Written Assignment #2:

To help with this assignment, I wrote several Python functions that use the Decision Tree algorithm. The two main functions are listed at the bottom of this section.

To start off, we have the following given:

Each row of the table is an example

examples = ({PlasticSurgery: FaceLift, TeethColor: Yellow, Manicure: No, Pedicure: No, IsMovieStar: No}, {PlasticSurgery: None, TeethColor: White, Manicure: Yes, Pedicure: Yes, IsMovieStar: Yes}, {PlasticSurgery: NoseJob, TeethColor: White, Manicure: No, Pedicure: Yes, IsMovieStar: Yes})

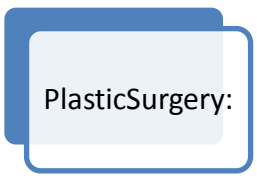
Each column is an attribute.

attributes = (PlasticSurgery, TeethColor, Manicure, Pedicure, IsMovieStar)

No parent examples yet.

parent_examples = ()

Our list of examples is not empty. Not all examples share the same classification, so we make a tree. The tree's root will have the most important attribute (argmax(attribute)) assigned to it. The Importance function returns the attribute that provides the most useful information. For the sake of simplicity, we will assume that the attributes are already ordered by importance.



After the root is created, new lists of examples are copied from the original list. For each new list, the most important attribute, PlasticSurgery, is the same across all examples in that list. Each list shares a different value for the PlasticSurgery attribute.

The new examples are:

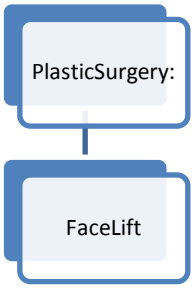
exs[0] = ({**PlasticSurgery: FaceLift**, TeethColor: Yellow, Manicure: No, Pedicure: No, IsMovieStar: No}, {**PlasticSurgery: FaceLift**, TeethColor: White, Manicure: Yes, Pedicure: Yes, IsMovieStar: Yes}, {**PlasticSurgery: FaceLift**, TeethColor: White, Manicure: No, Pedicure: Yes, IsMovieStar: Yes})

exs[1] = ({**PlasticSurgery: None**, TeethColor: Yellow, Manicure: No, Pedicure: No, IsMovieStar: No}, {**PlasticSurgery: None**, TeethColor: White, Manicure: Yes, Pedicure: Yes, IsMovieStar: Yes}, {**PlasticSurgery: None**, TeethColor: White, Manicure: No, Pedicure: Yes, IsMovieStar: Yes})

exs[2] = ({**PlasticSurgery: NoseJob**, TeethColor: Yellow, Manicure: No, Pedicure: No, IsMovieStar: No}, {**PlasticSurgery: NoseJob**, TeethColor: White, Manicure: Yes, Pedicure: Yes, IsMovieStar: Yes}, {**PlasticSurgery: NoseJob**, TeethColor: White, Manicure: No, Pedicure: Yes, IsMovieStar: Yes})

For each iteration, the original list of examples will be passed as "parent_examples".

Subtrees are created by depth, not breadth. For this particular next iteration, the function call will be "decision_tree_learning(exs, attributes - (PlasticSurgery,), examples)".



The variables for this new iteration will be:

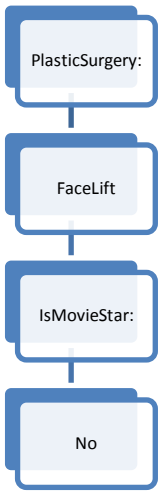
examples = ({PlasticSurgery: FaceLift, TeethColor: Yellow, Manicure: No, Pedicure: No, IsMovieStar: No}, {PlasticSurgery: FaceLift, TeethColor: White, Manicure: Yes, Pedicure: Yes, IsMovieStar: Yes}, {PlasticSurgery: FaceLift, TeethColor: White, Manicure: No, Pedicure: Yes, IsMovieStar: Yes})

attributes = (TeethColor, Manicure, Pedicure, IsMovieStar)

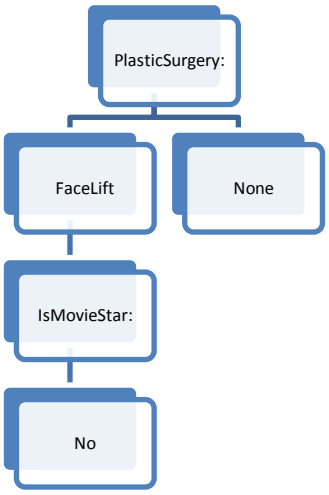
parent_examples = ({PlasticSurgery: FaceLift, TeethColor: Yellow, Manicure: No, Pedicure: No, IsMovieStar: No}, {PlasticSurgery: None, TeethColor: White, Manicure: Yes, Pedicure: Yes, IsMovieStar: Yes}, {PlasticSurgery: NoseJob, TeethColor: White, Manicure: No, Pedicure: Yes, IsMovieStar: Yes})

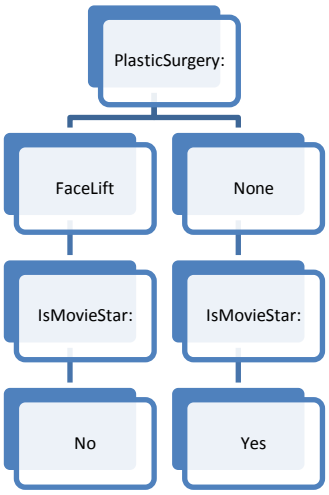
Here, there is only one example of each PlasticSurgery type. Therefore, all people with PlasticSurgery == FaceLift have yellow teech, no manicure, and no pedicure, et cetera. Therefore, all iterations for FaceLift examples will have the same list of examples, and the list of attributes will change as follows. Finally, this means that all (one) examples share the same classification, which means that the classification itself is returned. The classification FaceLift is then made as the leaf of the first major branch.

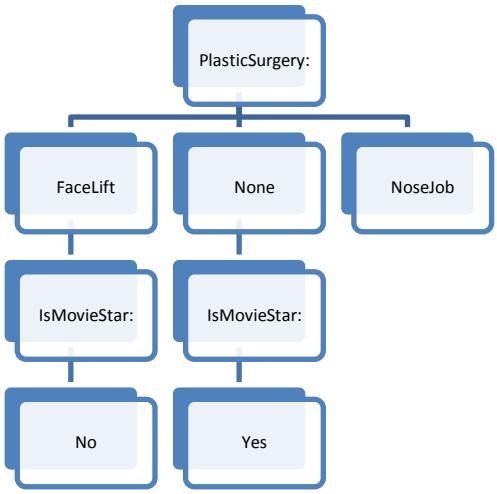
Since {IsMovieStar: No} will always be in the mapping while PlasticSurgery == FaceLift, the tree will look like this after its first completed branch.

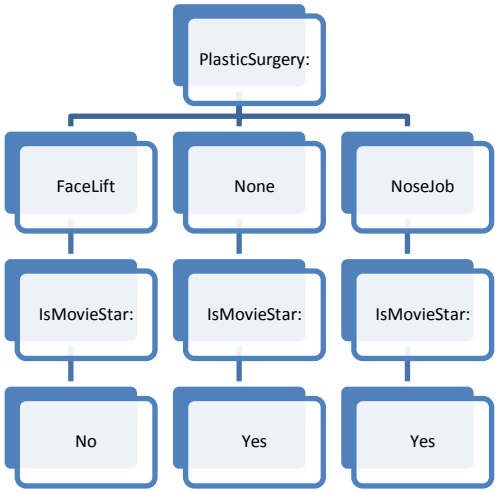


Again, there is exactly one example for any given type of PlasticSurgery. Repeating these steps, the classification will be return for the last two branches. The final tree will be:









Two of my Python functions.

```
def __build_decision_tree(examples, attributes):
    big_a = max_importance(attributes, examples)
    attributes_minus_big_a = tuple_without_e(attributes, big_a)
    tree = DecisionTree(big_a)
    for v_k in big_a.values:
        exs = {}
        for e in examples:
```

```
            exs[e] = v_k
        subtree = decision_tree_learning(exs, attributes_minus_big_a, examples)
        tree = tree.add_branch(subtree)
    return tree



def decision_tree_learning(examples, attributes, parent_examples):
    if not examples:
        return plurality_value(parent_examples)
    else:
        classification = __share_classification(examples)
        if classification is not None:
            return classification
        elif not attributes:
            return plurality_value(examples)
        else:
            return __build_decision_tree(examples, attributes)
```