Decision Tree

(Assignment #2)

2016025678 이강민

1. Summary of your algorithm

Decision tree는 다양한 항목에 대한 데이터를 트리로 구현하여 의사 결정에 사용할 수 있는 모델입니다. 트리 노드를 통해 leaf 노드에 도달할 때까지 공통 항목에 대해 데이터를 분석하고 결과에 따라 데이터를 분류할 수 있습니다.

제가 구현한 Decision tree는 주어진 train 데이터를 기반으로 decision tree를 구성한 후에 제시된 test data에 class label을 예측하여 output 파일로 만들어주는 구조를 갖고 있습니다.

2. Detailed description of your codes

1) Main

```
if __name__ == "__main__":
    df = pd.read_table(sys.argv[1],sep='\t')
    test = pd.read_table(sys.argv[2],sep='\t')
    tree = Dtree(df)
    test.loc[:,df.columns[-1]] = None
    temp = test.columns[-1]
    for i in range(0,len(test)):
        test.loc[i,temp] = tree.classify(test.loc[i])
    test.to_csv(sys.argv[3],sep='\t',index = False)
```

- Main에서는 train data, test data를 받고 결과를 output 파일에 작성하는 역할을 합니다. pandas의 dataframe을 사용하여 data를 read하고 write하였습니다.

2) Dtree class

```
class Dtree():
    def __init__(self,df,is_leaf=0):
        self.df = df
        self.attri = None
        self.label = None
        self.child = None
        self.is_leaf = is_leaf

        self.make_tree()
        self.label = self.df.iloc[:,-1].value_counts().idxmax()
```

- 트리를 구성하기 위해 class를 만들었습니다. 각 노드의 구성요소로는 attribute와 child, leaf인지 판단하는 is_leaf, child를 갖고 있습니다. 또한, 트리 생성시 자동으로 트리를 만들게 되어 있습니다. class label의 경우 다수결의 원칙을 사용하고 있습니다.

3) make_tree 함수 (class 내장 함수)

```
def make_tree(self):
    if len(self.df.iloc[:,-1].unique()) == 1:
        self.is_leaf = 1;

    else:
        att = list(self.df.columns[:-1])
        min_g = float('inf')
        for i in att:
            gain = info_gain(self.df,i)
            if gain <= min_g:
                min_g = gain
                self.attri = i
        self.child = dict()
        for i in self.df[self.attri].unique():
            new_input = self.df.loc[self.df[self.attri]==i]
            self.child[i] = Dtree(new_input.drop(self.attri,axis=1))
```

- make_tree 함수는 class 내장 함수로 class 생성 시 실행되는 함수입니다. 먼저 만들어진 노드의 class label이 하나로 정해지면 leaf 노드로 설정합니다. leaf 노드가 아닌 경우 attribute를 infomation gain값을 기준으로 선정합니다. 선정 후에 child를 dictionary를 이용하여 만들어줍니다.

4) classify 함수 (class 내장함수)

```
def classify(self,test):
    if self.is_leaf:
        return self.label
    else:
        try:
            res = self.child[test[self.attri]]
        except:
            return self.label
        return res.classify(test)
```

- test data의 한 개 row에 대해 분류를 하는 함수입니다. make_tree 함수와 마찬가지로 class 내장함수입니다. 먼저 해당 노드가 leaf인지 확인 후에 맞다면 해당 leaf 노드의 label를 반환해줍니다. leaf 노드가 아니라면 재귀를 이용하여 노드를 탐색합니다.

5) information gain 관련 함수

```
def entropy(res):
    info = 0
    for i in range(0,len(res)):
        info += -(res[i]/res.sum())*np.log2(res[i]/res.sum())
    return info

def info_gain(df,label):
    gain = 0
    class_ = df.iloc[:,-1]
    data_l = df.shape[0]
    att_dic = df[label].value_counts()
    for i,j in att_dic.items():
        res = df.groupby([label,class_]).size()[i]
        gain += (res.sum()/data_l)*entropy(res)
    return gain
```

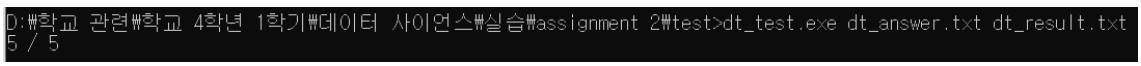
- 먼저 entropy 함수는 함수명 그대로 해당 attribute 세부항목의 entropy를 구하는 함수입니다. info_gain함수는 entropy에 가중치를 곱하여 해당 attribute의 information gain값을 구하는 함수입니다.

3. Instructions for compiling your sources coeds at TA's computer

- 파이썬을 이용하여 구현하였습니다.

4. Testing

- dt_test.txt



- dt_test1.txt