

# 알고리즘 응용(00) Lab06

201802161 조은빈

## 1. Goal

- Data 전처리
- Scikit-learn Decision Tree 활용 및 시각화하기
- 예측 결과 Kaggle에 제출하여 점수 확인하기

## 2. 코드 분석

```
1 import numpy as np
2 import pandas as pd
3 import pydot as pydot
4 from sklearn.tree import DecisionTreeClassifier, export_graphviz
```

코드 구현 시 필요한 라이브러리를 import 한다.

```
6 def get_data(path1, path2):
7
8     # 파일 불러오기
9     train = pd.read_csv(path1)
10    test = pd.read_csv(path2)
```

get\_data이 파라미터인 path1은 train 파일 경로이고, path2는 test 파일 경로이다. read\_csv를 통해 파일을 불러온다. test 파일과 train 파일을 보면 다 동일하지만 test 파일에는 "Survived" 항목이 존재하지 않는다. 즉 Decision Tree를 통해 예측해야 할 데이터는 "Survived" 항목의 데이터이다.

```
12 # 데이터 전처리
13
14 # male은 0으로, female은 1로 변경
15 train.loc[train["Sex"] == "male", "Sex"] = 0
16 train.loc[train["Sex"] == "female", "Sex"] = 1
17
18 test.loc[test["Sex"] == "male", "Sex"] = 0
19 test.loc[test["Sex"] == "female", "Sex"] = 1
20
21 # Embarked 항목의 데이터에 대해 One Hot Incoding
22 train["Embarked_C"] = train["Embarked"] == "C"
23 train["Embarked_S"] = train["Embarked"] == "S"
24 train["Embarked_Q"] = train["Embarked"] == "Q"
25
26 test["Embarked_C"] = test["Embarked"] == "C"
27 test["Embarked_S"] = test["Embarked"] == "S"
28 test["Embarked_Q"] = test["Embarked"] == "Q"
29
30
31 # test 데이터의 Fare 항목에 있는 빈칸에 0 채우기
32 test.loc[pd.isnull(test["Fare"]), "Fare"] = 0
33
```

파일을 불러와 데이터를 전처리 한다. Decision Tree 계산을 하려면 데이터가 숫자여야 한다. train과 test의 Sex 항목의 데이터는 문자로 되어 있으므로 숫자로 바꾼다. 값이 male일 경우 0으로 바꾸고 female일 경우 1로 바꾼다. Embarked 항목의 데이터는 C, S, Q로 이루어져 있다. 표현하고자 하는 단어에 1, 아닌 단어에 0을 매핑하는 One-hot Encoding으로 데이터를 전처리 한다.

```
train["Embarked_C"] = train["Embarked"] == "C"
```

위 코드를 통해 Embarked 항목의 값이 C일 경우 Embarked\_C에 True 즉, 1을 매핑하고 C가 아닐 경우 False 즉, 0을 매핑한다. S, Q 값도 동일한 방식으로 처리한다.

Fare 항목의 데이터는 숫자로 이루어져 있기 때문에 따로 숫자로 바꾸는 과정은 필요하지 않다. 하지만 test 파일의 Fare 항목을 보면 빈 칸이 존재한다. 빈 칸이 존재할 경우 에러가 발생하기 때문에 빈 칸을 찾아 0으로 채워준다.

```
34     # 예측할 때 고려할 요소들
35     features = ["Sex", "Pclass", "SibSp", "Parch", "Fare",
36                "Embarked_C", "Embarked_Q", "Embarked_S"]
37
38     X_train = train[features]
39
40     x_test = test[features]
41
42     # 정답 항목인 Survived 데이터를 y_train에 저장
43     y_train = train["Survived"]
44
45     run_dt(X_train, y_train, x_test, features)
```

features는 결과를 예측할 때 고려할 항목명이다. Sex, Pclass, SibSp, Parch, Fare, Embarked\_C, Embarked\_Q, Embarked\_S를 사용해 결과를 예측할 것이므로 features에 저장한다.

X\_train은 train에서 고려할 항목의 데이터만 가져온 것이다. x\_test는 test에서 고려할 항목의 데이터만 가져온 것이다. y\_train은 정답 항목인 Survived 데이터를 가져온 것이다.

```

47 def run_dt(X_train, y_train, x_test, features):
48
49     # Decision Tree 만들기
50     seed = 0
51     dt = DecisionTreeClassifier(max_depth = 30, random_state = seed)
52
53     # Decision Tree 학습시키기
54     dt.fit(X_train, y_train)
55
56     # 시각화, png 파일로 저장
57     export_graphviz(dt, out_file="dt.dot", class_names=['No', 'Yes'],
58                     feature_names=features, impurity=False, filled=True)
59     (graph, ) = pydot.graph_from_dot_file('dt.dot', encoding='utf8')
60     graph.write_png('dt.png')
61
62     # 결과 예측
63     result = dt.predict(x_test)
64
65     # 예측된 결과를 submission 형식에 맞게 작성
66     submission = pd.read_csv('gender_submission.csv', index_col='PassengerId')
67     submission["Survived"] = result
68
69     # 결과를 csv 파일로 저장
70     submission.to_csv('201802161_submission.csv')

```

DecisionTreeClassifier를 통해 Decision Tree dt를 만든다. 이때 최대 깊이는 30으로 설정했다. fit 함수를 통해 Decision Tree를 학습시킨다. X\_train과 정답 데이터인 y\_train으로 학습시킨다. 만들어진 Decision Tree를 시각화한다. export\_graphviz 함수를 통해 Decision Tree를 dot 파일로 만든다. 만들어진 dot 파일을 graph에 저장하고 이를 png 파일로 만들어 저장한다. predict 함수를 통해 Decision Tree에 x\_test를 넣고 결과를 예측한다. 예측한 결과를 result에 저장한다. result를 gender\_submission.csv의 형식에 맞게 작성해서 csv 파일로 저장한다. 저장된 csv 파일을 압축해서 Kaggle 사이트에 제출하여 점수를 확인한다.

```



72 def main():
73     get_data('train.csv', 'test.csv')
74
75     if __name__ == '__main__':
76         main()

```

main 함수에서는 train과 test 파일 경로를 파라미터로 get\_data를 실행한다.

### 3. 실행 결과

- Kaggle 점수

12...	Eunbin Cho00		0.77033	1	1m
<p><b>Your First Entry</b> </p> <p>Welcome to the leaderboard!</p> <p>Your score represents your submission's accuracy. For example, a score of 0.7 in this competition indicates you predicted Titanic survival correctly for 70% of people.</p> <p>What next? You've got a few options:</p> <ul style="list-style-type: none"> <li>🧠 Learn skills that can improve your score in <a href="#">our Intro to Machine Learning course by Dan Becker</a>.</li> <li>🔍 Check out <a href="#">the discussion forum</a> to find lots of tutorials and insights from other competitors.</li> <li>🏆 Find a new challenge by entering one of our <a href="#">open, active competitions</a> or searching our <a href="#">public datasets</a>.</li> </ul>					

- Decision Tree

