(364-1-1441) Foundations of Artificial Intelligence

Problem Set 3: Leaving, on a Jet Plane

Due: 11/7/2024

You need to submit both code and written answers. Problem 1 is a programmatic one. Problem 2 only requires written answers and not programming. You will submit one `q1.py` file containing your code, and one `answers.pdf` file containing your written, typewritten (not a scan of handwriting) answers, in English or Hebrew.

Make sure your code compiles, and the output matches what is requested. Your grader will not debug your code, and if it does not compile or output correctly, they will not be in charge of fixing your errors, even if its cause was a very minor mistake.

Also, to simplify your work process, as well as the grader's, please name your variables and methods in a meaningful way (e.g., name a function `entropyCalculation` and not `myAwesomeFunction`).

1 Can You Predict a Late Flight?

This exercise uses data from the US Bureau of Transportation, which you will use to build a decision tree to predict if a flight will be delayed by more than 15 minutes or not. The data for this is in the file `flightdelay.csv`, which you can download from the course Moodle. The description of the data file is found in the `FlightColumns.pdf` file you can also download from there.

Broadly, the last variable indicates if the plane is delayed or not. The other variables show various other measurements, related to the airline, the plane, the airport and the day itself. Some of these variables are continuous (or just with many possibilities), so you will need to set thresholds yourself to "bucket" them – which means each of your trees might look different than someone else's (which is fine!).

When you build your decision trees, you are expected to use entropy to calculate the more meaningful attributes, and to use the χ^2 test to prune vertices.

Your Python code will include the following functions. You can assume the file `flightdelay.csv` is found in the same directory as your code.

`build_tree(<float> ratio)` for `ratio`$\in (0, 1]$, so function definition:

```
def build_tree(ratio):
```

You need to build a decision tree, using *ratio* ratio of the data (so if $ratio = 0.6$, you arbitrarily choose 60% of the data), and validate it on the remainder. The outcome is printing out the decision tree, and reporting the error.

`tree_error(<int> k)` so function definition:

```
def tree_error(k):
```

You need to report the quality of the decision tree by building k-fold cross validation (k being an integer number received by the function for the number of folds), and reporting the error (not k different error rates, but the average).

`is_late(<array> row_input)` so function definition:

```
def is_late(row_input):
```

You receive an input for a particular flight. It will be an array of values, in the same order as the data file (but without the column if the plane is late, of course). You return 1 if you think the plane will be late, and 0 if not. You should build your decision tree based on the full data.

Bonus: Particularly well-performing results on this problem may receive a bonus of up to 10 points, with the criteria and decision depending on the sole discretion of the course staff.

Obviously, you cannot use any library, package, module, or existing code that implements decision trees. You can use mathematical libraries for simple calculations, but everything, including entropy calculations and χ^2 tests needs to implemented by you.

2 Learning

Could you use both neural nets and decision trees together in a boosting algorithm? How would you use the boosting weights in the training of a deep neural net?