HOMEWORK ASSIGNMENT 01: DECISION TREES

Ishan Joshi
USC ID- 34513341123
ijoshi@usc.edu

Contents

ID3 Decision Tree

The iterative deepening Decision Tree algorithm is a supervised learning algorithm that works on two main concepts – Information Gain and Recursion.

Information Gain:

Information gain is the measurement of the amount of reduction in uncertainty or entropy of the original output based on a feature of the given dataset. Entropy is mathematically defined as:

$$Entropy = -\sum_{i=1}^{n} P(i) * \log_2 P(i)$$

The Information Gain with respect to an attribute of the dataset is calculated by subtracting the entropy of the attribute from the original entropy of the output label.

$$I(\text{attribute}) = \text{Entropy}(\text{Original}) - \text{Entropy}(\text{attribute})$$

The objective of the algorithm is to maximise the Information Gain every time the data is split at a node. The repeated calculation and splitting of the tree based on values of Information Gain is achieved through recursion.

Algorithm:

1. Calculate the Information Gain for each of the attributes of the given dataset. Choose the attribute corresponding to the maximum value of Information Gain. This attribute is the root node.
2. Split the data based on the values of the root node.
3. Repeat steps 1 and 2 until you reach the leaf nodes of the tree.

At every step, the Information Gain calculations must be made on the subset of data obtained after splitting based on the values of the root node.

At the beginning of the program, the input file is read as a csv function using the pandas Dataframe structure.

Data Structures Used

1. **Dataframe**: A dataframe is a 2-dimensional data structure which is labelled. It consists of columns as labels and the corresponding values are stored in an SQL or spreadsheet format. This makes it easy to access data based on index values or column values. Pandas is a python library used to create data structures that make it easier to handle and access data for data analysis and machine learning algorithms. The data is read and converted to a csv file using pandas framework.
2. **Arrays:** An array is collection of values of the same data type stored in a contiguous location. Arrays are a fundamental data structure and are used at multiple places within the program.
3. **Dictionary:** A dictionary is an unordered changeable and indexed collection of data. It consists of key and value pairs where certain values are mapped to specific keys. The decision tree is displayed using a recursive dictionary where the root node is the starting key and each subsequent node is a nested dictionary.

Functions in code

1. **Entropy_Calc(Label_RV):** This function is used to calculate the entropy for a given value of an attribute. For eg, consider the attribute “Occupied”. The attribute has values “Low”, “Moderate” and “High”. We pass the values of the output label “Enjoy” where the value of “Occupied” is “Low” to the function as Label_RV. It returns the value of the entropy for the value stored in Label_RV.
2. **Info_Gain(data, split_attribute, target_value):** This function calculates and returns the Information Gain for a specific attribute within the dataset. The dataset is passed through the “data” parameter, the attribute in question through the “split_attribute” variable and the target_value is set to the output label. It returns the Information Gain.
3. **Tree_Build(features, sub_data, Input_File,target_value, parent_node):** This is the main function to build the tree. It accepts the attributes of the dataset through the “features” value. The “Input_File” and “target_value” parameters are set to the original dataset and the output label respectively. The parameters “sub_data” and “parent_node” are used to help in using the function recursively where “sub_data” uses the reduced dataset and “parent_node” is used to address the node that we need to further split the dataset on.
4. **Predict(query, tree):** This function is used to predict the output of the given query. It also calls itself recursively and accesses the tree and the query as dictionaries. It then checks if the node of the dictionary is in the node of the tree and traverses down the tree checking these conditions in the query until we reach a leaf node.

Challenges Faced

1. **Data Cleaning:** As an Electrical Engineering major, implementation of the algorithm, especially the data cleaning proved to be one of the biggest challenges of the assignment. I had to remove the string indexes present in the first attribute values, the semicolon at the end of the values in the output label as well as the commas and the brackets present in the input text file. All of this was done using pandas and converting the text file into a csv dataframe, which made future accessing of the data easier.
2. **Recursion:** Another major challenge was making sure that the recursion was working perfectly. To implement that I needed to define conditions which would determine if a leaf node was reached, and the recursion should be stopped. To rectify this issue, I imposed three limitations to be checked before making a recursive call:
    a. If all values of the output label are equal or the same.
    b. If the dataset is empty, return the last accessed feature in the traversal from the original dataset.
    c. If the feature space is empty, return the attribute present in parent node, which is the last attribute that accessed the function in a recursive call.
3. **Data Structures:** Data handling using dataframes and building the tree using nested dictionaries proved to be challenging at times, especially during recursive calling as the data set would have to be reduced at each call by removing the maximum information gain attribute from the previous run. The prediction also provided a key error initially.

Outputs

The tree is displayed as follows:

{'Occupied': {'High': {'Location': {'City-Center': 'Yes',

'German-Colony': 'No',

'Mahane-Yehuda': 'Yes',

'Talpiot': 'No'}},

'Low': {'Location': {'City-Center': {'Price': {'Cheap': 'No',

'Normal': {'Music': {'Quiet': {'VIP': {'No': {'Favorite Beer': {'No': 'No'}}}}}}}},

'Ein-Karem': {'Price': {'Cheap': 'Yes',

'Normal': 'No'}},

'Mahane-Yehuda': 'No',

```
                    'Talpiot': 'No'}},
        'Moderate': {'Location': {'City-Center': 'Yes',
                                  'Ein-Karem': 'Yes',
                                  'German-Colony': {'VIP': {'No': 'No',
                                                            'Yes': 'Yes'}},
                                  'Mahane-Yehuda': 'Yes',
                                  'Talpiot': {'Price': {'Cheap': 'No',
                                                        'Normal': 'Yes'}}}}}}
```

The input query is:

{‘Occupied’: Moderate, ‘Location’: City-Centre, ‘Price’: Cheap, ‘Music’: Loud, ‘VIP’:No, ‘Favorite Beer’: No}

Output:

The prediction for the given query is:

'Yes'

Applications of Decision Trees

1. **Engineering:** Operations such as energy consumption and fault diagnosis require decision trees. Energy consumption involves calculating the amount of energy consumed by individuals. Decision trees are the most preferred method of calculating energy consumption as they are an optimal method to display the depth and insight of the information.

   Identification of faulty bearing in rotary machinery is done using Decision trees.
2. **Healthcare Management:** Decision trees can be used to predict delayed cognitive, motor or language development in children early enough to provide appropriate treatment and help them in their development and growth.
3. **Business Management:** Corporations maintain large databases of their customers to optimize service. Decision trees are useful to locate relevant variables for a operation.