In our personal and professional lives, we are often presented with opportunities that require us to make critical decisions in the face of uncertainty: rent or buy, accept or reject, leave or stay, double down or cash out, and others.

Decision tree analysis is a simple yet powerful method for structuring and analyzing such problems. Wikipedia has [an article](https://en.wikipedia.org/wiki/Decision_tree) on the subject. In this post, I provide a brief summary of the method and a Python script for analyzing decision trees and rendering them as diagrams in a popular image format.

A decision tree consists of three types of nodes:

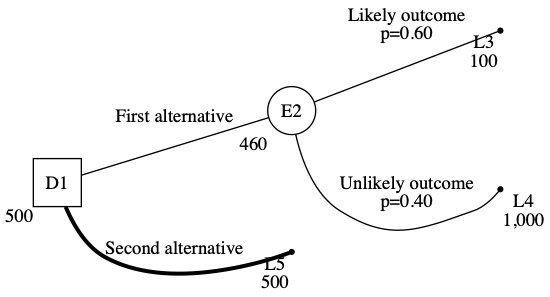
* an end node
* an event node
* a decision node

An end node represents an end state in your analysis. It has a utility value. It is shown as a dot in the tree diagram.

An event node represents an uncertain event that can have several outcomes. The outcomes are mutually exclusive and collectively exhaustive. An outcome has a probability value that reflects its likelihood. The probabilities of all outcomes must add to 1. An event node is shown as a circle in the diagram.

A decision node represents a decision. It has several alternatives pointing to other nodes in the tree. It is represented with a box in the diagram.

Here is a simple decision tree with one decision, one event, and three end nodes.



To analyze a decision tree, one has to start at the end nodes and move from right to left while computing nodes of all other nodes in the tree.

The value of an event node is equal to the weighted average of its outcomes with the probabilities of the outcomes used as the weights. In our sample tree, the value of the E2 node is computed as 0.60 \* 100 + 0.40 \* 1000 = 460.

The value of a decision node is equal to the maximum value of its alternatives. In the sample tree above, the value of the decision node is MAX(460,500) = 500.

Once we know the values of all nodes in the tree, we can follow the optimal decision path by choosing the alternative with the maximum value at each decision node. In the sample decision tree, we’d pick the second alternative.

When building a decision tree, one has to make a number of assumptions. The optimal path in the tree depends on the values that we assign to the end nodes and to the probabilities of the various outcomes.

We can test the sensitivity of the optimal path in the decision tree to the validity of these assumptions by changing some of the values and observing the corresponding changes in the optimal path.

One can observe, for instance, that the optimal path in the sample decision tree does not change as the probability of the likely outcome ranges from 0.6 to 1.0, which makes it fairly insensitive to the precision of our estimation of this probability.

In the real-life, decision trees can be much more complex than the one presented here. I wrote a simple Python script that can be used to render such trees as diagrams, compute values of the tree nodes, and produce an Excel spreadsheet with formulas assisting the sensitivity analysis.

The script can consume decision trees encoded in a simple [JSON format](https://www.json.org/).

Each node in the tree is presented as a collection of name/value pairs. These name/value pairs are called the properties of the node.

The end node has three properties: label, probability, and value. For example:

{ **"label"**: **"Likely outcome"**, **"probability"**: **0.6**, **"value"**: 100 }

An event node has four properties:

* type – the value of this property must be “event”
* label
* probability
* outcomes – an ordered list of nodes representing the possible outcomes

For example:

{  
 **"label"**: **"First alternative"**,  
 **"type"**: **"event"**,  
 **"outcomes"**: [{  
 **"label"**: **"Likely outcome"**,  
 **"probability"**: **0.6**,  
 **"value"**: 100  
 }, {  
 **"label"**: **"Unlikely outcome"**,  
 **"probability"**: **0.4**,  
 **"value"**: **1000** }]

}

Similarly, the decision node has four properties:

* type – the value of this property must be “decision”
* label
* probability
* alternatives – an ordered list of nodes representing possible alternatives

For example:

{  
 **"type"**: **"decision"**,  
 **"alternatives"**: [{  
 **"label"**: **"First alternative"**,  
 **"type"**: **"event"**,  
 **"outcomes"**: [{  
 **"label"**: **"Likely outcome"**,  
 **"probability"**: **0.6**,  
 **"value"**: 100  
 }, {  
 **"label"**: **"Unlikely outcome"**,  
 **"probability"**: **0.4**,  
 **"value"**: **1000** }]  
 }, {  
 **"label"**: **"Second alternative"**,  
 **"value"**: 500  
 }]  
}

The JSON representation of consists of two properties:

* parameters – an ordered list of parameters
* tree – the root node of the tree

Each parameter in the tree has three properties:

* name – the name of the parameter
* value – the numeric value of the parameter
* description – a description of the parameter

For example:

{  
 **"name"**: **"LIKELY\_OUTCOME\_PROBABILITY"**,  
 **"value"**: 0.6,  
 **"description"**: **"The probability of the likely outcome"**}

Parameters can be used in arithmetic expressions in the value properties of the various tree nodes.

Putting it all together, here is a complete representation of the simple tree that we considered earlier:

{  
 **"parameters"**: [{  
 **"name"**: **"LIKELY\_OUTCOME\_PROBABILITY"**,  
 **"value"**: 0.6,  
 **"description"**: **"The probability of the likely outcome"** }, {  
 **"name"**: **"UNLIKELY\_OUTCOME\_VALUE"**,  
 **"value"**: 1000,  
 **"description"**: **"The value of the unlikely outcome"** }],  
 **"tree"**: {  
 **"type"**: **"decision"**,  
 **"alternatives"**: [{  
 **"label"**: **"First alternative"**,  
 **"type"**: **"event"**,  
 **"outcomes"**: [{  
 **"label"**: **"Likely outcome"**,  
 **"probability"**: **"LIKELY\_OUTCOME\_PROBABILITY"**,  
 **"value"**: 100  
 }, {  
 **"label"**: **"Unlikely outcome"**,  
 **"probability"**: **"1-LIKELY\_OUTCOME\_PROBABILITY"**,  
 **"value"**: **"UNLIKELY\_OUTCOME\_VALUE"** }]  
 }, {  
 **"label"**: **"Second alternative"**,  
 **"value"**: 500  
 }]  
 }  
}

The Python script can be found on GitHub at <https://github.com/cherevik/DecisionTree>. The project page contains the instructions for installing and running the script.