

12

의사결정나무 (분류)

[의사결정나무_분류]

의사결정나무 분류트리는 지니지수를 기준으로 분기합니다.

00:00

myter 12-2. 의사결정나무(분류) Last Checkpoint: 한 시간 전 (autosaved)

Edit View Insert Cell Kernel Navigate Widgets Help Trusted Python 3 (ipy)

12.2 의사결정나무 (분류)

12.2.1 개념

가. 성장단계

- 각 마디에서 적절한 최적의 분리기칙(splitting rule) 을 찾아서 나무를 성장시키는 과정으로 적절한 정지규칙(stopping rule) d를 만족하면 중단한다
- 분리기칙을 설정하는 분리기준(splitting criterion)은 이산형 목표변수, 연속형 목표변수에 따라 다르다
- 영우형 목표변수

기준값 분리기준

지나지수 지나지수를 감소시켜주는 예측변수와 그 때의 최적 분리에 의해서 자식마디를 형성
엔트로피지수 엔트로피 지수가 가장 작은 예측변수와 이 때의 최적 분리에 의해서 자식마디를 형성

나. 가지치기 단계

- 나무의 크기를 모형의 복잡도로 볼 수 있음
- 과적합 방지를 위함

- max_depth:int, default=None
 - 트리의 최대 깊이
- min_samples_split:int or float, default=2
 - 내부 노드를 분할하는 데 필요한 최소 샘플 수
- min_samples_leaf:int or float, default=1
 - 리프노드에 있어야 하는 최소 샘플 수
- criterion:["gini", "entropy"], default="gini"
 - 분할할 때 사용할 함수
- splitter:["best", "random"], default="best"

[실습]

이번 시간에는 모델링에 집중하여 실습하도록 하겠습니다.

02:52

myter 12-2. 의사결정나무(분류) Last Checkpoint: 한 시간 전 (autosaved)

Edit View Insert Cell Kernel Navigate Widgets Help Trusted Python 3 (ipy)

예시

```
sklearn.tree.DecisionTreeClassifier(, criterion="gini", splitter="best", max_depth=None, min_samples_split=2, min_samples_leaf=1, max_features=None, random_state=None, ccp_alpha=0.0)
```

[예제 1:]
credit 데이터의 train 데이터로 credit.rating(기간 내 대출 상환 여부)을분류하는 의사결정나무 모델 만들어라

```
in [9]: import pandas as pd
import numpy as np

# 데이터 로드
credit = pd.read_csv('../data/credit_final.csv')
credit
```

Out[9]:

	credit.rating	account.balance	credit.duration.months	previous.credit.payment.status	credit.purpose	credit.amount	savings	employment.duration	installment
0	1	1	18	3	2	1049	1	1	
1	1	1	9	3	4	2799	1	2	
2	1	2	12	2	4	841	2	3	
3	1	1	12	3	4	2122	1	2	
4	1	1	12	3	4	2171	1	2	
...
995	0	1	24	2	3	1967	1	2	
996	0	1	24	2	4	2303	1	4	
997	0	3	21	3	4	12680	4	4	
998	0	2	12	2	3	6468	4	1	
999	0	1	30	2	2	6350	4	4	

[의사결정나무 모델링_매개변수 지정]

04:47

jupyter 12-2. 의사결정나무(분류) Last Checkpoint: 한 시간 전 (unsaved changes)

```

Edit View Insert Cell Kernel Navigate Widgets Help Trusted Python 3 (ipy
+ 3C Run Code
memory usage: 156.4 KB

In [8]: from sklearn.model_selection import train_test_split
# 훈련셋, 테스트셋 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.3, random_state=1)

In [6]: from sklearn.tree import DecisionTreeClassifier
# 의사결정나무 모델링
dt_clf = DecisionTreeClassifier()
## 모델 학습
dt_clf.fit(X_train, y_train)

In [25]: ## 학습용 데이터에 대한 정확도
dt_clf.score(X_train, y_train)
Out[25]: 1.0

In [26]: ## 테스트 데이터에 대한 정확도
dt_clf.score(X_test, y_test)
Out[26]: 0.8566666666666666

In [ ]: ## 예측
dt_prediction = dt_clf.predict(X_test)
dt_prediction

In [29]: from sklearn.metrics import confusion_matrix, classification_report, roc_auc_score, plot_roc_curve, precision_score, f1_score,
recall_score
confusion_matrix(y_test, dt_prediction)


```

[의사결정나무 가지치기]

그리드서치를 사용하여 정확도가 최대가 되는 가지치기를 할 수 있습니다.

07:41

jupyter 12-2. 의사결정나무(분류) Last Checkpoint: 한 시간 전 (unsaved changes)



```

Edit View Insert Cell Kernel Navigate Widgets Help Trusted Python 3 (ipy
+ 3C Run Code
memory usage: 156.4 KB

In [17]: ## ROC_AUC_SCORE
roc_auc_score(y_test, dt_clf.predict_proba(X_test)[:,1])
Out[17]: 0.6309523809523809

12.2.2 가지치기

12.2.2.1 주요 매개변수 조정

In [18]: dt_clf.get_depth()
Out[18]: 16

In [31]: from sklearn.model_selection import GridSearchCV
param_grid = {'max_depth': range(2, 16, 1), 'min_samples_leaf': range(1, 20, 1)}
model_grid_tree = GridSearchCV(DecisionTreeClassifier(), param_grid, cv=5)
model_grid_tree.fit(X_train, y_train)
Out[31]: GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
param_grid={'max_depth': range(2, 16),
'min_samples_leaf': range(1, 20)})

In [32]: model_grid_tree.best_estimator_
Out[32]: DecisionTreeClassifier(max_depth=9, min_samples_leaf=5)

In [38]: model_grid_tree.best_score_

```

[참고 : ccp-alpha 매개변수 조정]

ccp-alpha값으로 가지치기를 할 수 있습니다.

- 하지만 이전에 했던 매개변수 조정보다 복잡하므로 많이 사용하지는 않습니다.

13:28

myter 12-2. 의사결정나무(분류) Last Checkpoint: 전 시간 전 (autosaved)

```

In [40]: path = dt_clf.cost_complexity_pruning_path(X_train, y_train)
path

Out[40]: {'ccp_alphas': array([0.00990226, 0.00991097, 0.00119048, 0.00122449,
0.00122449, 0.00126211, 0.00126984, 0.00126984, 0.00131868,
0.00133929, 0.00134921, 0.00195714, 0.00196054, 0.00196364,
0.00198996, 0.00140056, 0.0014127, 0.0014891, 0.0015873,
0.00166667, 0.00166667, 0.00175283, 0.00182963, 0.00185064,
0.00190476, 0.00190476, 0.00190476, 0.00190476, 0.00190476,
0.00190476, 0.00202691, 0.00205882, 0.00214286, 0.00214286,
0.00214286, 0.00214286, 0.00217687, 0.00227546,
0.00233432, 0.00241758, 0.00243506, 0.00244988, 0.00245238,
0.00247619, 0.0025, 0.00258963, 0.00259662, 0.00259795,
0.00267284, 0.0027551, 0.00287415, 0.00306096, 0.00338624,
0.00378711, 0.00387636, 0.0041662, 0.00417262, 0.00445055,
0.00511711, 0.00606213, 0.00752796, 0.00857532, 0.01353158,
0.02487219, 0.04253428]),
'impurities': array([0.00270677, 0.00543969, 0.00782064, 0.01026962,
0.0127166, 0.01524282, 0.0177825, 0.0203219, 0.02295955,
0.02563812, 0.03103495, 0.0374923, 0.03647032, 0.03919759,
0.04197752, 0.04477864, 0.04760403, 0.05059022, 0.05534213,
0.06034213, 0.06534213, 0.07585909, 0.08134797, 0.08879054,
0.0960693, 0.0960693, 0.0960693, 0.0960693, 0.1001911])}

```