



분류 전체보기 

파이썬 머신러닝 완벽 가이드

AI 데이터 연구단

공지사항

최근글 : 인기글

[2022 동계
인턴십]암...
2022.01.12

PCANB0	996
PCANB9	982
FCANB0	978
FCANB1	931
FCANB2	999

[동계인턴
십] 암 예...
2022.01.11

ession 정
assifier 정
정확도:

[2022 동계인턴십] 암 예측
2022.01.10

```
import pandas as pd
data=pd.read_csv('C:/Users/dudtj/OneDrive - 송실대학교 - Soongsil University/Desktop/CL/python/동계인턴십_Data_1000/phenotype_1000.txt',engine="python",sep=" ")

print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 66 columns):
#   Column                Non-Null Count  Dtype
---  -
0   FID                    1000 non-null  object
1   IID                    1000 non-null  object
2   AGE_B                  1000 non-null  int64
3   SMOK_B                 990 non-null   float64
4   SMOKA_MOD_B           679 non-null   float64
5   ALCO_B                 965 non-null   float64
6   ALCO_AMOUNT_B         915 non-null   float64
7   EXER_B                927 non-null   float64
8   MDM_B                  46 non-null    float64
9   MHTN_B                120 non-null   float64
10  MLPD_B                 28 non-null    float64
11  PHTN_B                 153 non-null   float64
12  PDM_B                  67 non-null    float64
13  PLPD_B                 36 non-null    float64
14  HT_B                  998 non-null   float64
15  WT_B                  998 non-null   float64
16  WAIST_B               971 non-null   float64
17  SBP_B                 988 non-null   float64
18  DBP_B                 989 non-null   float64
19  CHO_B                 995 non-null   float64
20  LDL_B                 995 non-null   float64
21  TG_B                  995 non-null   float64
22  HDL_B                 987 non-null   float64
23  FBS_B                 998 non-null   float64
24  GOT_B                 996 non-null   float64
25  GPT_B                 996 non-null   float64
26  GGT_B                 992 non-null   float64
27  URIC_B                929 non-null   float64
28  PCAN80                 4 non-null     float64
29  PCAN81                 15 non-null    float64
30  PCAN82                 0 non-null     float64
31  PCAN83                 1 non-null     float64
32  PCAN84                 8 non-null     float64
33  PCAN86                 4 non-null     float64
34  PCAN89                 18 non-null    float64
35  FCAN80                 30 non-null    float64
36  FCAN81                 69 non-null    float64
37  FCAN82                 1 non-null     float64
38  FCAN83                 23 non-null    float64
39  FCAN84                 20 non-null    float64
40  FCAN86                 35 non-null    float64
41  FCAN89                 11 non-null    float64
42  FEV1                  569 non-null   float64
43  FVC                   569 non-null   float64
44  BIL                   948 non-null   float64
45  WBC                   928 non-null   float64
46  CREAT                 962 non-null   float64
47  STOMA                 1000 non-null  int64
48  COLON                 1000 non-null  int64
49  LIVER                 1000 non-null  int64
50  LUNG                  1000 non-null  int64
51  PROST                 1000 non-null  int64
52  THROI                 1000 non-null  int64
53  BREAC                 1000 non-null  int64
54  RECTM                 1000 non-null  int64
55  SCOLON                1000 non-null  float64
56  SRECTM                1000 non-null  float64
57  SPROST                1000 non-null  float64
58  STHROI                1000 non-null  float64
59  SBREAC                1000 non-null  float64
60  SLUNG                 1000 non-null  float64
61  SSTOMA                1000 non-null  float64
62  SLIVER                1000 non-null  float64
63  SEX1                  1000 non-null  int64
64  CRC                   1000 non-null  int64
65  SCRC                  1000 non-null  float64
dtypes: float64(53), int64(11), object(2)
memory usage: 515.8+ KB
None
```

AGE_B	0
HDL_B	0
FBS_B	0
GOT_B	0
GPT_B	0
GGT_B	0
URIC_B	0

[FIND-A] 글
 융경제학...
 2022.01.09



Sleep AI
 Challen...
 2022.01.07



최근댓글

태그

사이킷런, 머신러닝,
 파이썬머신러닝완벽가이
 드,
 수면다원검사, 수면,
 AI데이터연구단

위 그림과 같이 null값이 데이터의 절반 이상을 차지하는 변수들이 있다. 데
 이터 보존을 위해 non-null값이 900 이상인 데이터들만 가져와 다시 df로 저
 장하였다. (1000개의 데이터이므로 90%만 뽑았다.)

이렇게 할 수 있는 이유는 drop시킨 변수들은 암 예측에 중요하지 않다고 판
 단한 변수들이었기 때문이다.

data.columns

```
Index(['FID', 'IID', 'AGE_B', 'SMOK_B', 'SMOKA_MOD_B', 'ALCO_B',
       'ALCO_AMOUNT_B', 'EXER_B', 'MDM_B', 'MHTN_B', 'MLPD_B', 'PHTN_B',
       'PDM_B', 'PLPD_B', 'HT_B', 'WT_B', 'WAIST_B', 'SBP_B', 'DBP_B', 'CHO_B',
       'LDL_B', 'TG_B', 'HDL_B', 'FBS_B', 'GOT_B', 'GPT_B', 'GGT_B', 'URIC_B',
       'PCAN80', 'PCAN81', 'PCAN82', 'PCAN83', 'PCAN84', 'PCAN86', 'PCAN89',
       'FCAN80', 'FCAN81', 'FCAN82', 'FCAN83', 'FCAN84', 'FCAN86', 'FCAN89',
       'FEV1', 'FVC', 'BIL', 'WBC', 'CREAT', 'STOMA', 'COLON', 'LIVER', 'LUNG',
       'PROST', 'THROI', 'BREAC', 'RECTM', 'SCOLON', 'SRECTM', 'SPROST',
       'STHROI', 'SBREAC', 'SLUNG', 'SSTOMA', 'SLIVER', 'SEX1', 'CRC', 'SCRC'],
      dtype='object')
```

```
df=data.loc[:,['AGE_B', 'SMOK_B', 'ALCO_B', 'ALCO_AMOUNT_B', 'EXER_
B', 'HT_B', 'WT_B', 'WAIST_B', 'SBP_B', 'DBP_B', 'CHO_B', 'LDL_B', 'TG_
B', 'HDL_B', 'FBS_B', 'GOT_B', 'GPT_B', 'GGT_B', 'URIC_B', 'BIL', 'WBC',
'CREAT', 'STOMA', 'COLON', 'LIVER', 'LUNG', 'PROST', 'THROI', 'BREAC',
'RECTM', 'SCOLON', 'SRECTM', 'SPROST', 'STHROI', 'SBREAC', 'SLUNG', 'SS
TOMA', 'SLIVER', 'SEX1', 'CRC', 'SCRC']]
```

```
print(df.info())
```

—

전체 방문자

3

Today : 0

Yesterday : 0

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 41 columns):
#   Column                Non-Null Count  Dtype
---  -
0   AGE_B                 1000 non-null   int64
1   SMOK_B               990 non-null    float64
2   ALCO_B               965 non-null    float64
3   ALCO_AMOUNT_B        915 non-null    float64
4   EXER_B               927 non-null    float64
5   HT_B                 998 non-null    float64
6   WT_B                 998 non-null    float64
7   WAIST_B              971 non-null    float64
8   SBP_B                988 non-null    float64
9   DBP_B                989 non-null    float64
10  CHO_B                995 non-null    float64
11  LDL_B                995 non-null    float64
12  TG_B                 995 non-null    float64
13  HDL_B                987 non-null    float64
14  FBS_B                998 non-null    float64
15  GOT_B                996 non-null    float64
16  GPT_B                996 non-null    float64
17  GGT_B                992 non-null    float64
18  URIC_B               929 non-null    float64
19  BIL                  948 non-null    float64
20  WBC                  928 non-null    float64
21  CREAT                962 non-null    float64
22  STOMA                1000 non-null    int64
23  COLON                1000 non-null    int64
24  LIVER                1000 non-null    int64
25  LUNG                 1000 non-null    int64
26  PROST                1000 non-null    int64
27  THROI                1000 non-null    int64
28  BREAC                1000 non-null    int64
29  RECTM                1000 non-null    int64
30  SCOLON               1000 non-null    float64
31  SRECTM               1000 non-null    float64
32  SPROST               1000 non-null    float64
33  STHROI               1000 non-null    float64
34  SBREAC               1000 non-null    float64
35  SLUNG                1000 non-null    float64
36  SSTOMA               1000 non-null    float64
37  SLIVER               1000 non-null    float64
38  SEX1                 1000 non-null    int64

```

```
38  SEX1          1000 non-null    int64
39  CRC           1000 non-null    int64
40  SCRC          1000 non-null    float64
dtypes: float64(30), int64(11)
memory usage: 320.4 KB
None
```

이와 같이 null값이 많지 않은 변수들을 df에 저장한 뒤 dropna()함수를 통해 그 데이터를 날리고 검정을 하였다. 그 이유는 제거를 한 뒤 데이터의 양은 747개로 충분하다고 생각하여 이를 test와 train set으로 분리하여 진행하였다.

```
df.dropna(inplace=True)
print(df.isnull().sum())
```

AGE_B	0
SMOK_B	0
ALCO_B	0
ALCO_AMOUNT_B	0
EXER_B	0
HT_B	0
WT_B	0
WAIST_B	0
SBP_B	0
DBP_B	0
CHO_B	0
LDL_B	0
TG_B	0
HDL_B	0
FBS_B	0
GOT_B	0
GPT_B	0
GGT_B	0
URIC_B	0
BIL	0
WBC	0
CREAT	0
STOMA	0
COLON	0
LIVER	0
...	
SLIVER	0
SEX1	0
CRC	0
SCRC	0
dtype: int64	

```

from sklearn.preprocessing import LabelEncoder
def format_features(df):
    features=['STOMA', 'COLON', 'LIVER', 'LUNG', 'PROST', 'THROI', 'BREAC', 'RECTM']
    for feature in features:
        le=LabelEncoder()
        le=le.fit(df[feature])
        df[feature]=le.transform(df[feature])
    return df

```

features를 암종류로 놓고 y를 'LUNG'암으로 두고 X를 이 features를 제거한 변수로 둔 뒤 진행하였다.

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

train set과 test set의 비율은 8:2 비율로 진행하였다.

```
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.metrics import accuracy_score

dt_clf = DecisionTreeClassifier()
rf_clf = RandomForestClassifier()
lr_clf = LogisticRegression()

#DecisionTreeClassifier 학습/예측/평가
dt_clf.fit(X_train,y_train)
dt_pred = dt_clf.predict(X_test)
print("DecisionTreeClassifier 정확도 :",accuracy_score(y_test,dt_
_pred))

#RandomForestClassifier 학습/예측/평가
rf_clf.fit(X_train,y_train)
rf_pred = rf_clf.predict(X_test)

print("RandomForestClassifier 정확도 :",accuracy_score(y_test,rf_
_pred))
#LogisticRegression 학습/예측/평가
lr_clf.fit(X_train,y_train)
lr_pred = lr_clf.predict(X_test)
print("LogisticRegression 정확도 :",accuracy_score(y_test,lr_pre
d))
```

```
DecisionTreeClassifier 정확도 : 0.9866666666666667
RandomForestClassifier 정확도 : 0.9533333333333334
LogisticRegression 정확도 : 0.9666666666666667

C:\Users\dudtj\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(
```

위와 같이 DecisionTree나 RandomForest, LogisticRegression이 모두 95%보다 높은 정확도를 볼 수 있었다.

```
import numpy as np
from sklearn.model_selection import KFold
def exec_kfold(clf,folds=5):
    kfold=KFold(n_splits=folds)
    scores=[]
```

```

for iter_count, (train_index, test_index) in enumerate(kfold.split(X_df)):
    X_train, X_test = X_df.values[train_index], X_df.values[test_index]
    y_train, y_test = y_df.values[train_index], y_df.values[test_index]
    #Classifier 학습, 예측, 정확도 계산
    clf.fit(X_train, y_train)
    predictions = clf.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)
    scores.append(accuracy)
    print("교차 검증 정확도:", iter_count, accuracy)

mean_score = np.mean(scores)
print("평균 정확도:", mean_score)
exec_kfold(dt_clf, folds=5)

```

```

교차 검증 정확도: 0 0.9333333333333333
교차 검증 정확도: 1 0.9733333333333334
교차 검증 정확도: 2 0.959731543624161
교차 검증 정확도: 3 0.9932885906040269
교차 검증 정확도: 4 0.959731543624161
평균 정확도: 0.9638836689038032

```

교차 검증을 통한 정확도 또한 95%를 넘는 정확도로 상당히 높은 수치다.

```

from sklearn.model_selection import cross_val_score

scores = cross_val_score(dt_clf, X_df, y_df, cv=5)
for iter_count, accuracy in enumerate(scores):
    print("교차 검증 정확도:", iter_count, accuracy)

print("평균 정확도:", np.mean(scores))

```

```

교차 검증 정확도: 0 0.9266666666666666
교차 검증 정확도: 1 0.9533333333333334
교차 검증 정확도: 2 0.9932885906040269
교차 검증 정확도: 3 0.9865771812080537
교차 검증 정확도: 4 0.959731543624161
평균 정확도: 0.9639194630872483

```

```

from sklearn.model_selection import GridSearchCV
parameters = {'max_depth': [2, 3, 5, 10], 'min_samples_split': [2, 3, 5], 'min_samples_leaf': [1, 5, 8]}

grid_dclf = GridSearchCV(dt_clf, param_grid=parameters, scoring='accuracy', cv=5)
grid_dclf.fit(X_train, y_train)

print("GridSearchCV 최적 하이퍼 파라미터:", grid_dclf.best_params_)

```

```
print("GridSearchCV 최고 정확도:", grid_dclf.best_score_)
best_dclf = grid_dclf.best_estimator_

dpredictions = best_dclf.predict(X_test)
accuracy = accuracy_score(y_test, dpredictions)
print("테스트 세트에서의 DecisionTreeClassifier 정확도 :", accuracy)
```

```
GridSearchCV 최적 하이퍼 파라미터: {'max_depth': 2, 'min_samples_leaf': 1,
'min_samples_split': 2}
GridSearchCV 최고 정확도: 0.9849439775910364
테스트 세트에서의 DecisionTreeClassifier 정확도 : 0.9733333333333334
```

최적 하이퍼 파라미터는 max_depth가 2, min_samples_leaf가 1, min_samples_split이 2가 나왔고, 이에 대한 최고 정확도는 98.49%가 나왔다.

♡ 공감   

댓글 0

여러분의 소중한 댓글을 입력해주세요.

등록

 1 2 3 4 5 6 7 ... 15 