INF552 HW1: Decision Trees

Author: Vivek Bharadwaj <vivekb@usc.edu>

Part-1: Implementation

Data Formats:

- For sake of simplicity, train & test data are converted to csv format with the header row.
- The provided file *dt-data.txt* has been converted to a csv file - trainData.csv
- The test data for this experiment consists of the original training data and 1 addition column presented in the question.
  - This is contained in *testData.txt*
- The test data will contain only the feature columns ie. not the target *Enjoy* column.
- The predictions made by the algorithm is written to another file. It is formatted as a pythonic list of tuples.
  - [(idx-of-test-data, prediction)]

```
[(0, 'No'),
 (1, 'Yes'),
 (2, 'Yes'),
 (3, 'No'),
 (4, 'Yes'),
 (5, 'Yes'),
 (6, 'No'),
 (7, 'Yes'),
 (8, 'Yes'),
 (9, 'No'),
 (10, 'No'),
 (11, 'No'),
 (12, 'Yes'),
 (13, 'Yes'),
 (14, 'Yes'),
 (15, 'No'),
 (16, 'Yes'),
 (17, 'No'),
 (18, 'No'),
 (19, 'Yes'),
 (20, 'No'),
 (21, 'Yes'),
 (22, 'Yes')]
```

- The format of the generated decision tree is same as the prescribed in the question.

attribute A on the 1st level
 first value of attribute A: 1st attribute B_1 on the 2nd level
  first value of attribute B_1: 2st attribute B_1 on the 3nd level
  …
   n$^{th\ value}$ of attribute X: yes
 second value of attribute A: 2nd attribute B_2 on the 2nd level
  first value of attribute B_2: 2st attribute B_2 on the 3nd level

Software Used:

- The vanilla implementation is written in Python v3.6.4.
- The dataset has been modeled after Pandas DataFrame.
- A DataSet class is written as a wrapper around the DataFrame. I switched the implementation from python lists/dicts to Pandas. But, I did not want to change all calls, so, a used is wrapper.

Files Included:

- DecisionTree.py - Contains the vanilla implementation and the driver program.
- RunDecisionTree.sh - Shell script to run the vanilla implementation with the train, test. data files and passes filenames for outputs (tree structure and the predictions).
- trainData.csv - Training data in csv format.
- testData.csv - Test data in csv format.
- DecisionTree_sklearn.py - Contains the decision tree implemented with Scikit-Learn and the driver program.
- RunSkLearn.sh - Shell script to run the Scikit-Learn implementation.

Files Generated:

- tree.txt - Tree structure in the prescribed format.
- prediction.txt - prediction of the vanilla implementation on testData.csv.
- prediction.sklearn.txt - prediction of the Scikit-Learn implementation on testData.csv.

Code Organization:

- As mentioned, there is a class *DataSet* which wraps the Pandas DataFrame object and provides methods to perform ops: partition, entropy, unique attribute values, etc.
- The *DecisionTree* class models the tree. It exposes a method to build it from a *DataSet*. It assumes that the last column is the target/label column.
    - It also contains a method *predict* which predicts a *DataSet*'s label by walking the tree. The argument can contain multiple rows, it predicts results for each row.
- The DecisionTree itself is made of *Node* objects. There are of 2 types:
    - *DecisionNode*: A decision node in the tree - not leaf. It contains the attribute to test and the child nodes for each value of the host attribute.
    - LeafNode: A leaf node. It contains the target attribute and its value for the leaf. The walk of the tree terminates at the leaf.
- Further details of methods is listed in the source code documentation.

Part-2: Software Familiarization

Software Used:

- *Scikit-Learn* is used as a library implementation of decision-trees.
- Data has been modeled using Pandas DataFrame as in the vanilla implementation.

Comparison:

- First striking point about Scikit-Learn is that it does NOT support categorical data as the one provided.
  - Categorical is when all the features have discrete values. Can be strings or integers.
  - We must convert the categorical data into binary values using one-hot encoding.
  - If *WIND* has attributes *LOW* and *HIGH*, after one-hot encoding, we explode *WIND* into 2 columns, *WIND_LOW*, *WIND_HIGH*. These can have 0, 1 as it's values. 0 indicates it is off, 1 indicates it is on.
- The default criterion for calculating information-gain in Scikit-Learn is *gini* method. The vanilla implementation uses *entropy* as the criterion. Scikit-Learn does have an option to use *entropy*.

Ideas & Suggestions for Code:

- Scikit-Learn has a method to export the tree to a *dot* file which can be visualized easily using *GraphViz*.
- Leaf nodes are created only when the dataset under consideration is pure (entropy=1).
  - This might result in a *very deep* tree structure which can *overfit* the training data.
- At the decision nodes, we can consider a combination of features instead of only 1 feature.
  - If the dataset contains a large no. of features, it could improve interpretability of the tree and a possibly simpler hypothesis.
- Ability to handle continuous features while still being classification ie. not regression.
  - Handling continuous data by breaking into discrete sub-intervals.
- If any combination of the features are not a valid walk in the generated tree, it reports that it cannot make a prediction.
  - Instead, it can make a random guess among the known labels.
- Train different trees using cross-validation of the training data and picking the best tree.

Part-3: Applications

The biggest advantage of decision-trees is the simplicity of the hypothesis and the interpretability of the tree structure.

- Decision trees are used to great effect in fraud detections in financial transactions.
    - The knowledge base of any bank is huge - there are millions of transactions happening per day.
    - There are multiple features that can capture a fraudulent transaction with a certain degree of certainty.
    - Even if we have a false positive (falsely marked fraudulent), it is very easy with to interpret the reason why the system predicted this.
- Decision trees are increasingly used to make sense of the data rather than prediction itself.
    - The information gain strategy helps uncover the most significant features that influence the data.
    - This knowledge can be applied to more complex ML models.