

뇌인지과학의 방법

14주차 과제

Brain Imaging and Analysis for Clinical, Cognitive Neuroscience

차지욱 교수님 연구실 김가경
bettybetty3k@gmail.com

실습 과제 데이터

ABCD study 데이터

- 참여자: 9~10세 아동
- 데이터: 뇌, 유전, 다양한 환경 변수들
 - 실습에서는 뇌와 성별 정보만 다룰 것임.
 - 뇌 데이터: Morphometric Data (형태학적 데이터)
 - '(a)반구위치_(b)S/G_(c)region_(d)metric'의 형태
 - 예) lh_S_이름_area_.1
 - 성별: sex
 - Male == 0
 - Female == 1

| | |
|----------------------------|--|
| (a)반구위치 | (c)region |
| lh: left hemisphere (좌반구) | region of interest |
| rh: right hemisphere (우반구) | 세부 이름은 atlas 참고 |
| (b)S/G | (d)metric |
| S: sulcus | area |
| G: gyrus | mean curvature thickness volume |

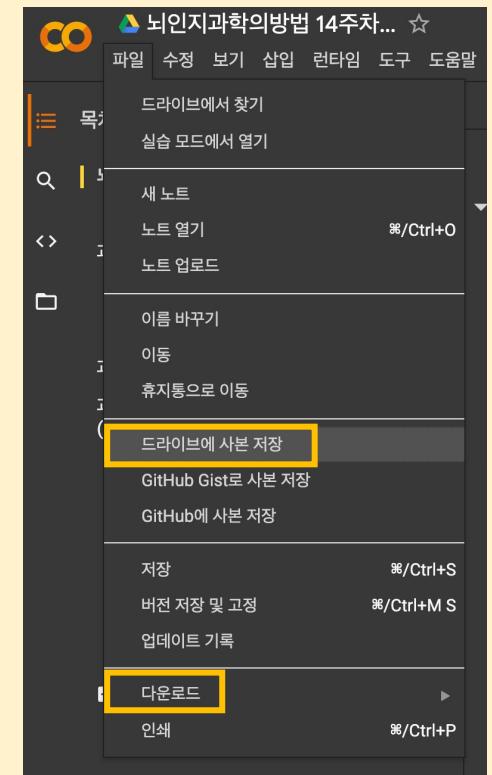
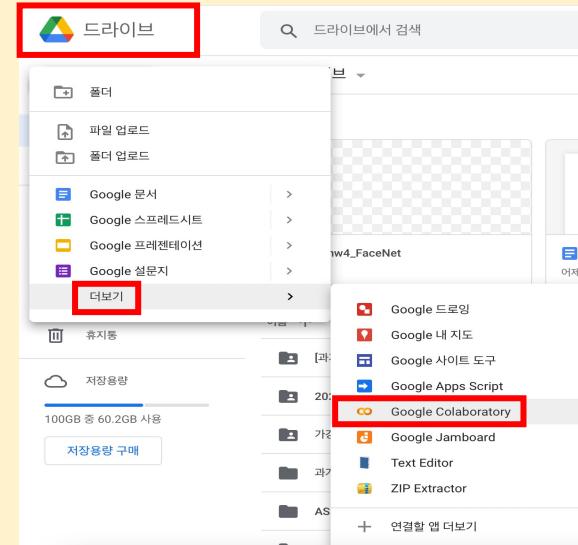
0. Colab 이용해서 실험하기

Colab

- <https://colab.research.google.com>

Colab tutorial code

- https://colab.research.google.com/drive/1ia6_c1K8jiKOGSluu1GIkdKLJwVlxdFN?usp=sharing
- 위 링크를 참고하여 데이터에 맞게 코드 수정
- 위 링크에 직접 수정하지 말고, 사본 만들어 수정



1. Preprocessing 전후 데이터로 실험 결과 비교하기

- 데이터 전처리의 중요성
 - 데이터 전처리는 통계적 추론을 보다 빠르고 정확하게 끌어낼 수 있도록 도와줌.
 - 예: 데이터의 표준화 (zero-mean, unit variance), gaussian transformation 등
 - 본 과제에서는 가장 기본적인 표준화 방법인 **standard scaler**와 non-gaussian 분포의 데이터를 gaussian 으로 가깝게 바꿔주면서 표준화를 하는 **power transformer**를 비교하고자 함.

1. Preprocessing 전후 데이터로 실험 결과 비교하기

- [과제] (1) Standard Scaler, (2) Power Transformer 의 전처리 방법으로 각각 classification 실험하고 결과 비교하기
 - 전처리: 실행 코드 및 preprocessing 완료된 데이터 일부 캡쳐 하여 제출
 - 전처리 데이터 실험 결과 비교하기
- 참고 사이트: <https://scikit-learn.org/stable/modules/preprocessing.html>

1. Preprocessing 전후 데이터로 실험 결과 비교하기

- 데이터 사용을 위한 환경 설정

```
▶ ### environment setting
from google.colab import drive
drive.mount('/content/drive')
```

... Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473

Enter your authorization code:



- 데이터 로드

```
[ ] ### load data
import pandas as pd
[ ] 데이터 경로 수정 필요
train = pd.read_csv("/content/drive/MyDrive/Colab/class_BCSmethod/TA_test.csv")
test = pd.read_csv("/content/drive/MyDrive/Colab/class_BCSmethod/TA_test.csv")
```

1. Preprocessing 전후 데이터로 실험 결과 비교하기

<주의>

노 데이터만 전처리 하여야 함.

Numpy array 형식을 data frame으로 변환하는 과정이 추가될 수 있음.

(1) Standard Scaler

```
▶ ### preprocessing with (1) standard scaler  
### import module  
  
### train data  
# Write your code here  
  
### test data  
# Write your code here
```

(2) Power Transformer

```
▶ ### preprocessing with (2) power transformer  
### import module  
  
### train data  
# Write your code here  
  
### test data  
# Write your code here
```



전처리: 실행 코드 및 preprocessing 완료된 데이터 일부 캡쳐하여 제출

1. Preprocessing 전후 데이터로 실험 결과 비교하기

- 파일 저장하기

```
### write csv file
from google.colab import drive
drive.mount('/content/drive')

%cd /content/drive/MyDrive/Colab/class_BCSmethod ##파일 경로 수정 필요
df_train_scaler.to_csv("df_train_scaler.csv") ##preprocessing한데이터.to_csv("저장할 파일 이름.csv")
df_test_scaler.to_csv("df_test_scaler.csv") ##preprocessing한데이터.to_csv("저장할 파일 이름.csv")
!cp df_train_scaler.csv "/content/drive/MyDrive/Colab/class_BCSmethod/" ##파일 저장할 경로
!cp df_test_scaler.csv "/content/drive/MyDrive/Colab/class_BCSmethod/" ##파일 저장할 경로
```

preprocessing한 데이터 저장한 이름으로 수정
→ 데이터 이름.to_csv("저장할 이름.csv")

2. PCA로 데이터의 군집과 분포 확인하기

- PCA의 의미
 - PCA는 signal/matrix를 분해(decompose)하는 방법으로 널리 이용되며, high-dimension 데이터를 차원 축소하여 통계적 특성을 보여주는 탐색적 분석 툴
- PCA tutorial
 - 예시 코드: 성별을 기준으로 나타낸 PCA 코드와 결과
 - [과제] 예시 코드를 참고하여 성별을 기준으로 나타낸 PCA 코드 결과 비교하고 의미 설명하기
 - 두가지 principal component 를 기준으로 성별이 구분되는가?
 - 두개의 component는 variance를 얼만큼 설명하는가?

2. PCA로 데이터의 군집과 분포 확인하기

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

from sklearn import decomposition
from sklearn import datasets

### data
X = train.iloc[:,2:]
y = train.iloc[:,1]
target_names = train.columns[1]

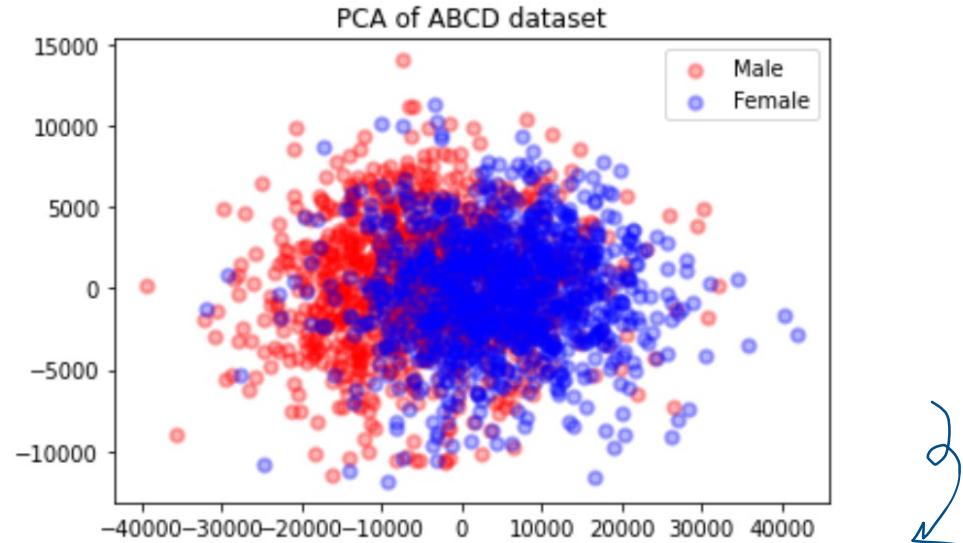
### PCA decomposition
pca = PCA(n_components=2)
X_r = pca.fit(X).transform(X)

# Percentage of variance explained for each components
print('explained variance ratio (first two components): %s'
      % str(pca.explained_variance_ratio_))

plt.figure()
colors = ['red', 'blue']
lw = 2

for color, i, target_name in zip(colors, [0, 1], target_names):
    plt.scatter(X_r[y == i, 0], X_r[y == i, 1], color=color, alpha=.2, lw=lw,
                label=target_name)
plt.legend(loc='best', shadow=False, scatterpoints=1, labels=['Male', 'Female'])
plt.title('PCA of ABCD dataset')
plt.show()

print(pca.explained_variance_ratio_)
```



★ PCA에 대해 공부한 후, 결과의 의미 기술하기

- 두 가지 principal component 를 기준으로 성별이 구분되는가?
- 두 개의 component는 variance를 얼만큼 설명하는가?

코드 수정하여 추가 분석해도 되지만,
결과의 의미만 기술해도 점수에는 영향 없음.

★ 3. Classification 실험하기

- <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html>
- <http://docs.h2o.ai/h2o-tutorials/latest-stable/>

1. H2O 패키지 설치하기
2. H2O 구동하기
3. H2O를 이용한 자동화된 기계학습 실험하기
4. 학습된 모델 성능 확인하기
5. 모델 설명하기

3-1) H2O 패키지 설치하기

H2O 사용을 위한 환경 세팅



```
! apt-get install default-jre  
! java -version  
! pip install h2o
```

3-2) H2O 구동하기

H2OAutoML 파이썬 모듈을 import 하고,
h2o.init()으로 H2O 구동 시작하기



```
import h2o
from h2o.automl import H2OAutoML
h2o.init()
```

H2OAutoML 파이썬 모듈 import

H2O 구동 시작하기

3-3) H2O를 이용한 자동화된 기계학습 실험

기계학습 실험하기

- Train / test set 나눠서 넣기
- classification task에서 y는 factor 처리
- 알고리즘 설정: exclude_algos = ["DRF", "GBM", "DeepLearning", "StackedEnsemble"]

<주의> 처음 데이터를 load 할 때 train, test 라고 저장한 것과
h2o 형식으로 불러온 파일을 train, test 라고 동일 이름으로 저장하면
에러가 발생할 수 있음.

```
### Import a sample binary outcome train/test set into H2O
train = h2o.import_file("https://s3.amazonaws.com/erin-data/higgs/higgs_train_10k.csv")
test = h2o.import_file("https://s3.amazonaws.com/erin-data/higgs/higgs_test_5k.csv")

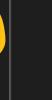
### Identify predictors and response
x = train.columns
y = "response"
x.remove(y)

### For binary classification, response should be a factor
train[y] = train[y].asfactor()
test[y] = test[y].asfactor() Classification을 위해 y를 categorical하게 코딩 (aka. "factor")

### Run AutoML (limited to 300 secs max runtime )
aml = H2OAutoML(exclude_algos = list({"GBM", "deeplearning", "StackedEnsemble", "DRF"}),
                 max_runtime_secs=300, seed=1)
aml.train(x=x, y=y, training_frame=train)

### View the AutoML Leaderboard
aml = H2OAutoML(max_models = 10, seed=1) → 10개 모델 후에 멈추기
aml = H2OAutoML(max_runtime_secs=300) → 300초 후에 멈추기
lb = aml.leaderboard
lb.head(rows=lb.nrows) # Print all rows instead of default (10 rows)
```

실험 코드 제출



시간이 오래 걸리면,
aml = H2OAutoML(max_models = 10, seed=1) → 10개 모델 후에 멈추기
aml = H2OAutoML(max_runtime_secs=300) → 300초 후에 멈추기

| A | B | C | D | E | F | G | H | I | J | K |
|------------|-------------|-------------|------------|------------|------------|------------|-------------|------------|------------|-------------|
| 1 Response | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 | x10 |
| 2 0 | 0.623878 | 0.4596591 | -0.3165119 | 1.55700946 | 0.64478427 | 0.45592913 | -2.2498279 | -0.3619488 | 0 | 0.83297758 |
| 3 0 | 1.68203664 | 0.77425104 | 1.32875454 | 0.23472911 | -0.5578297 | 0.44292089 | -0.0832048 | 1.08119273 | 2.17307615 | 1.1458478 |
| 4 1 | 1.09402835 | -0.8873397 | 0.94922942 | 0.41026053 | -1.5586771 | 0.5235203 | 0.13167513 | 0.30451295 | 0 | 1.11915624 |
| 5 0 | 1.3387115 | -0.8990273 | -1.25534 | 0.88428509 | -0.7479247 | 0.5474447 | -0.79518 | 0.17423473 | 2.17307615 | 0.71162253 |
| 6 0 | 0.69909477 | 1.43362796 | -1.7225332 | 0.65011257 | -0.1675736 | 0.96746385 | 0.73769587 | -0.8863906 | 2.17307615 | 1.66796947 |
| 7 1 | 0.84220785 | -0.1442015 | -1.7020034 | 0.68858754 | 0.54433763 | 0.57877439 | -0.221837 | -0.5864736 | 1.08653808 | 0.909401 |
| 8 1 | 0.45624167 | 1.74042809 | 0.25842708 | 1.45223081 | -0.0513728 | 0.67532843 | -0.7971605 | -1.347631 | 2.17307615 | 0.56858021 |
| 9 0 | 0.93371242 | 1.35084057 | 0.77333838 | 0.53263807 | -0.6795369 | 0.71801734 | 0.74561769 | -1.4962034 | 1.08653808 | 0.27421781 |
| 10 0 | 0.73350048 | -0.9983721 | -1.6870221 | 0.95116895 | -0.5917794 | 1.14912093 | -0.9427243 | -0.4240388 | 2.17307615 | 1.0418514 |
| 11 0 | 0.9681813 | 1.7190082 | 0.03093395 | 0.14912093 | -0.867323 | 0.6313569 | -0.8100336 | 1.2064817 | 0 | 0.8869308 |
| 12 1 | 1.87749016 | -0.3652925 | -0.4291487 | 0.71057469 | -0.3277389 | 0.8470003 | 0.0643399 | 0.79957008 | 0 | 0.92803466 |
| 13 1 | 1.82313669 | 0.33596507 | 0.84214115 | 0.25860361 | 1.62257099 | 1.97651732 | 0.98030221 | -0.445105 | 2.17307615 | 1.03203094 |
| 14 1 | 0.62003481 | -0.5785916 | -1.3180393 | 0.50328153 | 0.935413 | 1.03745174 | -0.6931863 | -0.1352093 | 2.17307615 | 1.2872379 |
| 15 0 | 0.87734586 | 0.88138753 | -1.6204387 | 1.40157483 | 0.33585978 | 0.28874594 | -0.3389312 | -0.7134255 | 0 | 0.68969935 |
| 16 1 | 1.25635743 | 0.34180889 | 1.14786983 | 0.15779532 | -0.7769958 | 1.38464308 | 0.15544066 | -0.3175988 | 2.17307615 | 0.86734921 |
| 17 0 | 0.34222701 | 0.82781935 | 1.35371785 | 1.07669044 | -0.7612171 | 0.39354458 | 0.61193663 | -0.5085888 | 1.08653808 | 0.820553808 |
| 18 0 | 0.44910434 | 1.587515 | 1.28492057 | 1.12922406 | -1.4708807 | 0.58885115 | -0.764483 | -0.6629774 | 0 | 0.68831438 |
| 19 1 | 0.83543652 | 1.08299911 | -1.3968296 | 0.63259619 | 0.97038677 | 0.89601016 | 0.06136882 | -0.4085163 | 2.17307615 | 0.82038724 |
| 20 1 | 1.10043371 | -0.3818499 | -0.3597911 | 0.8308442 | 0.28105021 | 0.98010403 | -0.8427111 | 1.59010935 | 1.08653808 | 0.54100734 |
| 21 1 | 1.00545192 | 0.31648567 | -1.4922658 | 0.85141695 | 0.50674438 | 0.84516823 | -0.6268165 | -0.4356807 | 2.17307615 | 0.84619743 |
| 22 1 | 1.38926035 | -1.4356841 | 0.58302093 | 1.60137939 | -1.3866142 | 1.7413615 | -2.1319907 | 0.46971679 | 0 | 1.68534422 |
| 23 0 | 1.07792354 | -0.6469613 | -0.1822354 | 1.23644653 | -1.1063251 | 1.10982144 | -0.7011082 | 1.19297478 | 1.08653808 | 0.78500831 |
| 24 1 | 1.2404356 | -0.046637 | -0.0845798 | 1.16198218 | -0.9064676 | 1.09195805 | 0.22079584 | 1.23364592 | 2.17307615 | 0.95434856 |
| 25 0 | 0.79279841 | -2.3083601 | 1.25051916 | 0.94266156 | 1.49298823 | 0.66964877 | 1.0763545 | -0.6258342 | 2.17307615 | 0.65520179 |
| 26 0 | 0.375161865 | -1.0314871 | -0.2427159 | 1.5408349 | -1.3677828 | 0.80092192 | -1.7883806 | -1.3032809 | 0 | 0.78198665 |
| 27 0 | 0.91248333 | -1.4493197 | -1.4623032 | 0.85770816 | 1.30911493 | 0.61715782 | -0.980582 | 0.03009714 | 0 | 0.8317185 |
| 28 1 | 1.78525376 | -0.8649384 | -1.1083083 | 0.39643604 | 1.142874 | 0.56310958 | -0.41317052 | 0 | 0.26668239 | |
| 29 1 | 0.70202291 | -0.8103962 | 1.65889704 | 1.62776387 | -0.2243729 | 1.468647 | 0.26436594 | 1.40273046 | 1.08653808 | 0.88686424 |
| 30 1 | 0.93828762 | 0.9942014 | -0.196107 | 1.14514434 | -0.5007413 | 0.54946011 | -0.4684042 | 0.90102077 | 2.17307615 | 1.09032428 |
| 31 1 | 0.40811026 | 0.81028789 | 0.49368829 | 1.13511145 | -0.9397735 | 1.41212523 | 1.55166495 | 1.43432975 | 0 | 1.78971815 |
| 32 0 | 0.37754774 | -1.0139556 | -0.2843299 | 1.61541188 | -0.3336324 | 0.75237006 | 0.32873088 | 0.79735255 | 2.17307615 | 1.59318292 |
| 33 1 | 0.62863618 | 1.2534436 | 0.47371328 | 0.4057285 | -1.0705006 | 1.73916304 | 0.30760303 | 0.27956605 | 0 | 1.64203334 |
| 34 1 | 1.10757399 | 0.39148128 | 1.19447804 | 1.24501324 | 0.42861566 | 0.66552645 | 0.12771422 | -1.5472076 | 1.08653808 | 0.55880432 |
| 35 1 | 2.42615175 | 0.5755614 | -0.1250847 | 0.32140562 | 0.2375818 | 2.2342999 | 2.11906672 | -1.2678008 | 0 | 1.04097009 |
| 36 1 | 0.99373937 | -0.26107178 | 1.31876695 | 0.5597527 | 0.08053499 | 1.11495137 | 0.06334928 | -0.0620317 | 2.17307615 | 1.30058312 |
| 37 0 | 1.02960908 | 0.76353735 | 1.17616773 | 1.47258008 | 0.43636188 | 1.22597945 | -1.5388427 | 1.62669814 | 0 | 2.11631179 |
| 38 0 | 0.32905033 | 0.22103676 | -0.4668793 | 0.81319373 | 0.22772713 | 1.19730628 | 1.94280577 | 0.64545375 | 0 | 2.59764099 |
| 39 1 | 0.46191496 | -2.2236247 | -0.0030154 | 0.86322111 | 1.69592607 | 0.62366194 | -0.8367698 | -0.6635318 | 0 | 0.64928436 |

3-3) H2O를 이용한 자동화된 기계학습 실험

| Parse progress: | 100% | | | | | | |
|--|----------|----------|----------|----------------------|----------|----------|----------|
| Parse progress: | 100% | | | | | | |
| AutoML progress: | 100% | | | | | | |
| | | | | | | | |
| model_id | auc | logloss | aucpr | mean_per_class_error | rmse | mse | |
| XGBoost_grid_1_AutoML_20210604_001648_model_5 | 0.778913 | 0.561461 | 0.797798 | | 0.339159 | 0.437196 | 0.191141 |
| XGBoost_grid_1_AutoML_20210604_001648_model_22 | 0.774082 | 0.567147 | 0.791927 | | 0.339063 | 0.439875 | 0.19349 |
| XGBoost_grid_1_AutoML_20210604_001648_model_16 | 0.773814 | 0.56618 | 0.792684 | | 0.328456 | 0.439536 | 0.193192 |
| XGBoost_grid_1_AutoML_20210604_001648_model_18 | 0.772823 | 0.567167 | 0.790262 | | 0.337952 | 0.440022 | 0.193619 |
| XGBoost_grid_1_AutoML_20210604_001648_model_14 | 0.771794 | 0.574107 | 0.79139 | | 0.340541 | 0.442476 | 0.195785 |
| XGBoost_3_AutoML_20210604_001648 | | 0.765302 | 0.57678 | 0.784152 | 0.334265 | 0.444144 | 0.197264 |
| XGBoost_grid_1_AutoML_20210604_001648_model_10 | 0.763923 | 0.600666 | 0.784465 | | 0.337997 | 0.451546 | 0.203894 |
| XGBoost_grid_1_AutoML_20210604_001648_model_19 | 0.761751 | 0.608778 | 0.781994 | | 0.34474 | 0.454201 | 0.206298 |
| XGBoost_grid_1_AutoML_20210604_001648_model_9 | 0.761366 | 0.594626 | 0.777956 | | 0.34441 | 0.449949 | 0.202454 |
| XGBoost_grid_1_AutoML_20210604_001648_model_15 | 0.760704 | 0.600787 | 0.778721 | | 0.331794 | 0.451946 | 0.204256 |
| XGBoost_grid_1_AutoML_20210604_001648_model_20 | 0.757116 | 0.69125 | 0.775699 | | 0.356894 | 0.470757 | 0.221612 |
| XGBoost_grid_1_AutoML_20210604_001648_model_21 | 0.756757 | 0.609084 | 0.773028 | | 0.365386 | 0.454995 | 0.20702 |
| XGBoost_grid_1_AutoML_20210604_001648_model_8 | 0.756755 | 0.598923 | 0.77612 | | 0.353988 | 0.452375 | 0.204643 |
| XGBoost_grid_1_AutoML_20210604_001648_model_11 | 0.756038 | 0.60387 | 0.772726 | | 0.369826 | 0.453544 | 0.205702 |
| XGBoost_2_AutoML_20210604_001648 | | 0.755459 | 0.60549 | 0.772642 | 0.33953 | 0.454056 | 0.206167 |
| XGBoost_grid_1_AutoML_20210604_001648_model_7 | 0.754155 | 0.620028 | 0.77306 | | 0.361099 | 0.458726 | 0.210429 |
| XGBoost_grid_1_AutoML_20210604_001648_model_1 | 0.748807 | 0.64848 | 0.767313 | | 0.371905 | 0.465754 | 0.216927 |
| XGBoost_grid_1_AutoML_20210604_001648_model_3 | 0.747634 | 0.624482 | 0.759859 | | 0.353408 | 0.459866 | 0.211476 |
| XGBoost_grid_1_AutoML_20210604_001648_model_13 | 0.747495 | 0.626118 | 0.762367 | | 0.367984 | 0.460913 | 0.21244 |
| XGBoost_grid_1_AutoML_20210604_001648_model_17 | 0.747303 | 0.610972 | 0.764303 | | 0.358037 | 0.457075 | 0.208917 |
| XGBoost_1_AutoML_20210604_001648 | | 0.745481 | 0.621079 | 0.759972 | 0.359974 | 0.45988 | 0.21149 |
| XGBoost_grid_1_AutoML_20210604_001648_model_12 | 0.741079 | 0.642451 | 0.756715 | | 0.364094 | 0.465674 | 0.216853 |
| XGBoost_grid_1_AutoML_20210604_001648_model_6 | 0.740729 | 0.652532 | 0.758086 | | 0.383713 | 0.468185 | 0.219197 |
| XGBoost_grid_1_AutoML_20210604_001648_model_4 | 0.739667 | 0.708107 | 0.756176 | | 0.382851 | 0.478581 | 0.22904 |
| XGBoost_grid_1_AutoML_20210604_001648_model_2 | 0.738971 | 0.644534 | 0.755686 | | 0.367957 | 0.466896 | 0.217992 |
| GLM_1_AutoML_20210604_001648 | | 0.682648 | 0.63852 | 0.680715 | 0.397234 | 0.472683 | 0.223429 |

실험 완료 후 결과

3-4) 학습된 모델 성능 확인하기

Leaderboard

- model_id: hyperparameter 바꿔가며 최적의 모델 performance 보여주는 것
- 학습된 모델 성능을 나타내는 metric을 기준으로 모델 비교하기

실험 완료 후

```
lb = h2o.automl.get_leaderboard(aml, extra_columns = 'ALL')  
lb
```

★ 각 metric에 대해 공부한 후 모델 예측력 비교 및 근거 제시

| model_id | auc | logloss | aucpr | mean_per_class_error | rmse | mse | training_time_ms | predict_time_per_row_ms | algo |
|--|----------|----------|----------|----------------------|----------|----------|------------------|-------------------------|--------------------------|
| StackedEnsemble_AllModels_AutoML_20210514_062212 | 0.777452 | 0.56232 | 0.795709 | | 0.319729 | 0.437611 | 0.191504 | 2040 | 0.023838 StackedEnsemble |
| XGBoost_3_AutoML_20210514_062212 | 0.765302 | 0.57678 | 0.784152 | | 0.334265 | 0.444144 | 0.197264 | 2967 | 0.008744 XGBoost |
| XGBoost_2_AutoML_20210514_062212 | 0.755459 | 0.60549 | 0.772642 | | 0.33953 | 0.454056 | 0.206167 | 3961 | 0.010459 XGBoost |
| XGBoost_1_AutoML_20210514_062212 | 0.745481 | 0.621079 | 0.759972 | | 0.359974 | 0.45988 | 0.21149 | 2452 | 0.010434 XGBoost |

ML Metric

- Classification
 - Auc
 - Logloss
 - mean_per_class_error
- Regression
 - Rmse
 - Mse

3-5) 모델 설명하기



각 metric에 대해 공부한 후 예측 모델 해석 및 근거 제시

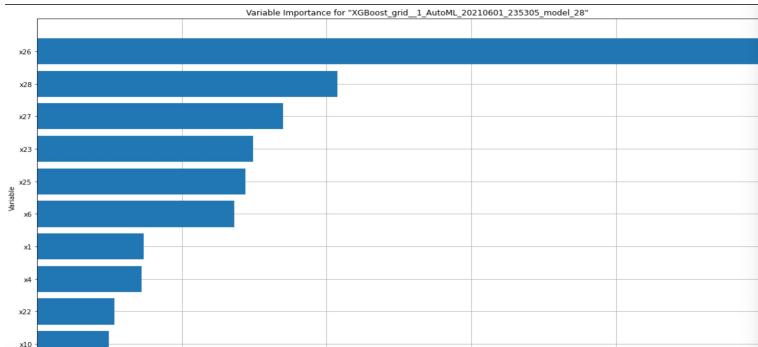
```
exa = aml.explain(test)
```

- Confusion matrix shows a predicted class VS. an actual class

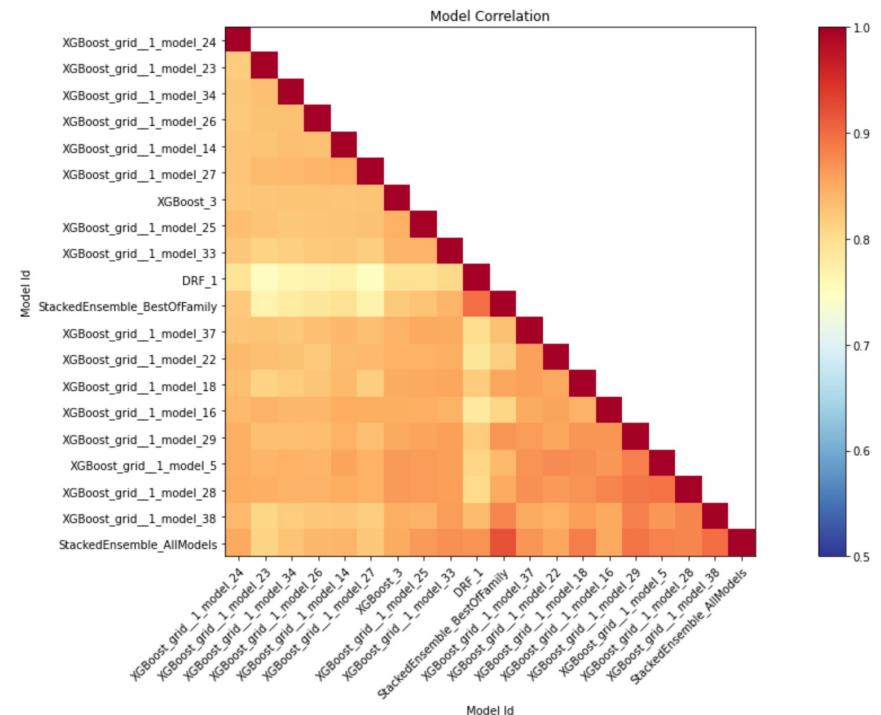
StackedEnsemble_AllModels_AutoML_20210601_235305

| Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.4910814205869044: | | | | |
|--|-------|----------------|---------------|-----------------|
| | 0 | 1 | Error | Rate |
| 0 | 0 | 4487.0 | 218.0 | 0.0463 |
| 1 | 1 | 95.0 | 5200.0 | 0.0179 |
| 2 | Total | 4582.0 | 5418.0 | 0.0313 |
| | | (218.0/4705.0) | (95.0/5295.0) | (313.0/10000.0) |

- Variable importance shows the relative importance of the most important variable in the model



- Model correlation shows the correlation between the predictions of the models.



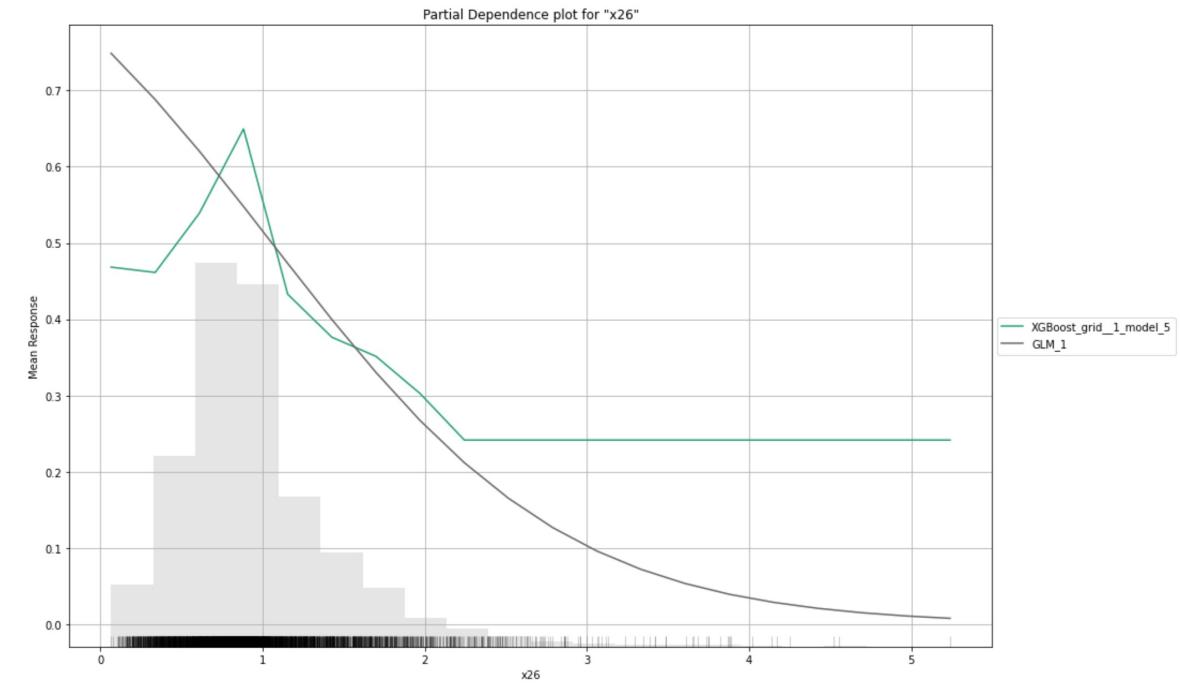
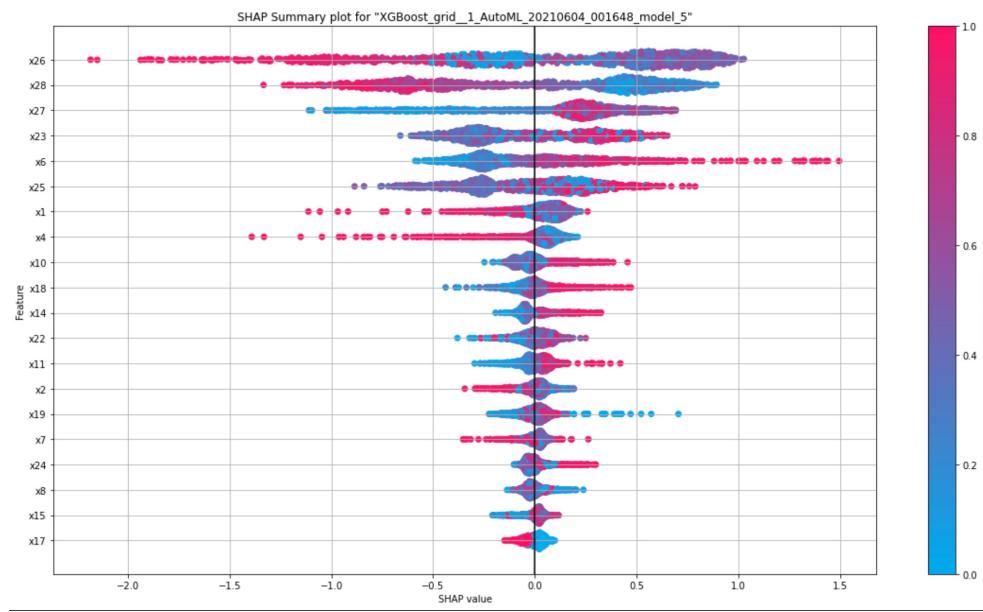
3-5) 모델 설명하기



각 metric에 대해 공부한 후 예측 모델 해석 및 근거 제시

```
exa = aml.explain(test)
```

- **SHAP summary plot** shows the contribution of the features for each instance
- **Partial dependence plot** gives a graphical depiction of the marginal effect of a variable on the response



과제

1. Preprocessing 전후 데이터로 실험 결과 비교하기

- [과제 설명] (1) Standard Scaler, (2) Power Transformer 의 전처리 방법으로 각각 classification 실험하고 결과의 의미 설명하기
- 참고 사이트: <https://scikit-learn.org/stable/modules/preprocessing.html>

2. PCA로 데이터의 군집과 분포 확인하기

- [과제 설명] 예시 코드를 참고하여 성별을 기준으로 나타낸 PCA 코드 결과 비교하고 의미 설명하기
 - (1) 두가지 principal component 를 기준으로 성별이 구분되는가?
 - (2) 두개의 component는 variance를 얼만큼 설명하는가?
- 참고 사이트: https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_vs_lda.html

3. Sex Classification 실험하기: 선형모델(GLM)과 기계학습(ML) 비교하기

- [과제 설명] 선형 모델(GLM)과 xgboost를 포함하는 ML 모델의 예측력 비교 및 논의하기
 - (1) 모델 비교: 어떤 모델이 예측력이 좋았는가? 그렇게 생각한 이유는 무엇인가?
 - (2) 모델 해석(machine learning interpretation): 어떤 변수가 예측에 중요하였는가? 그렇게 생각한 이유는 무엇인가?

4. 본인의 연구에 기계학습 분석이 어떻게 적용될 수 있을지 간략하게 기술하기