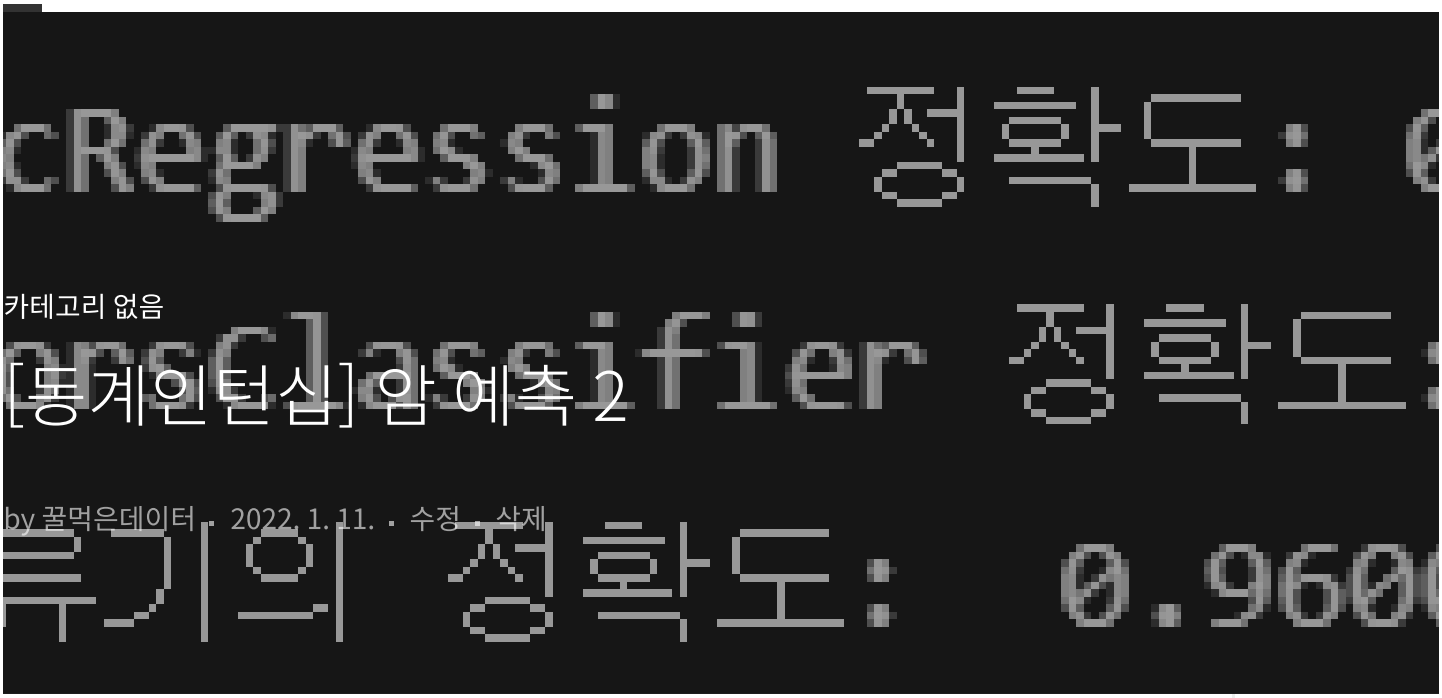


카테고리 없음

[동계인턴십]암 예측 2

by 꿀먹은데이터 · 2022. 1. 11. · 수정 · 삭제

```
features=['STOMA','COLON','LIVER','LUNG','PROST','THROI','BREAC'
,'RECTM']
y_df =df['LUNG']
X_df =df.drop(features, axis=1)

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X_df,y_df,test_
size=0.2)

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score
```

```
logistic_regression = LogisticRegression()
knn = KNeighborsClassifier(n_neighbors=8)

voting_model = VotingClassifier(estimators=[ ('LogisticRegressio
n', logistic_regression),
                                             ('KNN', knn) ], voti
ng='soft')
```

```
classifiers = [logistic_regression, knn]
for classifier in classifiers:
```

분류 전체보기

파이썬 머신러닝 완벽 가이드

AI 데이터 연구단

공지사항

최근글 : 인기글

[2022 동계인턴십]암… 2022.01.12

[동계인턴십] 암 예… 2022.01.11

[2022 동계인턴십] 암 예측 2022.01.10

```
    classifier.fit(X_train, y_train)
    pred = classifier.predict(X_test)
    class_name = classifier.__class__.__name__
    print('{0} 정확도: {1:.4f}'.format(class_name, accuracy_scor
e(y_test, pred)))

voting_model.fit(X_train, y_train)
pred = voting_model.predict(X_test)
print('보팅 분류기의 정확도: {0: .4f}'.format(accuracy_score(y_t
est, pred)))
```

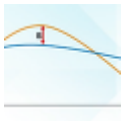
```
LogisticRegression 정확도: 0.9667
KNeighborsClassifier 정확도: 0.9533
보팅 분류기의 정확도:  0.9600
```

```
import xgboost as xgb ## XGBoost 불러오기
from xgboost import plot_importance ## Feature Importance를 불러
오기 위함
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, pr
ecision_score, recall_score
from sklearn.metrics import confusion_matrix, f1_score, roc_auc_
score
import warnings
warnings.filterwarnings('ignore')

# 전체 데이터셋을 학습용 80%, 테스트용 20%로 분할
X_train, X_test, y_train, y_test = train_test_split(X_df, y_df,
 test_size=0.2, random_state=156)
print(X_train.shape, X_test.shape)
```

```
(597, 33) (150, 33)
```

```
# 넘파이 형태의 학습 데이터 세트와 테스트 데이터를 DMatrix로 변
환하는 예제
dtrain = xgb.DMatrix(data=X_train, label = y_train)
dtest = xgb.DMatrix(data=X_test, label=y_test)

params = {'max_depth' : 3,
          'eta' : 0.1,
          'objective' : 'binary:logistic',
          'eval_metric' : 'logloss',
          'early_stoppings' : 100 }

num_rounds = 400

# train 데이터 세트는 'train', evaluation(test) 데이터 세트는 'e
val' 로 명기
wlist = [(dtrain, 'train'), (dtest,'eval')]
# 하이퍼 파라미터와 early stopping 파라미터를 train() 함수의 파
라미터로 전달
xgb_model = xgb.train(params = params, dtrain=dtrain, num_boost_
round=num_rounds, evals=wlist)
```

[FIND-A] 금융경제학…
2022.01.09



Sleep AI Challen…
2022.01.07

최근댓글

태그

사이킷런, 머신러닝, 파이썬머신러닝완벽가이드, 수면다원검사, 수면, AI데이터연구단

—

전체 방문자

3

Today : 0
Yesterday : 0

```
[0]    train-logloss:0.60389   eval-logloss:0.60396
[1]    train-logloss:0.53027   eval-logloss:0.53107
[2]    train-logloss:0.46889   eval-logloss:0.47020
[3]    train-logloss:0.41671   eval-logloss:0.41902
[4]    train-logloss:0.37218   eval-logloss:0.37465
[5]    train-logloss:0.33382   eval-logloss:0.33669
[6]    train-logloss:0.30031   eval-logloss:0.30338
[7]    train-logloss:0.27116   eval-logloss:0.27563
[8]    train-logloss:0.24548   eval-logloss:0.25081
[9]    train-logloss:0.22300   eval-logloss:0.22899
[10]   train-logloss:0.20267   eval-logloss:0.20977
[11]   train-logloss:0.18456   eval-logloss:0.19321
[12]   train-logloss:0.16846   eval-logloss:0.17886
[13]   train-logloss:0.15419   eval-logloss:0.16629
[14]   train-logloss:0.14136   eval-logloss:0.15492
[15]   train-logloss:0.12986   eval-logloss:0.14412
[16]   train-logloss:0.11964   eval-logloss:0.13518
...
[395]  train-logloss:0.00425   eval-logloss:0.07943
[396]  train-logloss:0.00424   eval-logloss:0.07957
[397]  train-logloss:0.00424   eval-logloss:0.07964
[398]  train-logloss:0.00424   eval-logloss:0.07981
[399]  train-logloss:0.00423   eval-logloss:0.07967
```

```
pred_probs = xgb_model.predict(dtest)
print('predict() 수행 결과값을 10개만 표시, 예측 확률 값으로 표시됨')
print(np.round(pred_probs[:10], 3))

# 예측 확률이 0.5보다 크면 1, 그렇지 않으면 0으로 예측값 결정해 리스트 객체인 preds에 저장
preds = [ 1 if x > 0.5 else 0 for x in pred_probs]
print('예측값 10개만 표시: ', preds[:10])
```

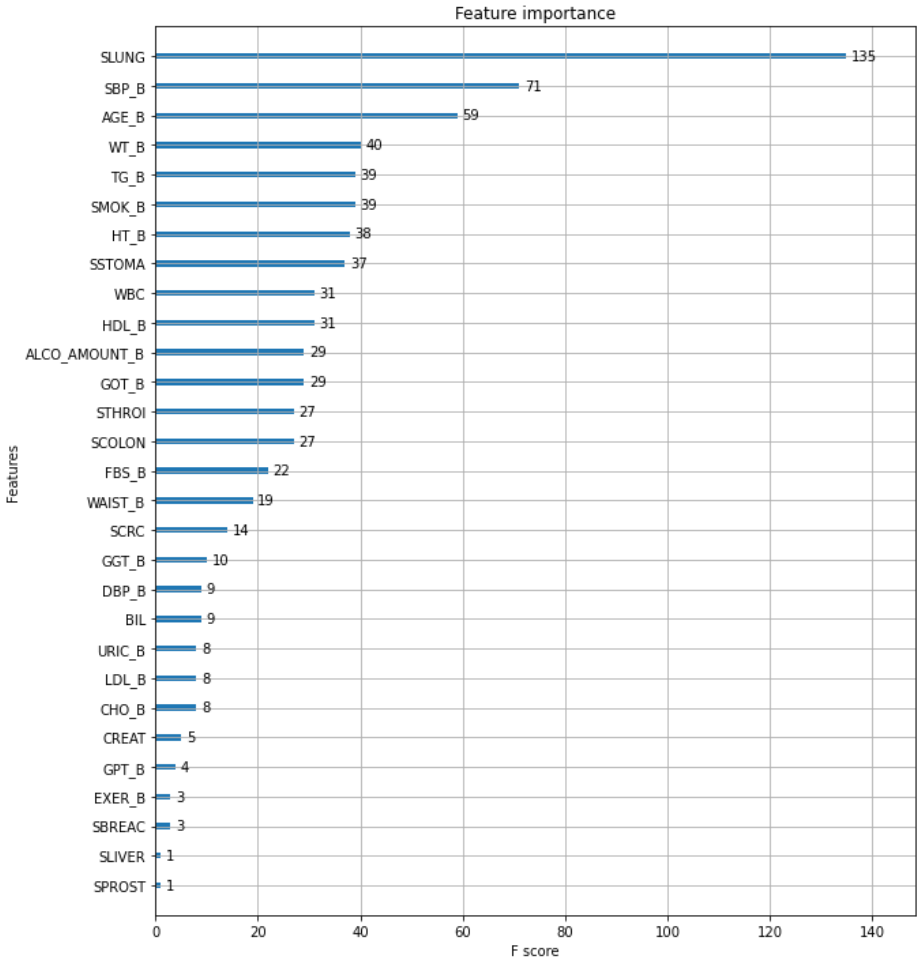
```
predict() 수행 결과값을 10개만 표시, 예측 확률 값으로 표시됨
[0.926 0.    0.848 0.002 0.    0.063 0.    0.    0.001 0.011]
예측값 10개만 표시:  [1, 0, 1, 0, 0, 0, 0, 0, 0, 0]
```

```
# 혼동행렬, 정확도, 정밀도, 재현율, F1, AUC 불러오기
def get_clf_eval(y_test, y_pred):
    confusion = confusion_matrix(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    F1 = f1_score(y_test, y_pred)
    AUC = roc_auc_score(y_test, y_pred)
    print('오차행렬:Wn', confusion)
    print('Wn정확도: {:.4f}'.format(accuracy))
    print('정밀도: {:.4f}'.format(precision))
    print('재현율: {:.4f}'.format(recall))
```

```
    print('F1: {:.4f}'.format(F1))
    print('AUC: {:.4f}'.format(AUC))

get_clf_eval(y_test, preds)
```

```
오차행렬:
 [[144   0]
 [  3   3]]

정확도: 0.9800
정밀도: 1.0000
재현율: 0.5000
F1: 0.6667
AUC: 0.7500
```

```
from xgboost import plot_importance
import matplotlib.pyplot as plt
%matplotlib inline

fig, ax = plt.subplots(figsize=(10, 12))
plot_importance(xgb_model, ax=ax)
```




```
# max_depth = 3, 학습률은 0.1, 예제가 이진분류이므로 목적함수(objective)는 binary:logistic(이진 로지스틱)
# 부스팅 반복횟수는 400
```

```
from xgboost import XGBClassifier

xgb_wrapper = XGBClassifier(n_estimators = 400, learning_rate =
0.1, max_depth = 3)
xgb_wrapper.fit(X_train, y_train)
w_preds = xgb_wrapper.predict(X_test)
```

```
[17:49:10] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
# max_depth = 3, 학습률은 0.1, 예제가 이진분류이므로 목적함수(ob
jective)는 binary:logistic(이진 로지스틱)
# 오류함수의 평가성능지표는 logloss
# 부스팅 반복횟수는 400
# 조기중단을 위한 최소 반복횟수는 100

# 아래 예제에서는 평가를 위한 데이터 세트로 테스트 데이터 세트를
사용했지만, 바람직하진 않습니다.
# 테스트 데이터 세트는 학습에 완전히 알려지지 않은 데이터 세트를
사용해야 합니다.
# 평가에 테스트 데이터 세트를 사용하면 학습시에 미리 참고가 되어
과적합할 수 있기 때문입니다.

xgb_wrapper = XGBClassifier(n_estimators = 400, learning_rate =
0.1 , max_depth = 3)
evals = [(X_test, y_test)]
xgb_wrapper.fit(X_train, y_train, early_stopping_rounds = 100,
                eval_metric="logloss", eval_set = evals, verbose
=True)
ws100_preds = xgb_wrapper.predict(X_test)
```

```
[0]     validation_0-logloss:0.60396
[1]     validation_0-logloss:0.53107
[2]     validation_0-logloss:0.47020
[3]     validation_0-logloss:0.41902
[4]     validation_0-logloss:0.37465
[5]     validation_0-logloss:0.33669
[6]     validation_0-logloss:0.30338
[7]     validation_0-logloss:0.27563
[8]     validation_0-logloss:0.25081
[9]     validation_0-logloss:0.22899
[10]    validation_0-logloss:0.20977
[11]    validation_0-logloss:0.19321
[12]    validation_0-logloss:0.17886
[13]    validation_0-logloss:0.16629
[14]    validation_0-logloss:0.15492
[15]    validation_0-logloss:0.14412
[16]    validation_0-logloss:0.13518
[17]    validation_0-logloss:0.12691
[18]    validation_0-logloss:0.12019
[19]    validation_0-logloss:0.11389
[20]    validation_0-logloss:0.10893
[21]    validation_0-logloss:0.10362
[22]    validation_0-logloss:0.09895
[23]    validation_0-logloss:0.09498
[24]    validation_0-logloss:0.09060
...
[146]   validation_0-logloss:0.07594
[147]   validation_0-logloss:0.07619
[148]   validation_0-logloss:0.07647
[149]   validation_0-logloss:0.07616
[150]   validation_0-logloss:0.07644
```

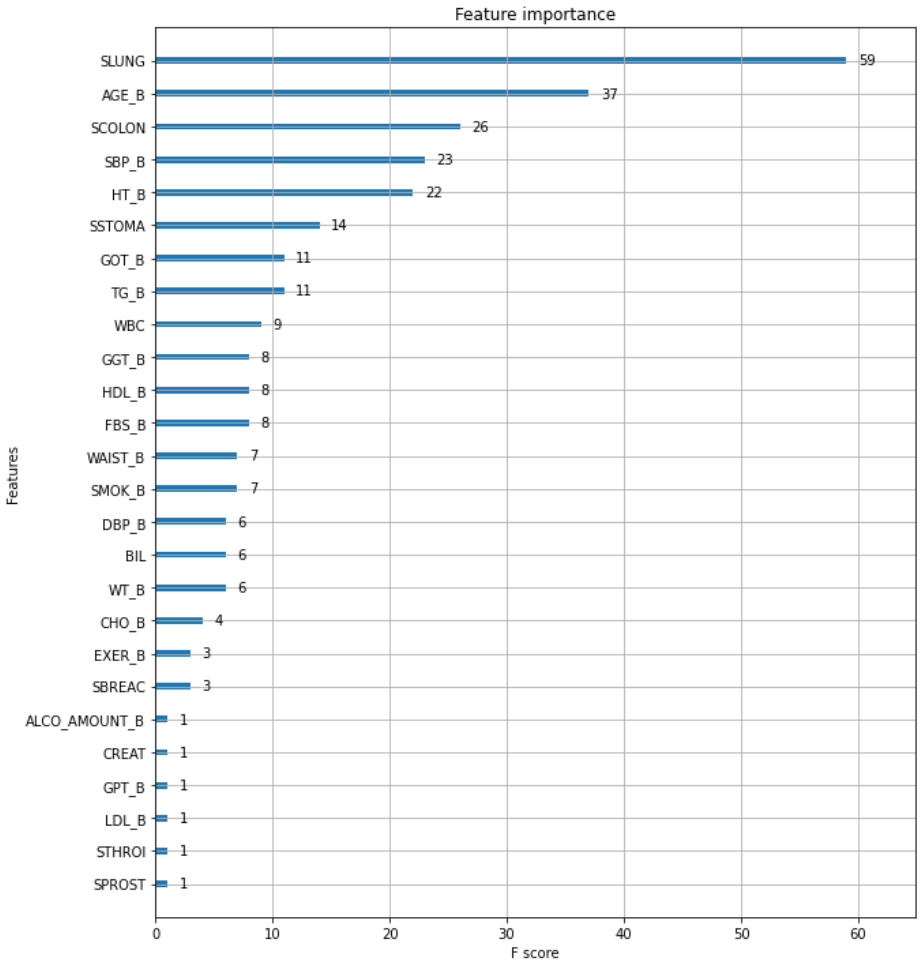
```
get_clf_eval(y_test, ws100_preds)
```

```
오차행렬:
 [[144   0]
 [  3   3]]

정확도: 0.9800
정밀도: 1.0000
재현율: 0.5000
F1: 0.6667
AUC: 0.7500
```

```
# early_stopping_rounds = 10 으로 설정하고 재학습
xgb_wrapper.fit(X_train, y_train, early_stopping_rounds = 10,
                eval_metric='logloss', eval_set=evals , verbose=
True)

ws10_preds = xgb_wrapper.predict(X_test)
get_clf_eval(y_test, ws10_preds)
```

```
[0]    validation_0-logloss:0.60396
[1]    validation_0-logloss:0.53107
[2]    validation_0-logloss:0.47020
[3]    validation_0-logloss:0.41902
[4]    validation_0-logloss:0.37465
[5]    validation_0-logloss:0.33669
[6]    validation_0-logloss:0.30338
[7]    validation_0-logloss:0.27563
[8]    validation_0-logloss:0.25081
[9]    validation_0-logloss:0.22899
[10]   validation_0-logloss:0.20977
[11]   validation_0-logloss:0.19321
[12]   validation_0-logloss:0.17886
[13]   validation_0-logloss:0.16629
[14]   validation_0-logloss:0.15492
[15]   validation_0-logloss:0.14412
[16]   validation_0-logloss:0.13518
[17]   validation_0-logloss:0.12691
[18]   validation_0-logloss:0.12019
[19]   validation_0-logloss:0.11389
[20]   validation_0-logloss:0.10893
[21]   validation_0-logloss:0.10362
[22]   validation_0-logloss:0.09895
[23]   validation_0-logloss:0.09498
[24]   validation_0-logloss:0.09060
...
정확도: 0.9800
정밀도: 1.0000
재현율: 0.5000
F1: 0.6667
AUC: 0.7500
```

```
from xgboost import plot_importance
import matplotlib.pyplot as plt
%matplotlib inline

fig, ax = plt.subplots(figsize=(10, 12))

plot_importance(xgb_wrapper, ax=ax)
```



공감

댓글 0

여러분의 소중한 댓글을 입력해주세요.

등록

TEL. 02.1234.5678 / 경기 성남시 분당구 판교역로