CSC 635 Data Mining

Assignment 2 Report

Submitted to:

Dr. Jamil Saquer

Author:

*Sharmin Sultana*

**Introduction**

For the part one of this assignment, a training dataset was given about the hiring status (Yes or No) of fourteen (14) candidates after an interview. Each record has 5 attributes. These attributes are ‘level’, ‘lang’, ‘tweets’, ‘phd’ and ‘hire’. The ‘level’ attribute represents the position (Junior, Mid and Senior) for which each candidate attends the interview, and the ‘lang’ attribute tells us on which programming language each interviewee was proficient. The ‘tweets’ and ‘phd’ are categorical attributes that describe whether a candidate is active on tweeter and has phd degree. Based on these attributes the target attribute ‘hire’ tells us whether a candidate was hired by the company or not. These attributes are used to build a decision tree by using ID3 algorithm and missing cases are also handled as well. I also tested the algorithm’s accuracy with few test cases.

In part two of the assignment, I used the popular dataset of playing tennis [1]. This dataset is about whether tennis will be played or not on a certain day based on few attributes. These attributes are ‘day’, ‘outlook’, ‘temp’, ‘humidity’, ‘wind’ and ‘play’. The ‘day’ attribute enumerates the days from 1 to 14. The ‘outlook’ tells how the weather looked like on a particular day and the ‘temp’ column measures the hotness or coldness or mildness on that day. The ‘humidity’ attribute describes the presence of vapor in the air and ranks it as High and Normal. The ‘wind’ attribute demonstrates whether the wind is strong or weak. Based on these features, the target attribute ‘play’ defines whether tennis was played or not on that day. This dataset has been used to construct decision tree by using ID3 algorithm and tested the efficiency with sample cases.

**Background**

For both part of the given assignment, I have constructed the decision tree by using the ID3 algorithm that works in a top-down recursive manner. To make the branch, I choose attributes based on Information Gain of each attribute, and entropy is also needed to be calculated with respect to the target attribute.

The whole procedure is explained below:

1. At first, I calculate the entropy of the target attributes (‘hire’ and ‘play’) for both dataset by using formula (I).
2. Then also calculated the entropy of the feature attributes with respect to the target variables.
3. After entropy calculation, information gain of each attribute has been calculated with respect to the dataset entropy by using formula (II).
4. In the next step, ID3 algorithm was implemented where root is selected based on the highest information tree and in recursive manner subtrees are built
5. Finally, we tested few samples to be ensure that ID3 algorithm is working properly.

**Implementation**

1. def **entropy():** this function which takes the probability as parameter calculates the entropy of target variables along with feature attributes.
2. To calculate entropy, I need to find out the probability of each attribute based on their class level. For this I used the function **probability\_cal ()** that takes two arguments: one is ls that maps each attribute and another is their corresponding values.
3. def **information\_gain():** this is used to calculate the information gain of each attribute. At first, I grouped the attributes based on their values by using groupby and then stored these in glist. After that I measure the proportion of each value with respect to the target attributes and saved it in new\_entropy which is compared with default attribute entropy value stored in old\_attribute and their difference is return as the information fain.
4. def **id3 ():** This function is used to build the tree based on ID3 algorithm. For this I considered 3 cases:

* If all the samples in the datasets belong to the same class C, then return np.unique(df[target\_attribute])[0] i.e. a leaf node labeled as C.
* If the dataset is empty, then leaf node is labeled by the most common class.
* If attributes are empty, then returns the default class.
* If the above cases are not true, then calculates the information gain of each attribute and keeps the gains in an array. From this array, I choose the attribute as root that has highest gain and follow this step recursively to build the subtree.
* Missing values are also handled here by assigning the most common target values to the unlisted attribute values. i.e. *tree[best\_attr][None] = cnt.most\_common(1)[0][0].*

1. def **classify** (): it takes the generated tree and sample case as parameter and checks each attribute and results are generated by matching the attribute keys of the sample with the tree.

For part one of the assignment, firstly I loaded the dictionary of given dataset into a .csv file and saved it in ‘hire.csv’. The second dataset is loaded into testing\_dataset. For both datasets, I followed the procedure from 1 to 5 to build the tree and classify the test results.

**Experimental Results**

In part 1 after constructing the decision tree by considering unexpected and missing values, the output looks like this:

Text

Description automatically generated

*Figure 1: Decision Tree of given dataset*

Few samples are also passed through the classify function to check that the implemented decision tree is working perfectly. The output of the sample input are as follows:

Text

Description automatically generated

*Figure 2: Sample output for the sample input for given dataset*

Since, hiring status ‘True’ is dominant over ‘False’, that’s why True will be the outcome when an unexpected value Intern is given. Similarly for the value Senior, the dominant status is ‘Do not hire’ that’s why False will be the result.

In part 2, the attribute ‘day’ seems irrelevant for decision making so I dropped the column by using drop function. Now if we analyze the target attribute values then it is visible that ‘Play’ is dominant over ‘Do not Play’. That’s why when an unexpected input is given, output will be ‘Yes’. Again, for the Sunny outlook, dominant condition is ‘No’, that’s why we will receive ‘Do not Play’ for this. Constructed decision tree with none cases and output of the sample cases are given below for the dataset ‘play\_tennis.csv’.

A screenshot of a computer

Description automatically generated with medium confidence

*Figure 3: Decision Tree and sample output of play\_tennis dataset*

Finally, the accuracy of the second dataset has been calculated by splitting it. For the training sample 10 information were considered and rest information are partitioned as testing cases. After splitting, call the prediction function to match the output with the desired output. Here, because of the smaller dataset, the accuracy comes out 100%, if we consider large dataset then the accuracy will be slightly less.

**Conclusion**

In conclusion, this assignment helps me to learn implementation of decision tree by using ID3 algorithm in python, and also to take care of the unexpected attribute values in the dataset.

**References**

[1] “Play Tennis.” *Kaggle.com*, [www.kaggle.com/fredericobreno/play-tennis](http://www.kaggle.com/fredericobreno/play-tennis)

‌