Second Practice for Machine Learning:Details about Codes

[Teacher : Dr. Ahmad Ali Abin] [Name : Hossein Simchi]
Student Number : 98443119
Computer Science and Engineering , Shahid Beheshti University of Tehran , Iran
[h.simchi@mail.sbu.ac.ir]

May 29, 2020

1 Introduction

In the second exercise, the ID5 algorithm is implemented, which we will discuss in detail .

2 Load Dataset

We first try to read the dataset using the Pandas Library. To make it easier to

```
book = pd.read_excel('C:\\Users\\Lenovo\\Desktop\\data.xls')
```

Figure 1: Load dataset with pandas

continue, we try to convert the column of the labels to numbers, to do this we convert the phrase *yes* to 1 and *no* to 0.

```
book=book.replace({'Label': r'no'}, {'Label': 0}, regex=True)
book=book.replace({'Label': r'yes'}, {'Label': 1}, regex=True)
```

Figure 2: convert Label with pandas

3 ID5 Algorithm

ID5 algorithm works in an incremental way. In this way, by entering each data, we first calculate the entropy of each feature and place the feature with the least entropy at the root and then we do these steps again with each new data.
To make sure our algorithm selects the best feature for the initial nodes, we use the *pull up algorithm*, which selects the feature with the most $information$ *gain* each time and puts it at the top of the tree.

4 Details in code

The following steps are all written in a for loop, in which case each time the for loop is repeated, it reads a row of data that means new data and performs the explanations described in the previous section.
First, we should select the desired number of rows using the following command.

```
book3 = book.iloc[0:i][:]
```

4.1 Feature entropy function

The input of this function is the name of the feature and all kinds of types of this feature. It then calculates how many of each type has a positive label and how many has a negative label. We have to repeat this for each type, to determine how many of them are positive and how many are negative.
Once it has been determined whether any of these types are negative or positive, we should try to calculate the entropy belonging to each feature in the next step. To do this, it is sufficient to add all the entropy values for each type together.
After determining the amount of entropy for each feature, *information gain*

```
def Features_entropy (feature_name,feature_type,feature_type1,feature_type2,feature_type3):
```

can be calculated.

4.2 count function

It does almost the same thing as the previous function, except that it doesn't calculate the *entropy* for each feature.

```
def count(feature_name,attr1,attr2,attr3,attr4):
```

4.3 attribute function

The input of this function is the name of the selected feature (the feature with the highest *information gain*) and its output returns all types of the feature.

4.4 positive(pos) negative(neg) function

We use this function to draw a tree. When we can detect the label of any data with all types of that feature, this function draws a positive or negative node, which is actually the final node.

```
def attribute(Feature_name):
```

```
def pos_neg(t,attr1,attr2,attr3,attr4):
```

4.5 Anytree Library

Using anytree library, you can draw all the selected nodes.
One of the functions in this library is the $Node\ function$, which has two inputs, one called the node that is to be placed and the other the parent of that node. Obviously, the node at the root does not have a parent, so its value can be set to None.
Another function that is used is the $rendertree\ function$, the input of which is the root node and draws the tree from the root in a chain.

4.6 precision function

To calculate precision of this algorithm on test set, we used the standard precision formula ($\frac{TP}{TP+FP}$). The way it works is that it calculates the ratio of the

```
def precision():
```

number of positive and negative labels in the training set. If the ratio of positive labels is higher, it assigns a positive label, otherwise it allocates a negative label to our data.