Overview of approach:

The attributes provided by the sample data set from Figure 7.1 were able to be represented as a suitable categorical decision tree learning problem. The four input features and target feature with differentiate of true or false (skips or read); Author, thread, length, Where_Read, and User_actions respectively, were these attributes. The User_actions attribute specifically had a suitable feature-value pair to be observed as per the assignment. This representation supported a straightforward split defined by the attribute values and made the learning model able to be used/interpreted by the AIPython files for learning tree implementation. This representation was directly supported by the current learning tree learning model builder in AIPython, one-hot encoding was not necessary as AIPython in place decision tree learning already managed symbolic branching.

The data set contained eighteen examples (e 1 - 18) towards training and two test examples (e19 & e20) all of which adhering to the aforementioned attributes with varied data. That data set was represented within the file Code.py and loaded to the method import of Data_set, this preserved the structure and passed it to the decision tree learner within learnerProblem.py in the AIPython folder. That decision tree was trained with the import method DT_learner with the child weight set to 1 and gamma set to 0.0 (min_child_weight = 1, gamma = 0.0) to prevent empty leaves and disable pruning probability, respectively. Using these configurations enabled the tree to capture the structure of Figure 7.1's relatively small dataset without removing branches which would have caused generalization and lack of distinct decisions on the path.

Tree visualization:

In context of the approach taken to achieve this assignment, using AIPython's currently implemented routes of data set retrieval, tree learning, and decision output. The visualization of the learning path was a direct output of the implementation already in place from AIPython, the output specifically being the decision tree's predictors displayed on a documented string. The string is displayed in the terminal to the user and exhibits the root attribute chosen by the information retrieved and the rest of the branches in order of decreased relevance of the learning model. Each node represents a test of one of the four attributes, and each leaf assigned is a class label based on the majority class of examples matching the branch. An example can be seen here:

```
jh1163@cell07-cse:~/IntroAI/PA5$ python3 CodePart.py
Training set has 17 examples. Number of columns:  {5}
Test set has 2 examples. Number of columns:  {5}
Validation set has 1 examples. Number of columns:  {5}
There are 4 input features
Summary of Given Dataset:


Data: 17 training, 1 validation{len(self.test)} test examples; {len(self.train[0])} features.

Learning Decision Tree:


(if Length==short then (if Thread==followup then (if Author==unknown then {'reads': 0.0, 'skips': 1.0} else {'reads': 1.0, 'skips': 0.0}) else {'reads': 1.0, 'skips': 0.0}) else {'reads': 0.0, 'skips': 1.0})

Predictions on Given Test Example Theta 19 and 20 Respectively:

Theta 19: features = ('unknown', 'new', 'long', 'work'), predicted = {'reads': 0.0, 'skips': 1.0}, true = reads
Theta 20: features = ('unknown', 'followup', 'short', 'home'), predicted = {'reads': 0.0, 'skips': 1.0}, true = skips
jh1163@cell07-cse:~/IntroAI/PA5$
```

Predictions for e_{19} and e_{20}:

The test examples e19 and e20 were given to the predictor and evaluated against the learned decision paths. Their user-action labels were given by the assignment, e19 = reads and e20 = skips. For e19, the feature pattern influenced the predictor to enter the branch where length did not equal short, producing “skips” for the prediction, which did not match the given test of “reads”. For e20, the attributes were akin to the nested conditions of Length being short, Thread being follow up, and Author being unknown, which when satisfied are predicted to have user-action to be a ”skips” leaf, which aligned with the given assignment label. The outcomes present an exact application of predictions from the learned model from the Figure 7.1 Dataset.

Results and Limitations:

After the learning model was applied to the dataset, the resulting decision tree was completely interpretable. The predictions coincided with the explicit attribute sequence, leading to the belief that the model’s justification is complete and visible. Feature importance over themselves was implicitly determined by the order of the tests in the tree, with attributes exhibiting near the near the root exerting the most influence on the final classification. The tree’s structure directly reflected the categorical patterns present in the training cases, so the reasoning behind each prediction remained visible and understandable.

The biggest limitation was the size of the dataset, following the coverage. Decision trees, like probability in statistics, grow more accurate and discernable with an increased example pool/sample size. The dataset from 7.1 only had eighteen certified examples with filled attribute lists used on the learning model for predictions. These small amount of examples along with amount of varied attribute to value combinations restricted the models ability to form stable splits. Multiple branches were supported by only one or two instances within the data set which made the final rules easily affected to individual cases/examples of user behavior. The decision to disable pruning allowed the tree to capture all the narrowed patterns within the data set directly. Which represented the dataset reproduction more exactly but gives the risk of overfitting into the resulting tree structure, which can lead to errors with certain examples derived from attribute combinations in the original eighteen cases that influenced the model. The model is still readable and applicable but the generalization