데이터 사이언스 Project2 Report

컴퓨터소프트웨어학부

2016024839 박정민

1. 코드 구성

주어진 데이터셋 파싱하여 Attribute별로 분류하여 Decision Tree를 만드는 함수와 그에 필요한 자료구조들을 모두 DecisionTree라는 클래스에서 관리하고 사용하였습니다.

```
class DecisionTree:

    def __init__(self, trainFile, testFile, resultFile):
        self.trainfile = open(trainFile, 'r')
        self.testfile = open(testFile, 'r')
        self.resultfile = open(resultFile, 'w')
        self.attribute = []
        self.trainSet = []
        self.Class = dict()
        self.root = None
        self.testSet = []
```

attribute : attribute을 관리하는 list입니다.

```
['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'car_evaluation']
```

trainSet : 학습할 데이터셋을 파싱하여 각 tuple별로 list화 하여 저장하고 관리하였습니다.

Class : trainset 데이터들의 class부분을 dict자료형으로 각 label들이 얼마나 있는지 저장하였습니다.

```
{'unacc': 975, 'acc': 302, 'vgood': 53, 'good': 52}
```

root : Decision Tree를 구성하는 Node 클래스의 객체인 root Node입니다.

testSet : 테스트할 데이터셋들을 파싱하여 각 tuple별로 list화 하여 저장하였습니다.

Decision Tree를 구성하는 각 node를 구성하는 Node Class입니다.

```
class Node:

    def __init__(self, attribute, data):
        self.attribute = attribute
        self.data = data
        self.child = []
```

attribute : 어떤 attribute으로 분류하였는지, 분류 기준은 어떻게 되는지를 저장하였습니다.

data : 학습시킨 data셋을 저장하였습니다.

child : 자식 node를 저장하였습니다.

2. Class 내 method

```
# Parse Input file and get Training Dataset
def parseLine(self):
    lines = self.trainfile.readlines()

    file_length = len(lines)

    for line in range(file_length):

        if line == 0:
            temp_attribute = ''
            for parse in lines[line]:
                if parse != '\t' and parse != '\n':
                    temp_attribute += parse

                else:
                    self.attribute.append(temp_attribute)

                    temp_attribute = ''

            print(self.attribute)
```

Train data set이 들어있는 파일을 파싱하여 self.attribute, self.trainSet 에 저장합니다. 또한 모든 데이터를 포함하는 root Node를 만들어 self.root에

저장합니다. 그리고 Decision Tree를 만드는 self.classify(self, node, attributes)를 실행합니다.

```
# Get Entropy from the Node
def getEntropy(self, labels):

    label_num = sum(labels.values())

    entropy = 0

    for p in labels.values():

        entropy += - (p / label_num) * math.log2(p / label_num)

    return entropy
```

Entropy를 구하는 함수입니다. 파라미터인 labels는 dictionary 타입으로, 각 label별로 빈도수를 저장하고 있는 labels를 참조하여 entropy를 구하여 return합니다.

```
# Calculate Entropy and Information Gain and Classify the Node
# node : Node Class Object
def classify(self, node, attributes):
    #print("Candidate Attributes : ", attributes)
    attr_len = len(attributes)
    tup_len = len(node.data)
    entropy_before = 0
    entropy_list = []

    labels = dict()

    max_info_gain = 0

    chosen_attr = ""
```

실질적으로 Decision Tree 학습 모델을 만드는 과정입니다.

각 node별로 어떤 attribute로 분류를 해야 information gain이 가장 높은지 계산한 다음, 해당 attribute로 분류하는 함수입니다. 파라미터로 Node class의 node 객체와 분류할 attribute list를 받으며, 재귀적으로 구현하였

습니다. 더 이상 분류할 attribute가 남아있지 않거나, label이 homogeneous하게 분류되었으면 함수를 종료합니다.

만약에 age attribute를 기준으로 분류하였으면 30>, 30...40, 40< 으로 분류되는 node들을 자식 node로 만들어 준 다음, 해당 자식 node들 마다 classify함수를 다시 호출합니다.

```
def parseTestFile(self):

    lines = self.testfile.readlines()

    file_length = len(lines)

    for line in range(file_length):

        if line > 0:

            temp_str = ''
            parsed = []

            for parse in lines[line]:
                if parse != '\t' and parse != '\n':
                    temp_str += parse

                else:
                    parsed.append(temp_str)

                    temp_str =''
```

Test data set파일을 파싱하여 self.trainSet에 저장하는 함수입니다.

```
# Classify Test Data Set and Predict Labels
def test(self):

    #print("Test Files")

    root = self.root

    for test_tup in self.testSet:

        #print("Test Tuple : ", test_tup)
```

만들어진 decision tree 모델에 test data set을 집어넣어 각 tuple별로 label을 예측하는 함수입니다. 각 test data set별로 while 문을 돌며 leaf node에 도착할 때까지 while 문을 실행합니다.

첫 root node부터 시작하여 자식 node 에 분류되는 attribute가 있다면 자식 node를 따라갑니다. 현재 node에 자식 node가 더 이상 없는 leaf node거나, 분류 가능한 attribute가 없으면 while 문을 종료하고 가장 비율이 높은 label을 test tuple에 배정하여 줍니다.

```
# Make Result File
def printResultFile(self):

    for idx, attr in enumerate(self.attribute):
        attr = str(attr)

        if idx == len(self.attribute) - 1:

            attr += '\n'
            self.resultfile.write(attr)

        else:
            attr += '\t'
            self.resultfile.write(attr)
```

Label 예측이 끝난 test data set을 label을 포함하여 파일로 출력하는 함수입니다.

```
def closeFile(self):
    self.testfile.close()
    self.trainfile.close()
```

열어두었던 test file과 train file을 닫는 함수입니다.

```python
def main():
    train_file = sys.argv[1]
    test_file = sys.argv[2]
    result_file = sys.argv[3]

    print(train_file, test_file, result_file)

    dt = DecisionTree(train_file, test_file, result_file)
    dt.parseLine()
    dt.parseTestFile()
    dt.test()
    dt.printResultFile()
    dt.closeFile()
```

Main 함수입니다. 모든 기능은 DecisionTree 클래스 내에 구현되어 있으므로 DecisionTree객체를 생성하여 클래스 method를 호출합니다.

3. 컴파일 환경 및 실행방법

Python 3.9.2 버전을 사용하였고 launch.json파일을 이용하여 학습시킬 데이터와 테스트 데이터, 결과 출력 파일을 실행 parameter로 전달하였습니다.

```jsonc
{
    // Use IntelliSense to learn about possible
    // Hover to view descriptions of existing a
    // For more information, visit: https://go.
    "version": "0.2.0",
    "configurations": [

        {
            "name": "Python: Current File",
            "type": "python",
            "request": "launch",
            "program": "${file}",
            "console": "integratedTerminal",
            "args": [
                "dt_train1.txt",
                "dt_test1.txt",
                "dt_result1.txt"
            ]
        }
    ]
}
```