Due Date: 01.11.2019, 23:55

CENG 113
Homework #1

Preliminary: Running Scripts from Command-Line Interface

You can write your Python script using any text editor and run this script using terminal (without using any IDE). For example:

```
$ python hello.py
Hello, World!
```

In order to do this, first, you need to set the working directory to the directory which includes this script file using *cd* command (which stands for "change directory"). For example:

```
$ cd /home/ersin/ceng113
```

Alternatively, without changing the working directory, you can provide absolute path for the script file:

```
$ python /home/ersin/ceng113/hello.py
```

Python 2: In your system, *python* command may call Python 2 by default. You can check this by running the command without arguments and observing the console output (Type *exit()* to leave the interpreter).

```
$ python
Python 3.6.8 (default, Oct  7 2019, 12:59:55)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
```

If you encounter Python 2.x instead of Python 3.x, you should use *python3* command instead. For example:

```
$ python3 main.py
```

Command not found: In Linux and macOS, *python* command should be pre-defined. If you are using Windows, first you may need to edit *PATH* environment variable. If you need help, feel free to consult an assistant.

Preliminary: Command-Line Arguments

It is possible to pass some arguments to a Python script. This way of getting inputs from the user can be considered as an alternative to the *input* function. These arguments can make your script more practical and will enable different programs to run your script.

Let our script file main.py contain the following snippet:

```
import sys
script_name = sys.argv[0]
first_arg = sys.argv[1]
second_arg = sys.argv[2]
# …
```

If we run this script using the command

```
python main.py fast 0
```

we will have

```
script_name = 'main.py'
first_argument = 'fast'
second_argument = '0'
```

Part 1: Decision Tree

A decision tree is a simple model for predicting the value of an attribute given the values of a set of relevant attributes of an instance. Figure 1 shows a decision tree which can be used for predicting whether a particular football match will be played in time or delayed due to bad weather. Note that trees in computer science are conventionally drawn upside-down. Inference starts from the root (*Outlook*) and ends at a leaf (*Yes* or *No*).

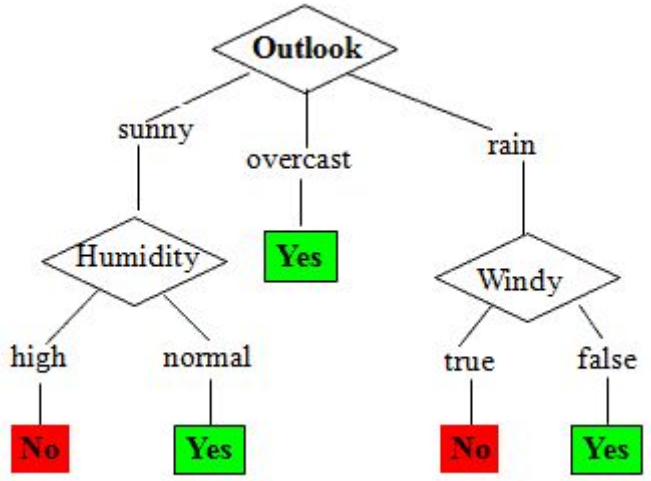

Fig. 1: A decision tree
[http://jcsites.juniata.edu/faculty/rhodes/ida/decisionTrees.html]

For example, our decision tree suggests the following instance be classified as "*Yes*".

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	normal	false	?

This tree was somehow "learned" by observing the data shown in Fig. 2. In this kind of tabular data, a row corresponds to an instance and a column corresponds to an attribute. In this assignment, however, we are not interested in "learning algorithms".

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Fig. 2: A sample of tabular data
[http://jcsites.juniata.edu/faculty/rhodes/ida/decisionTrees.html]

In part 1 of this assignment, please implement the given decision tree in a Python script file named decision_tree.py. This program should get user input through command-line arguments (You can test your program further, using the data shown in fig. 2):

```
$ python decision_tree.py sunny hot normal false
Yes
```

This script should be in the following form:

```
# Student ID: <write your student ID here>

import sys
outlook = sys.argv[1]
temperature = sys.argv[2]
humidity = sys.argv[3]
windy = sys.argv[4]

<Write your code here; do not use input or print>

print(play)
```

In your code, you should not use the value of the variable *temperature* (Our model ignores this attribute).

Part 2: Decision Forest

A decision tree may be prone to do certain kind of errors. Because of this, sometimes we may want to use a decision forest instead. A decision forest is a set of decision trees. When working with a decision forest, we perform inference on each decision tree independently. Then, we average over all the results from the individual trees (If we predict a nominal attribute, we calculate the mode - the most frequent value - since averaging over such data is impossible).

Figure 3 shows a decision forest of 3 binary decision trees (Binary decision trees is a special class of decision trees with binary yes/no splitting). If you are curious about the semantics of the attributes, see the Boston housing dataset: http://lib.stat.cmu.edu/datasets/boston (We predict the last attribute, *MEDV*). In the part 2 of this assignment, please implement this decision forest in a Python file named decision_forest.py.

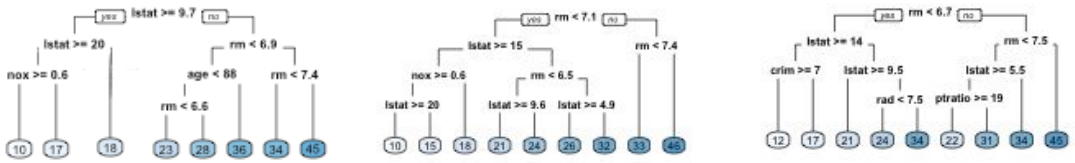

Fig. 3: A decision forest
From left to right; decision tree A, B and C.
[Decision tree 1 (one branch modified), 3 and 4 from https://uc-r.github.io/random_forests]

Similar to part 1, write your student ID at the beginning of your code. At the end of your code, print individual results of each of 3 decision trees (A, B and C) and the final result which is the average of these 3 results.

In this program, do not use command-line arguments. Instead, take user inputs using Python's *input* function in an interactive and lazy way: ask these values only when you need to know (and only if you need to know).

An example run of the desired program is shown below (Mind the colors: inputs and outputs of the program). The explanation in the second column is for helping you and not a part of the program.

lstat: 4.98 rm: 6.575 age: 65.2 rad: 1 Decision tree A: 23 Decision tree B: 26 Decision tree C: 24 Decision forest: 24.333333333333332	Needed for level 0 in tree A Needed for level 1 in tree A. Needed for level 2 in tree A. Needed for level 3 in tree C.

SUBMISSION RULES

- You should submit your solution to CMS until due date.
- Your homework should be named as below (Note that it is not a RAR file, but a ZIP file!):
  - ceng113_hw1_StudentID.zip (e.g. ceng113_hw1_123456789.zip)
    This compressed file should include 2 files:
    - decision_tree.py
    - decision_forest.py
- Write your student ID as a comment at the beginning of your code.
- Cheating, including teamwork, will not be tolerated.