A.I.D.D. 2021

'2021 인공지능 학습용 데이터 구축사업'을 통해 구축된

인공지능 학습용 데이터를 이용한 당뇨병 발병 예측 모델 개발 챌린지

경희의료원

데이터 및 코드 소개

1.Overview

A.I.D.D. 2021

- **목표**

✓ 「2021 인공지능 학습용 데이터 구축사업 - 당뇨병 추적 관찰 데이터」를 통해 확보된 건강검진 데이터를 활용해 향후 당뇨병 발병 여부를 예측하는 모델 구축

- **배경**

✓ 당뇨병 : 혈중 당 농도를 조절하는 신체 기능의 이상으로 고혈당 상태가 지속되면서 이로 인한 여러 증상 및 합병증이 발생하게 되는 만성질환

✓ 당뇨병은 장기간에 걸쳐 환자의 삶의 질을 크게 떨어뜨릴 뿐 만 아니라 심뇌혈관질환, 망막질환, 신장질환 등의 심각한 합병증 및 사망으로 이어질 수 있어 예방 및 조기 진단을 통한 적절한 개입이 매우 중요

*** 개인의 당뇨병 발병 여부 사전 예측 가능 시,**
조기에 당뇨병의 위험을 알리고 생활습관 개선을 유도하는 등 당뇨병 예방 및 치료 가능

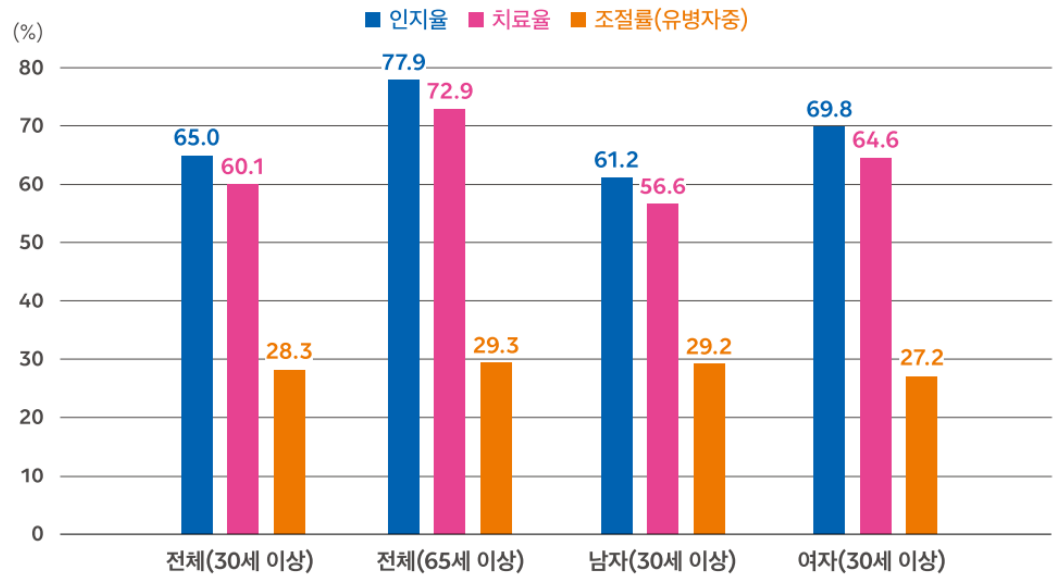
1.Overview

A.I.D.D. 2021

- ## 당뇨병 관리 수준 (2016-2018년 통합)

당뇨병을 가진 성인 10명 중 6~7명만이 당뇨병을 가진 것을 알고 있었고,
치료를 받고 있는 경우는 10명 중 6명이었으며, 10명 중 3명만이 당화혈색소 6.5% 미만이었다.

- 당뇨병 위험 인식 부족
- 당뇨병 인지 : 10명 중 6~7명



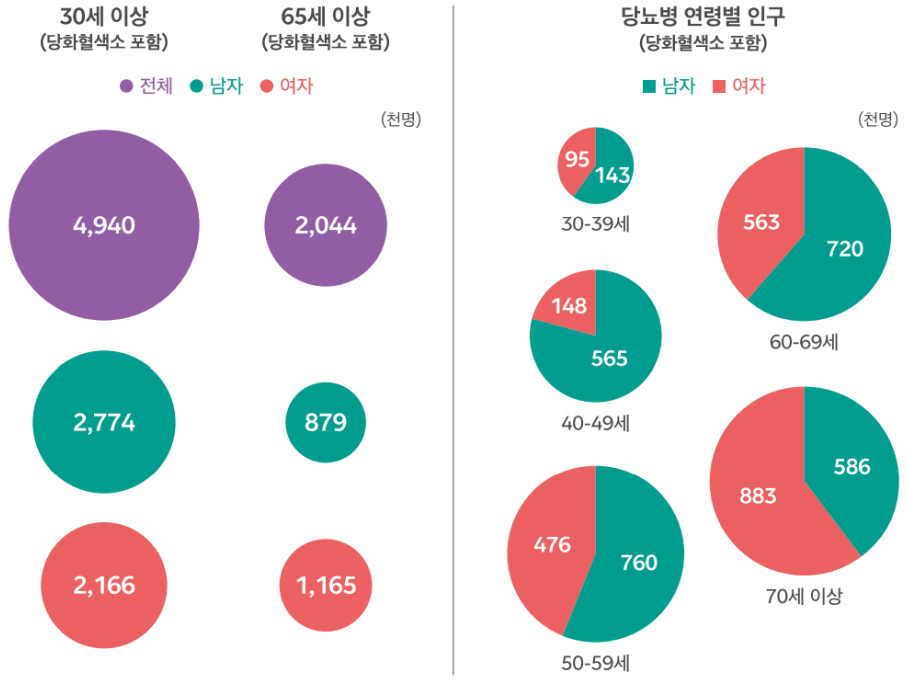
출처 : DIABETES FACT SHEET IN KOREA,2020

1.Overview

A.I.D.D. 2021

- ## 당뇨병 인구 (2018년)

30세 이상 성인 중 당뇨병을 가진 사람은 494만 명으로 추정되었다.
30대 24만 명, 40대 71만 명이 당뇨병이 있었다.



· **추정방법** 2018년 추계 인구 적용

- 30세 이상 성인 당뇨병 발병
- 7명 중 1명

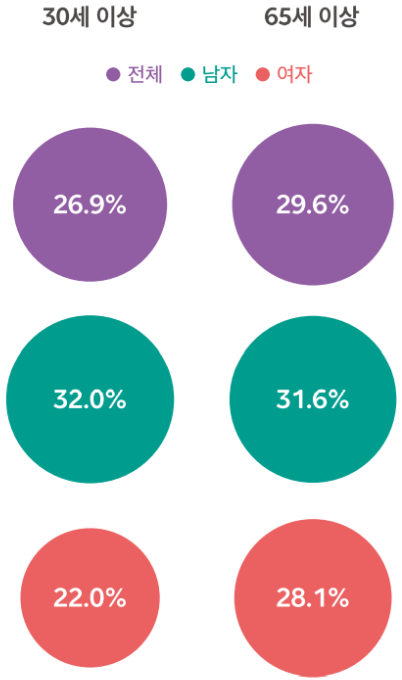
출처 : DIABETES FACT SHEET IN KOREA,2020

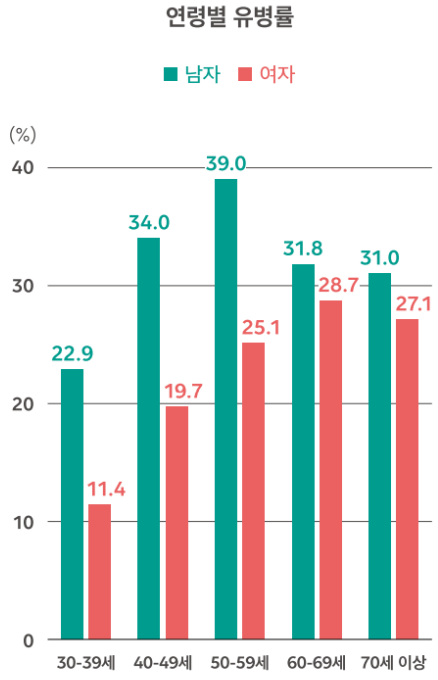
1.Overview

A.I.D.D. 2021

- ## 공복혈당장애 유병률 (2018년)

30세 이상 성인 약 4명 중 1명(26.9%)이 공복혈당장애를 가지고 있다.
65세 이상 성인에서는 10명 중 3명(29.6%)이다.





- 공복 혈당 장애
  (성인 당뇨병 전 단계)
- 30대 성인 4명중 1명

출처 : DIABETES FACT SHEET IN KOREA,2020

2.Data Information

A.I.D.D. 2021

- **DATA 설명**

✓ 제공되는 건강검진 데이터는 22개의 임상 변수와, baseline 및 endpoint 시점의 건강검진 일자 변수 2개로 구성

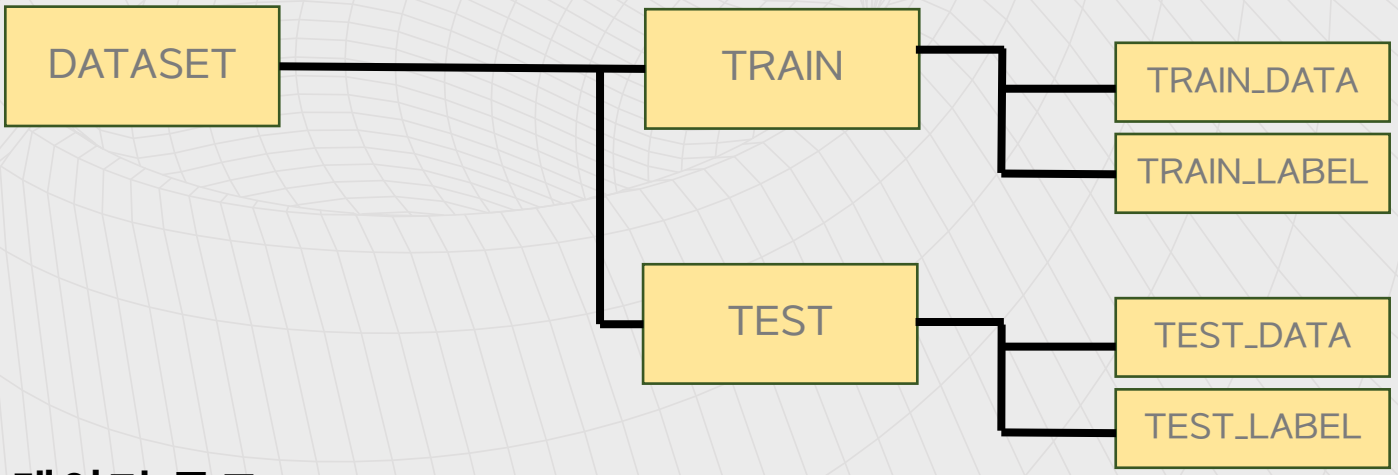
번호	변수명	설명
01	gender	성별
02	age	나이
03	**date**	**Baseline 건강검진 일자**
04	Ht	신장
05	Wt	체중
06	BMI	체질량지수(BMI)
07	SBP	수축기혈압
08	DBP	이완기혈압
09	PR	맥박
10	**HbA1c**	**당화혈색소**
11	**FBG**	**공복혈당**
12	TC	총콜레스테롤

번호	변수명	설명
13	TG	중성지방
14	LDL	LDL 콜레스테롤
15	HDL	HDL 콜레스테롤
16	Alb	알부민
17	BUN	혈중요소질소
18	Cr	크레아티닌
19	CrCl	크레아티닌 청소율
20	AST	아스파테이트아미노전이효소
21	ALT	알라닌아미노전이효소
22	GGT	감마글루타밀전이효소
23	ALP	알칼리인산분해효소
24	**date_E**	**Endpoint 건강검진 일자**

당뇨병 진단 기준: (1) 공복혈당(FBG) 126 mg/dL 이상, 또는 (2) 당화혈색소(HbA1c) 6.5% 이상

2.Data Information

- **예선 대회용 데이터 디렉토리 구조**



- **데이터 규모**

✓ 예선 데이터셋 환자 2,869명 추적 관찰 데이터 (신규 당뇨 발병자 242명, 약 8.4%)

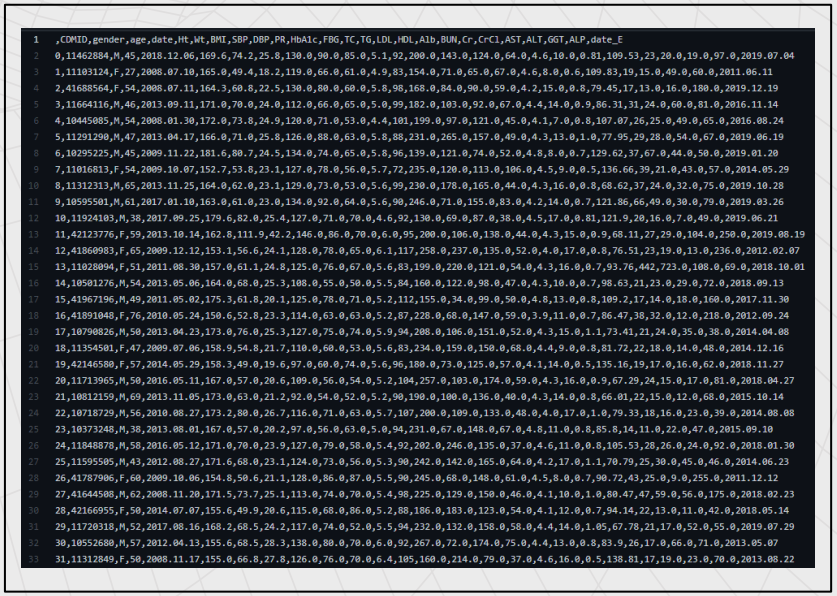
✓ 본선 데이터셋 환자 7,172명 추적 관찰 데이터 (신규 당뇨 발병자 606명, 약 8.4%)

2.Data Information

A.I.D.D. 2021

- **대회용 학습 데이터**

✓ 제공되는 건강검진 데이터는 22개의 임상 변수와, baseline 및 endpoint 시점의 건강검진 일자 변수 2개로 구성되어 있습니다.
✓ 24종의 건강검진 데이터가 CSV(Comma Separated Values) 형식으로 제공됩니다.



```
,COMID,gender,age,date,Ht,Wt,BMI,SBP,DBP,PR,HbA1c,FBG,TC,TG,LDL,HDL,Alb,BUN,Cr,CrCl,AST,ALT,GGT,ALP,date_E
0,11462884,M,45,2018.12.06,169.6,74.2,25.8,130.0,90.0,85.0,5.1,92,200.0,143.0,124.0,64.0,4.6,10.0,0.81,109.53,23,20.0,19.0,97.0,2019.07.04
1,11103124,F,27,2008.07.10,165.0,49.4,18.2,119.0,66.0,61.0,4.9,83,154.0,71.0,65.0,67.0,4.6,8.0,0.6,109.83,19,15.0,49.0,60.0,2011.06.11
2,41688564,F,54,2008.07.11,164.3,60.8,22.5,130.0,80.0,60.0,5.8,98,168.0,84.0,90.0,59.0,4.2,15.0,0.8,79.45,17,13.0,16.0,180.0,2019.12.19
3,11664116,M,46,2013.09.11,171.0,70.0,24.0,112.0,66.0,65.0,5.0,99,182.0,103.0,92.0,67.0,4.4,14.0,0.9,86.31,31,24.0,60.0,81.0,2016.11.14
4,10445085,M,54,2008.01.30,172.0,73.8,24.9,120.0,71.0,53.0,4.4,101,199.0,97.0,121.0,45.0,4.1,7.0,0.8,107.07,26,25.0,49.0,65.0,2016.08.24
5,11291290,M,47,2013.04.17,166.0,71.0,25.8,126.0,88.0,63.0,5.8,88,231.0,265.0,157.0,49.0,4.3,13.0,1.0,77.95,29,28.0,54.0,67.0,2019.06.19
6,10295225,M,45,2009.11.22,181.6,80.7,24.5,134.0,74.0,65.0,5.8,96,139.0,121.0,74.0,52.0,4.8,8.0,0.7,129.62,37,67.0,44.0,50.0,2019.01.20
7,11016813,F,54,2009.10.07,152.7,53.8,23.1,127.0,78.0,56.0,5.7,72,235.0,120.0,113.0,106.0,4.5,9.0,0.5,136.66,39,21.0,43.0,57.0,2014.05.29
8,11312313,M,65,2013.11.25,164.0,62.0,23.1,129.0,73.0,53.0,5.6,99,230.0,178.0,165.0,44.0,4.3,16.0,0.8,68.62,37,24.0,32.0,75.0,2019.10.28
9,10595501,M,61,2017.01.10,163.0,61.0,23.0,134.0,92.0,64.0,5.6,90,246.0,71.0,155.0,83.0,4.2,14.0,0.7,121.86,66,49.0,30.0,79.0,2019.03.26
10,11924103,M,38,2017.09.25,179.6,82.0,25.4,127.0,71.0,70.0,4.6,92,130.0,69.0,87.0,38.0,4.5,17.0,0.81,121.9,20,16.0,7.0,49.0,2019.06.21
11,42123776,F,59,2013.10.14,162.8,111.9,42.2,146.0,86.0,70.0,6.0,95,200.0,106.0,138.0,44.0,4.3,15.0,0.9,68.11,27,29.0,104.0,250.0,2019.08.19
12,41860983,F,65,2009.12.12,153.1,56.6,24.1,128.0,78.0,65.0,6.1,117,258.0,237.0,135.0,52.0,4.0,17.0,0.8,76.51,23,19.0,13.0,236.0,2012.02.07
13,11028094,F,51,2011.08.30,157.0,61.1,24.8,125.0,76.0,67.0,5.6,83,199.0,220.0,121.0,54.0,4.3,16.0,0.7,93.76,442,723.0,108.0,69.0,2018.10.01
14,10501276,M,54,2013.05.06,164.0,68.0,25.3,108.0,55.0,50.0,5.5,84,160.0,122.0,98.0,47.0,4.3,10.0,0.7,98.63,21,23.0,29.0,72.0,2018.09.13
15,41967196,M,49,2011.05.02,175.3,61.8,20.1,125.0,78.0,71.0,5.2,112,155.0,34.0,99.0,50.0,4.8,13.0,0.8,109.2,17,14.0,18.0,160.0,2017.11.30
16,41891048,F,76,2010.05.24,150.6,52.8,23.3,114.0,63.0,63.0,5.2,87,228.0,68.0,147.0,59.0,3.9,11.0,0.7,86.47,38,32.0,12.0,218.0,2012.09.24
17,10790826,M,50,2013.04.23,173.0,76.0,25.3,127.0,75.0,74.0,5.9,94,208.0,106.0,151.0,52.0,4.3,15.0,1.1,73.41,21,24.0,35.0,38.0,2014.04.08
18,11354501,F,47,2009.07.06,158.9,54.8,21.7,110.0,60.0,53.0,5.6,83,234.0,159.0,150.0,68.0,4.4,9.0,0.8,81.72,22,18.0,14.0,48.0,2014.12.16
19,42146580,F,57,2014.05.29,158.3,49.0,19.6,97.0,60.0,74.0,5.6,96,180.0,73.0,125.0,57.0,4.1,14.0,0.5,135.16,19,17.0,16.0,62.0,2018.11.27
20,11713965,M,50,2016.05.11,167.0,57.0,20.6,109.0,56.0,54.0,5.2,104,257.0,103.0,174.0,59.0,4.3,16.0,0.9,67.29,24,15.0,17.0,81.0,2018.04.27
21,10812159,M,69,2013.11.05,173.0,63.0,21.2,92.0,54.0,52.0,5.2,90,190.0,100.0,136.0,40.0,4.3,14.0,0.8,66.01,22,15.0,12.0,68.0,2015.10.14
22,10718729,M,56,2010.08.27,173.2,80.0,26.7,116.0,71.0,63.0,5.7,107,200.0,109.0,133.0,48.0,4.0,17.0,1.0,79.33,18,16.0,23.0,39.0,2014.08.08
23,10373248,M,38,2013.08.01,167.0,57.0,20.2,97.0,56.0,63.0,5.0,94,231.0,67.0,148.0,67.0,4.8,11.0,0.8,85.8,14,11.0,22.0,47.0,2015.09.10
24,11848878,M,58,2016.05.12,171.0,70.0,23.9,127.0,79.0,58.0,5.4,92,202.0,246.0,135.0,37.0,4.6,11.0,0.8,105.53,28,26.0,24.0,92.0,2018.01.30
25,11595505,M,43,2012.08.27,171.6,68.0,23.1,124.0,73.0,56.0,5.3,90,242.0,142.0,165.0,64.0,4.2,17.0,1.1,70.79,25,30.0,45.0,46.0,2014.06.23
26,41787906,F,60,2009.10.06,154.8,50.6,21.1,128.0,86.0,87.0,5.5,90,245.0,68.0,148.0,61.0,4.5,8.0,0.7,90.72,43,25.0,9.0,255.0,2011.12.12
27,41644508,M,62,2008.11.20,171.5,73.7,25.1,113.0,74.0,70.0,5.4,98,225.0,129.0,150.0,46.0,4.1,10.0,1.0,80.47,47,59.0,56.0,175.0,2018.02.23
28,42166955,F,50,2014.07.07,155.6,49.9,20.6,115.0,68.0,86.0,5.2,88,186.0,183.0,123.0,54.0,4.1,12.0,0.7,94.14,22,13.0,11.0,42.0,2018.05.14
29,11720318,M,52,2017.08.16,168.2,68.5,24.2,117.0,74.0,52.0,5.5,94,232.0,132.0,158.0,58.0,4.4,14.0,1.05,67.78,21,17.0,52.0,55.0,2019.07.29
30,10552680,M,57,2012.04.13,155.6,68.5,28.3,138.0,80.0,70.0,6.0,92,267.0,72.0,174.0,75.0,4.4,13.0,0.8,83.9,26,17.0,66.0,71.0,2013.05.07
31,11312849,F,50,2008.11.17,155.0,66.8,27.8,126.0,76.0,70.0,6.4,105,160.0,214.0,79.0,37.0,4.6,16.0,0.5,138.81,17,19.0,23.0,70.0,2013.08.22
```

학습 데이터 예시

2.Data Information

- **대회용 학습 데이터 라벨**

✓ 학습 데이터의 순서대로 당뇨병 미발병의 경우 0, 발병의 경우 1로 제공됩니다.

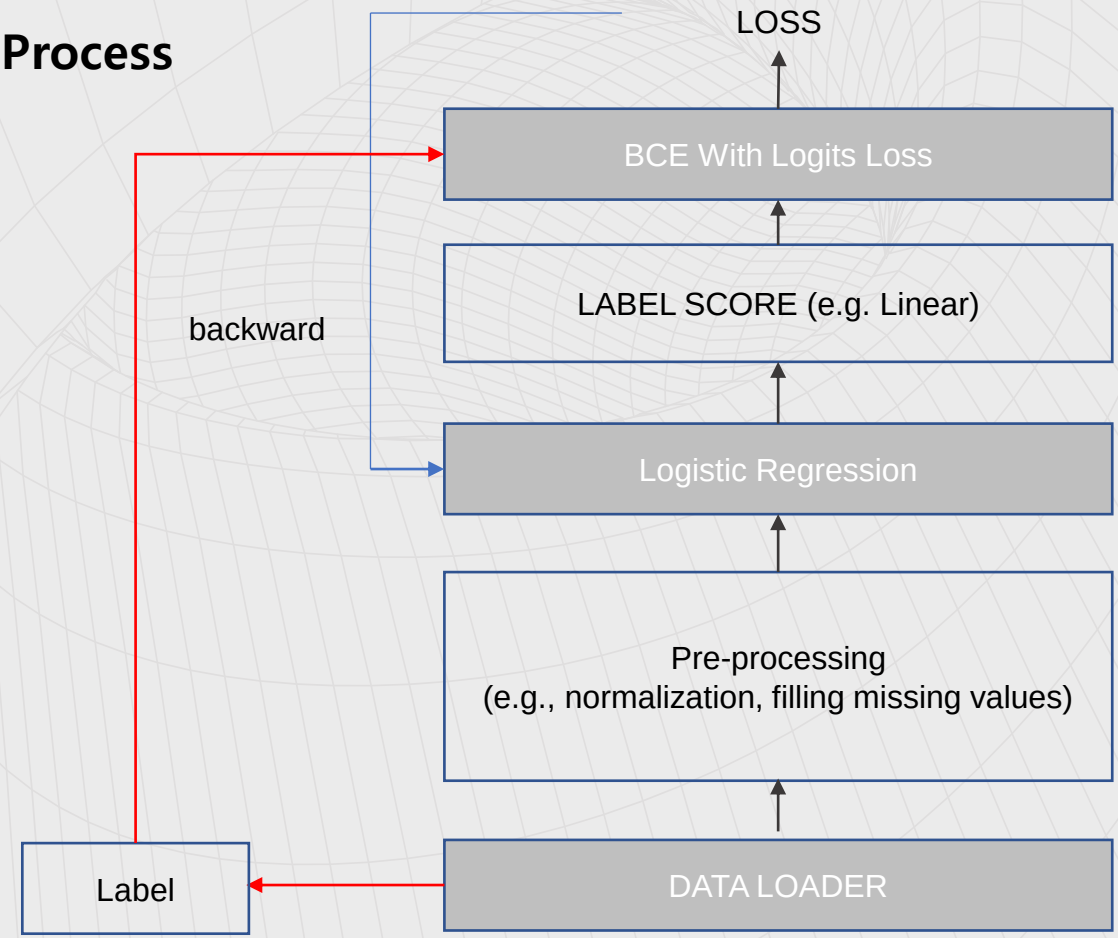
1	0	32	1	64	0
2	0	33	0	65	0
3	0	34	0	66	0
4	0	35	0	67	0
5	0	36	0	68	0
6	0	37	0	69	0
7	0	38	0	70	0
8	0	39	1	71	0
9	0	40	0	72	0
10	0	41	0	73	0
11	0	42	1	74	0
12	0	43	0	75	0
13	0	44	0	76	0
14	0	45	0	77	0
15	0	46	0	78	0
16	1	47	0	79	0
17	0	48	0	80	0
18	0	49	0	81	0
19	0	50	0	82	0

```
{
  “당뇨 미발병”: 0,
  “당뇨 발병” : 1
}
```

학습 데이터 라벨 예시

3.Baseline Code

- **Process**



3.Baseline Code

A.I.D.D. 2021

- 데이터 전처리

```python
def preproc_data(data, label=None, train=True, val_ratio=0.2, seed=1234):
    if train:
        dataset = dict()

        # NaN 값 0으로 채우기
        data = data.fillna(0)

        # 성별 ['M', 'F'] -> [0, 1]로 변환
        data['gender_enc'] = np.where(data['gender'] == 'M', 0, 1)

        # 날짜 datetime으로 변환
        # df.loc[:, 'date'] = pd.to_datetime(df['date'], format='%Y%m%d')

        DROP_COLS = ['CDMID', 'gender', 'date', 'date_E']
        X = data.drop(columns=DROP_COLS).copy()
        y = label

        X_train, X_val, y_train, y_val = train_test_split(X, y,
                                                          stratify=y,
                                                          test_size=val_ratio,
                                                          random_state=seed,
                                                          )

        X_train = torch.as_tensor(X_train.values).float()
        y_train = torch.as_tensor(y_train.reshape(-1, 1)).float()
        X_val = torch.as_tensor(X_val.values).float()
        y_val = torch.as_tensor(y_val.reshape(-1, 1)).float()

        dataset['train'] = TensorDataset(X_train, y_train)
        dataset['val'] = TensorDataset(X_val, y_val)

        return dataset
```

```python
    else:
        # NaN 값 0으로 채우기
        data = data.fillna(0)

        # 성별 ['M', 'F'] -> [0, 1]로 변환
        data['gender_enc'] = np.where(data['gender'] == 'M', 0, 1)

        # 날짜 datetime으로 변환
        # df.loc[:, 'date'] = pd.to_datetime(df['date'], format='%Y%m%d')

        DROP_COLS = ['CDMID', 'gender', 'date']
        data = data.drop(columns=DROP_COLS).copy()

        X_test = torch.as_tensor(data.values).float()

        return X_test
```

- ✓ 성별에 포함되어 있는 문자열을 INT Type으로 변환
- ✓ [학습모드일 경우] Train set과 Validation set으로 분할 (기본 비율 8:2)
- ✓ 데이터를 모델 학습 및 추론을 위해 Tensor형으로 변환

3.Baseline Code

A.I.D.D. 2021

- **Train Data Loader**

```
# training mode
if config.mode == 'train':
    data_path = DATASET_PATH + '/train/train_data'
    label_path = DATASET_PATH + '/train/train_label'

    raw_data = pd.read_csv(data_path)
    raw_labels = np.loadtxt(label_path, dtype=np.int16)
    dataset = preproc_data(raw_data, raw_labels, train=True, val_ratio=0.2, seed=1234)

    train_dl = DataLoader(dataset['train'], config.batch_size, shuffle=True)
    val_dl = DataLoader(dataset['val'], config.batch_size, shuffle=False)
    time_dl_init = time.time()
    print('Time to dataloader initialization: ', time_dl_init - time_init)
```

- ✓ Train Data와 Train Label의 path에서 각각의 데이터를 읽음
- ✓ 전처리 과정을 통해 학습 가능한 형태의 데이터로 정제
- ✓ Torch.util.data.DataLoader를 통해 DataLoader를 생성

3.Baseline Code

A.I.D.D. 2021

- **Model**

```
# 모델
class LogisticRegression(nn.Module):
    def __init__(self, input_size, output_size):
        super(LogisticRegression, self).__init__()
        self.linear = nn.Linear(input_size, output_size)

    def forward(self, x):
        return self.linear(x)
```

✓ Torch.nn.Module을 이용하여 Logistic 회귀 모델 생성

3.Baseline Code

A.I.D.D. 2021

- 학습

```
min_val_loss = np.inf
for epoch in range(config.epochs):
    # train model
    running_loss = 0.
    num_runs = 0
    model.train()
    total_length = len(train_dl)
    for iter_idx, (data, labels) in enumerate(train_dl):
        data = Variable(data)
        labels = Variable(labels)

        output_pred = model(data)
        loss = loss_fn(output_pred, labels)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        num_runs += 1

        # get current lr
        opt_params = optimizer.state_dict()['param_groups'][0]
        step = epoch*total_length+iter_idx

        nsml.report(
            epoch=epoch+int(config.iteration),
            epoch_total=config.epochs,
            iter=iter_idx,
            iter_total=total_length,
            batch_size=config.batch_size,
            train__loss=running_loss / num_runs,
            step=step,
            lr=opt_params['lr'],
            scope=locals()
        )
```

- ✓ Train Loader를 통해 데이터를 순차적으로 Load
- ✓ Loss와 optimizer를 이용하여 학습을 진행
- ✓ 각 train epoch종료시마다 nsml.report()를 통해 결과치를 nsml에 업로드

3.Baseline Code

A.I.D.D. 2021

- 검증

```
# test model with validation data
model.eval()
running_loss = 0.
num_runs = 0
for data, labels in val_dl:
    data = Variable(data)
    labels = Variable(labels)

    output_pred = model(data)
    loss = loss_fn(output_pred, labels)

    running_loss += loss.item()
    num_runs += 1

print(f"[Validation] Loss: {running_loss / num_runs}")

nsml.report(
    summary=True,
    epoch=epoch,
    epoch_total=config.epochs,
    val__loss=running_loss / num_runs,
    step=(epoch+1) * total_length,
    lr=opt_params['lr']
)

if (running_loss < min_val_loss) or (epoch % 10 == 0):
    nsml.save(epoch)
```

- ✓ Train이 끝나면 validation Loader를 통해 데이터를 순차적으로 Load
- ✓ Prediction과 Ground Truth를 비교하여 Loss를 계산
- ✓ 각 validation epoch 종료시마다 nsml.report()를 통해 결과치를 nsml에 업로드
- ✓ 전체 Epoch 종료시마다 nsml.save(epoch)를 통해 모델 weight를 nsml에 저장

3.Baseline Code

A.I.D.D. 2021

- 환경설정

```
#nsml: pytorch/pytorch:1.5-cuda10.1-cudnn7-runtime

from distutils.core import setup

setup(
    name='nia dm hackathon example',
    version='1.0',
    install_requires=[
        'pandas',
        'joblib',
        'sklearn',
        'lightgbm'
    ]
)
```

- ✓ #nsml: 에 Docker tag를 넣어 런타임의 베이스가 될 도커 이미지를 설정
- ✓ Install_requires에 pip로 설치할 라이브러리를 리스트로 입력

4.NSML 학습 및 제출 가이드

A.I.D.D. 2021

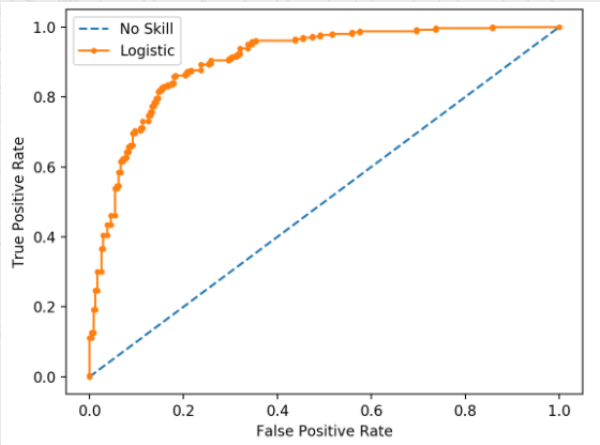
```
# 명칭이 'nia_dm'인 데이터셋을 사용해 세션 실행하기
$ nsml run -d nia_dm
# 메인 파일명이 'main.py'가 아닌 경우('-e' 옵션으로 entry point 지정)
$ nsml run -d nia_dm -e [파일명]

# 세션 로그 확인하기
# 세션명: [유저ID/데이터셋/세션번호] 구조
$ nsml logs -f [세션명]

# 세션 종료 후 모델 목록 및 제출하고자 하는 모델의 checkpoint 번호 확인하기
# 세션명: [유저ID/데이터셋/세션번호] 구조
$ nsml model ls [세션명]


# 모델 제출 전 제출 코드에 문제가 없는지 점검하기('-t' 옵션)
$ nsml submit -t [세션명] [모델_checkpoint_번호]

# 모델 제출하기
# 제출 후 리더보드에서 점수 확인 가능
nsml submit [세션명] [모델_checkpoint_번호]
```

***1시간에 1회 Submit 가능**

5.평가지표

- **AUC_ROC**

✓ Sensitivity = True Positive / (True Positive + False Negative)
✓ Specificity = True Negative / (True Negative + False Positive)
✓ TPR(True Positive Rate) = Sensitivity
✓ FPR(False Positive Rate) = 1-Specificity



- **네이버 NSML을 이용한 인공지능 모델 개발**

✓ 종료 시점의 NSML 리더보드 점수 기준으로 수상팀이 선정되며,
✓ 동점팀 발생 시 ①모델 제출이 빠른 순서 ②모델 크기가 작은 순으로 순위가 선정됩니다.

경희의료원

A.I.D.D. 2021

'2021 인공지능 학습용 데이터 구축사업'을 통해 구축된

인공지능 학습용 데이터를 이용한 당뇨병 발병 예측 모델 개발 챌린지