[Data Science] 2. Decision Tree

Environment & Goal

- *OS* : Windows
- *Language* : Python
- *Goal* : Build a Decision Tree, and then classify the test set using it

Summary of algorithm

- 전체적인 algorithm flow는 다음과 같다.
    1. Given Data를 통해, *Top-down recursive divide-and-conquer greedy algorithm*을 통한 Model Construction ~ **def model_construction(self, attr, value, sample)**
    2. 이후, 각 Test sample의 class label을 prediction ~ **def prediction(self, path)**

Detail of Design

- *class DecisionTree*
    - *class Node*
        - Decision Tree의 각 Node를 구성하는 class로, {attr, value}가 하나의 prediction이 되고, 그에 해당하는 sample만을 소유한다.
        - label의 값이 존재할 경우에만 Leaf node로 간주

```
class Node:
        def __init__(self, attr, value, sample):
            self.attr = attr
            self.value = value
            self.label = None
            self.sample = pd.DataFrame(sample)
            self.child = []
```

- *def load_data(path)*
    - path를 통해 dt_train.txt를 한줄 씩 읽어 dataframe에 저장

```
@staticmethod
    def load_data(path):
        data = []
        input_ = open(path, 'r')
        attrs = input_.readline().split()
        while True:
            data_ = input_.readline()
            if not data_:
                break
            data_ = data_.split()
            data.append(data_)
        input_.close()

        return pd.DataFrame(data, columns=attrs)
```

- ***def model_construction(self, attr, value, sample)***
    - Tree Construction의 Stopping Condition
        - *if sample.shape[0] == 0* : 더 이상 sample이 남지 않았을 때, None을 return
        - *elif sample[sample.columns[-1]].unique().shape[0] == 1* : 현재 partition의 class label이 하나만 존재할 경우, Leaf node를 만들어 반환
        - *elif sample.columns.shape[0] == 1* : 현재 partition에 남아있는 attribute가 존재하지 않을 때, majority vote로 class label을 부여한 Leaf node를 반환
    - Stopping condition에 해당하지 않을 경우
        - Attribute selection measure를 통해 attribute를 고른다.
        - attribute를 통해 parent sample을 partition한 후, 해당 attribute column을 삭제하여 child node에 대한 model_construction 호출
        - child node construction이 완료되면, parent node를 반환하여 전체 Tree를 Construct

```
def model_construction(self, attr, value, sample):
        if sample.shape[0] == 0:
            return None
        elif sample[sample.columns[-1]].unique().shape[0] == 1:
            node = self.Node(attr, value, sample)
            node.label = sample[sample.columns[-1]].unique()[0]
            return node
        elif sample.columns.shape[0] == 1:
            count = sample.groupby(sample.columns[-1]).count()
            label = count.idxmax()
            node = self.Node(attr, value, sample)
            node.label = label
            return node
        else:
            selected = Measures.gain_ratio(sample)
            subset = sample.groupby(selected)
            values = sample[selected].unique()
            node = self.Node(attr, value, sample)
            li = []

            for value_ in values:
                partition = subset.get_group(value_)
                partition = partition.drop([selected], axis=1)
                child = self.model_construction(selected, value_, partition)
                if child is not None:
                    li.append(child)

            node.child = li
            return node
```

- ***def prediction(self, path)***
    - 각 test sample에 대해 prediction 진행
    - Tree Traversal 도중, 특정 attribute에 대한 값이 Tree에는 존재하지 않는 경우가 있다.
    - 이때, parent node인 cur의 sample에서 majority vote를 통해 test sample의 class label을 결정

```
def prediction(self, path):
        data = []
        input_ = open(path, 'r')
        attrs = input_.readline().split()
        while True:
            data_ = input_.readline()
            if not data_:
                break
            data_ = data_.split()
            data.append(data_)
        input_.close()

        data = pd.DataFrame(data, columns=attrs)
        data[self.data.columns[-1]] = None

        for sample in data.itertuples():
            cur = self.root
            while cur.label is None:
                attr = cur.child[0].attr
                for child in cur.child:
                    if child.value == sample[self.data.columns.get_loc(attr) + 1]:
                        cur = child
                        break

                if child.value != sample[self.data.columns.get_loc(attr) + 1]:
                    parent = cur.sample[cur.sample.columns[-1]].value_counts()
                    cur.label = parent.idxmax()
                    break

            data.at[sample[0], self.data.columns[-1]] = cur.label
        return data
```

- *def write(self, result, path)*
    - prediction의 결과를 File의 양식에 맞게 작

```
def write(self, result, path):
        input_ = open(path, 'w')
        line_ = ""
        for i in self.data.columns:
            line_ += str(i) + "\t"
        input_.write(line_ + "\n")

        for line in result.itertuples():
            line_ = ""
            for i in line[1:]:
                line_ += str(i)
                if i is not line[-1]:
                    line_ += "\t"

            input_.write(line_ + "\n")

        input_.close()
```

- *class Measures*
    - ***def info_gain(data)***
        - $Gain(A) = Info(D) - Info_A(D)$
        - Info(D)의 값은 모두 동일하므로, Info_A(D)의 값이 가장 작은 Attribute를 select

```
@staticmethod
def info_gain(data):
        attrs = data.columns
        min_gain = 1
        min_attr = attrs[0]
        # pick one attribute and partition data based on attr types

        for attr in attrs[:-1]:
            subset = data.groupby(attr)
            values = data[attr].unique()
            info = 0

            # compute Info(D_a)
            for value in values:
                partition = subset.get_group(value)
                li = partition[attrs[-1]].value_counts()
                li = li / partition.shape[0]
                info += -np.sum(li * np.log2(li)) * partition.shape[0] / data.shape[0]

            # search minimum info_gain
            if min_gain >= info:
                min_gain = info
                min_attr = attr

        return min_attr
```

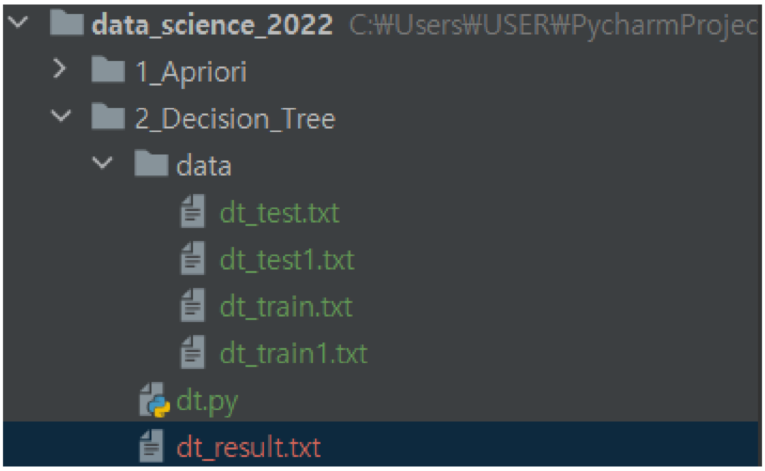
    - ***def gain_ratio(data)***
        - $GainRatio(A) = Gain(A)\ /\ SplitInfo(A)$

    - Info_gain에서 splitinfo(A)를 구하는 코드를 추가

```
@staticmethod
    def gain_ratio(data):
        attrs = data.columns
        max_ratio = -1
        max_attr = attrs[0]

        before = data[attrs[-1]].value_counts() / data.shape[0]
        before = -np.sum(before * np.log2(before))

        # pick one attribute and partition data based on attr types
        for attr in attrs[:-1]:
            subset = data.groupby(attr)
            values = data[attr].unique()
            ratio = 0
            split_info = 0

            # compute Info(D_a)
            for value in values:
                partition = subset.get_group(value)
                p = partition.shape[0] / data.shape[0]
                split_info += -p * np.log2(p)
                li = partition[attrs[-1]].value_counts()
                li = li / partition.shape[0]
                ratio += -np.sum(li * np.log2(li)) * p

            ratio = (before - ratio) / split_info

            # search maximum gain_ratio
            if max_ratio <= ratio:
                max_ratio = ratio
                max_attr = attr

        return max_attr
```

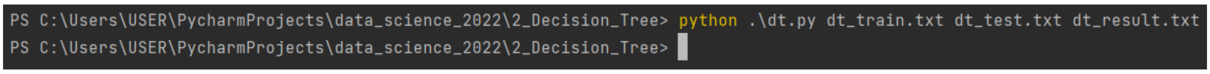
Test & Result

- 현재 경로는 다음과 같이 설정했다
- main function 내에 추가 경로를 붙여주었다

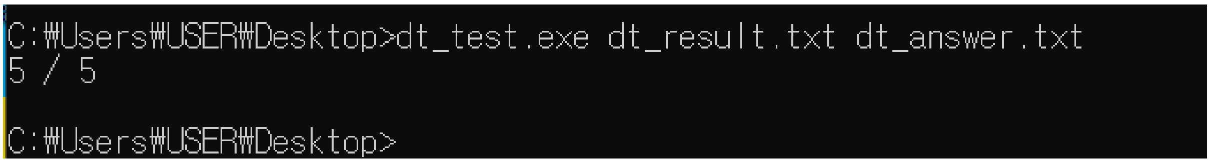


```
data_science_2022  C:₩Users₩USER₩PycharmProjec
  1_Apriori
  2_Decision_Tree
    data
      dt_test.txt
      dt_test1.txt
      dt_train.txt
      dt_train1.txt
    dt.py
    dt_result.txt
```

1) dt_train.txt



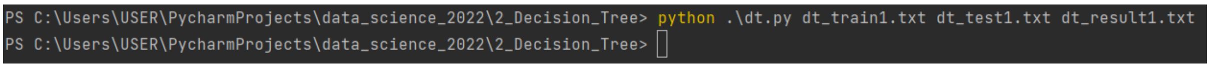
```
PS C:\Users\USER\PycharmProjects\data_science_2022\2_Decision_Tree> python .\dt.py dt_train.txt dt_test.txt dt_result.txt
PS C:\Users\USER\PycharmProjects\data_science_2022\2_Decision_Tree>
```



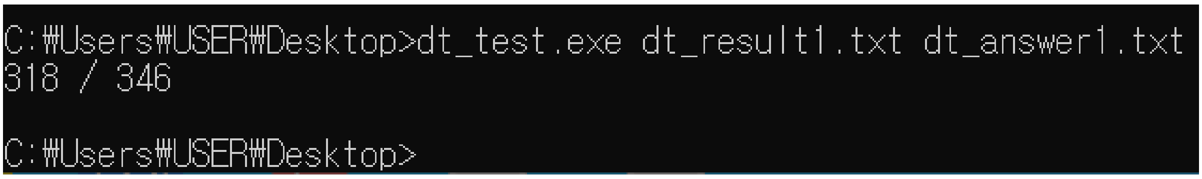
```
C:₩Users₩USER₩Desktop>dt_test.exe dt_result.txt dt_answer.txt
5 / 5

C:₩Users₩USER₩Desktop>
```

- information gain과 gain ratio 모두 같은 결과를 나타내었다.

2) dt_train1.txt



```
PS C:\Users\USER\PycharmProjects\data_science_2022\2_Decision_Tree> python .\dt.py dt_train1.txt dt_test1.txt dt_result1.txt
PS C:\Users\USER\PycharmProjects\data_science_2022\2_Decision_Tree>
```



```
C:₩Users₩USER₩Desktop>dt_test.exe dt_result1.txt dt_answer1.txt
318 / 346

C:₩Users₩USER₩Desktop>
```

- information gain과 gain ratio 모두 같은 결과를 나타내었다.