决策树

注意：

决策树的训练与测试可以分开进行。

在训练中，通过训练数据集 `dataset` 构建整个树的结构，然后将树状的分支储存起来。

在测试中，直接使用训练好的决策树模型，对新的数据进行分类，在测试中不需要再使用 `dataset` 进行最优特征选择、分类等。

可以使用递归等方式进行决策树的

In [1]:

```
import numpy as np

# You can modify the code as what you want

class decision_tree:
    def __init__(self, features, output, dataset):
        self.features = features
        self.output = output
        self.dataset = dataset


    def log(self, x):
        return np.log2(x)

    # you can use this function to calculate the emprical probability of a random variable under a dataset
    def get_prob(self, array):
        (unique, counts) = np.unique(array, return_counts=True, axis=0)
        return counts/len(array)

    # you can use this function to calculate the emprical entropy of a random variable under a dataset
    def entropy(self, array):
        p = self.get_prob(array)
        return -np.sum(p*np.log2(p))

    def output_entropy(self):
        # calculate the emprical entropy of the output
        # you can use your code in the last assignment
        return

    def conditional_entropy(self, feature):
        # calculate the emprical conditional entropy of the output relative to the "feature"
        # you can use your code in the last assignment
        return

    def feature_selection(self):
        # select the feature has maximum mutual information
        # you can use your code in the last assignment
        return


    def predict(self, data):
        # make prediction for an arbitrary data input
        return




dataset = [
    {"age": 19, "male": False, "single": False, "visit_library_in_Sunday": False},
    {"age": 19, "male": False, "single": False, "visit_library_in_Sunday": False},
    {"age": 19, "male": True,  "single": False, "visit_library_in_Sunday": True},
    {"age": 19, "male": True,  "single": True,  "visit_library_in_Sunday": True},
    {"age": 19, "male": False, "single": False, "visit_library_in_Sunday": False},

    {"age": 20, "male": False, "single": False, "visit_library_in_Sunday": False},
    {"age": 20, "male": False, "single": False, "visit_library_in_Sunday": False},
    {"age": 20, "male": True,  "single": True,  "visit_library_in_Sunday": True},
    {"age": 20, "male": False, "single": True,  "visit_library_in_Sunday": True},
    {"age": 20, "male": False, "single": True,  "visit_library_in_Sunday": True},

    {"age": 21, "male": False, "single": True,  "visit_library_in_Sunday": True},
    {"age": 21, "male": False, "single": True,  "visit_library_in_Sunday": True},
    {"age": 21, "male": True,  "single": False, "visit_library_in_Sunday": True},
    {"age": 21, "male": True,  "single": False, "visit_library_in_Sunday": True},
    {"age": 21, "male": False, "single": False, "visit_library_in_Sunday": False},
    {"age": 21, "male": False, "single": False, "visit_library_in_Sunday": False},
    {"age": 21, "male": False, "single": False, "visit_library_in_Sunday": True}
]


my_tree = decision_tree(\
    ["age", "male", "single"], "visit_library_in_Sunday", dataset)


# Test 1
print(my_tree.predict({"age": 19, "male": False, "single": False}), "should be 0 or False")

# Test 2
# use the feature "single" to classify as last time
def tree_example(data):
    if data["single"]:
        return True
    else:
        return False

true_classification = 0

for data in dataset:
    if tree_example(data) == data["visit_library_in_Sunday"]:
        true_classification += 1

print("One feature classification accuracy:", true_classification/len(dataset))

# use a decision tree to classify
true_classification = 0

for data in dataset:
    if my_tree.predict(data) == data["visit_library_in_Sunday"]:
        true_classification += 1

print("Decision tree classification accuracy:", true_classification/len(dataset), "should be around 0.9412")
```

```
None should be 0 or False
One feature classification accuracy: 0.7647058823529411
Decision tree classification accuracy: 0.0 should be 0.9412
```

In [ ]: