source code and report will be submitted through Canvas.

4. You need to follow the submission instructions for file organization (located at the end of the report).

5. Please run your code before submission on one of the OSU EECS servers (i.e. babylon01.eecs.oregonstate.edu). You can make your own virtual environment with the packages we've listed in either your user directory or on the scratch directory. If you're unfamiliar with any of this process, or have limited access, please contact one of the TA's.

6. We must be able to run your code on the server, otherwise, the assignment will result in a zero grade. If your code fails to run, you will be notified by the TA and be given 24 hours to correct it.

7. Be sure to answer all the questions in your report. You will be graded based on your code as well as the report. In particular, **the clarity and quality of the report will be worth 10 pts**. So please write your report in clear and concise manner. Clearly label your figures, legends, and tables. It should be a PDF document.

8. 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 and Random Forest
(total points: 90 pts + 10 report pts)

Data. We will continue using the same data as the previous assignment. This dataset consists of vehicle insurance customer demographics, as well as collected information related to the customers' driving situation. Your goal is to use this data to predict whether or not a customer may be interested in purchasing home insurance as well (this is your "Response" variable). The dataset description (dictionary) is included. **Do not use existing code from outside sources for any portions of this assignment. This would be a violation of the academic integrity policy.**

The data is provided to you in both a training set: **pa4_train.csv**, and a validation set: **pa4_dev.csv**, with an X and y for both (X being features, y being labels). You have labels for both sets of data. There is no need to do any preprocessing.

Preprocessing we have done for you. We have pre-processed the data to minimize non-essential work for this assignment. In particular, we have treated [**Gender, Driving_License, Region_Code, Previously_Insured, Vehicle_Age, Vehicle_Damage, Policy_Sales_Channel**] as categorical features. We have converted those with multiple categories (some that originally contained textual descriptions) into one-hot vectors. You are to leave these as is and not modify further for this assignment, but understand the process. The numeric and ordinal features [**Age, Annual_Premium, Vintage**] are discretized uniformly based on quantiles into bins using a Pandas functionality called 'qcut'. In real practice you do not need to discretize the continuous features for learning, but for this assignment, we performed this step to reduce the runtime for building the tree, as this gives a substantially lower amount of feature/value combinations to check. Continuous features in decision trees are often treated as categorical in nature, which can give potentially N values (one for each example). Packages optimize the run-time for these, but here it could be quite expensive in your own implementation.

1 Part 1 (55 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. All features in this dataset should have been converted into binary features for you. It is not necessary to use an intercept feature, and you should not include one.
- 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.

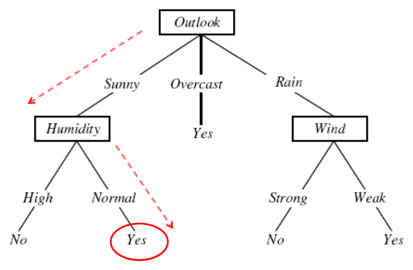

Figure 1: A depth 2 decision tree.

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 $dmax = 2, 5, 10, 20, 25, 30, ..., 50$. 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 $dmax$. When do you see your tree start overfitting?

2 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.
- $dmax$: the maximum depth of the trees in your random forest.

For this part of the experiments, you will test your random forest algorithm with three different $dmax$ values: 2, 10 and 25. With each $dmax$ value, you will run experiments with different m values, where $m = 5, 25, 50, 100$, grow your random forest with increasing size T ranging from 10 to 100 in increments of 10 (note that for each combination of m and $dmax$, you can do one run and build 100 trees and then use $10-100$ of these trees for your experiments). Please specify in NumPy a random seed of 1 - *np.random.seed(1)* - before running the tree building. This will keep results consistent.
For your report, please complete the following:

(a) For each $dmax$ value, create two figures, one for training accuracy and one for validation accuracy. For the training accuracy figure, it will contain four curves, each showing the train accuracy of your random forest with a particular m value as we increase the ensemble size $T = 10, 20, ..., 100$. That is, plot the training accuracy (y axis) as a function of the ensemble size T (x-axis), for each m value. Be sure to use different colors/lines to indicate which curve corresponds to which m value, and include a clear legend for your figure 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? Are there any issues inherent with the data you can find that make the performance increase difficult?

Be aware that these may take some time to run all experiments for, as it requires the construction of many individual trees over all the experiments. For reference, 50 trees of depth 25 with 20 features took about 4 minutes. This will likely be higher in cases with more features as well as more trees.

Submission. Your submission should include the following:
1) Your source code. **One file for each Part**. The files should be named (for example) **part1.py**, and should run with simply **python part1.py**.
2) Your report (see general instruction items 7 and 8 on page 1 of the assignment), which should begin with a general introduction section, followed by one section for each part of the assignment;
3) Please submit the report PDF, along with a .zip containing the code to Canvas. The PDF should be outside the .zip so it's easier to view the report.
4) Please include the data in your zip file to make it easier for the TAs to test your code. The code should work properly if run after unzipping your submission.