황승현 활동정리

주요 활동 (내용 추가 중)

- 데이터 전처리
- 모델 선정 및 학습
- 모델 설계

데이터 전처리

1. 데이터셋 분리
    - 남성 사용 안 함
    - 여성(폐경 전) < AS1_PMYN_C: 1
    - 여성(폐경 후) < AS1_PMYN_C: 2
2. 각각 사용할 독립변수, 종속변수 행 추출
3. 질환유무, 약물력 변수 가공
4. 가족력, 가족과의 관계 변수 가공
    - 부, 모, 형제자매, 기타 -> 부모, 형제자매, 기타, 해당없음
5. 종속변수 AS1_OP 가공
    - AS1_DT
    - AS1_MT
6. 데이터셋을 DataFrame으로 불러오고 독립변수는 4가지로 분류 (binary, cath0, cath1, cnt, tumor_name)
    - binary: 범주형(binary) 변수, 0 / 1 로 변경
    - catH0: 계층 없는 범주형(>03) 변수, 벡터화(one-hot-encoding)
    - catH1: 계층 있는 범주형 변수, 표준화
    - cnt: 연속형 변수, 정규화, 표준화

7. 결측값 제거 : 일단 0으로 대치
8. 연속형 변수 스케일링
    - ~~정규화(normalization): 0, 1~~
    - ~~표준화(standardization): 평균: 0 표준편차: 1~~
    - 계층화(quantile transform): 4분위 수

        > QuantileTransformer provides a non-parametric transformation to map the data to a uniform distribution with values between 0 and 1
        >
        > 출처: scikit learn docs

개발 환경

- Windows 11
- Python 3.9.13
- 라이브러리
    - iPython == 8.11.0
    - numpy == 1.24.2
    - pandas == 1.5.3
    - scikit-learn == 1.2.2

1. 데이터셋 불러오기

```
dataset = pd.read_csv("dataset/Dataset_OP_230330_raw.csv", index_col=0, na_values=
[77777, 99999, '#NULL!', ' '])
var_to_use = pd.read_csv("dataset/var_to_use_2023-03-30T2240I.csv")
```

- dataset
    - 식품영양학과에서 보내준 원본 데이터
    - 코드북 참고
- var_to_use
    - 데이터셋 종류별로 분류

- 데이터 가공 후 사용할 변수 목록

binary(0 또는 1)	cath0 (계층 없는 범주형 변수)	cath1 (계층 있는 범주형 변수)	cnt (연속형 변수)	tumor_nar 또는 여성
AS1_SEX	AS1_JOBB	AS1_EDUA	AS1_AGE	AS1_PDTOT
AS1_HT	AS1_FMHTREL_N	AS1_DRINK	AS1_TOTALC	AS1_TRTTC
AS1_DM	AS1_FMHTREL_P	AS1_SMOKEA	AS1_SLPAMTM	AS1_OBGY
AS1_UL	AS1_FMHTREL_S	AS1_INCOME	AS1_ENERGY	
AS1_AL	AS1_FMHTREL_O	AS1_PHYSTB	AS1_CARBO	
AS1_MI	AS1_FMDMREL_N	AS1_PHYSIT	AS1_FAT	
AS1_TH	AS1_FMDMREL_P	AS1_PHYACTL	AS1_PROT	
AS1_CH	AS1_FMDMREL_S	AS1_PHYACTM	AS1_FIBER	
AS1_CD	AS1_FMDMREL_O	AS1_PHYACTH	AS1_RETINOL	
AS1_LP	AS1_FMHEREL_N		AS1_BETACARO	
AS1_AS	AS1_FMHEREL_P		AS1_VITD	
AS1_CL	AS1_FMHEREL_S		AS1_VITE	
AS1_PV	AS1_FMHEREL_O		AS1_VITK	
AS1_KD	AS1_FMOSREL_N		AS1_VITC	
AS1_TOTCA1	AS1_FMOSREL_P		AS1_VITB12	
AS1_CV	AS1_FMOSREL_S		AS1_CALCIUM	
AS1_HN	AS1_FMOSREL_O		AS1_PHOS	
AS1_GT	AS1_FMCVAREL_N		AS1_SODIUM	
AS1_ARRM	AS1_FMCVAREL_P		AS1_POTASS	
AS1_DRST	AS1_FMCVAREL_S		AS1_MAGNE	
AS1_DRINS	AS1_FMCVAREL_O		AS1_FE	
AS1_DRHT	AS1_FMCVBREL_N		AS1_ZN	
AS1_DRAR	AS1_FMCVBREL_P		AS1_COPPER	

binary(0 또는 1)	cath0 (계층 없는 범주형 변수)	cath1 (계층 있는 범주형 변수)	cnt (연속형 변수)	tumor_nar 또는 여성
AS1_DRUL	AS1_FMCVBREL_S		AS1_MANGAN	
AS1_DRTH	AS1_FMCVBREL_O		AS1_SE	
AS1_DROS	AS1_FMCDREL_N		AS1_CHOL	
AS1_DRDM	AS1_FMCDREL_P		AS1_SFA	
AS1_DRSTK	AS1_FMCDREL_S		AS1_USFA	
AS1_DRAS	AS1_FMCDREL_O		AS1_MUFA	
AS1_DRLP	AS1_FMCHREL_N		AS1_PUFA	
	AS1_FMCHREL_P		AS1_N3N6RATIO	
	AS1_FMCHREL_S		AS1_HOMAIR	
	AS1_FMCHREL_O		AS1_NEAP	
	AS1_FMPVREL_N		AS1_PRAL	
	AS1_FMPVREL_P		AS1_DASH	
	AS1_FMPVREL_S		AS1_GLU0_TR	
	AS1_FMPVREL_O		AS1_ALBUMIN_TR	
	AS1_FMLPREL_N		AS1_CREATININE_TR1	
	AS1_FMLPREL_P		AS1_AST_TR	
	AS1_FMLPREL_S		AS1_ALT_TR	
	AS1_FMLPREL_O		AS1_TCHL_TR	
	AS1_FMGTREL_N		AS1_HDL_TR	
	AS1_FMGTREL_P		AS1_TOTPRT	
	AS1_FMGTREL_S		AS1_CA	
	AS1_FMGTREL_O		AS1_NA	
	AS1_DRCP		AS1_K	

binary(0 또는 1)	cath0 (계층 없는 범주형 변수)	cath1 (계층 있는 범주형 변수)	cnt (연속형 변수)	tumor_nar 또는 여성
	AS1_DRFH		AS1_CRP	
	AS1_PREG		AS1_HBA1C	
	AS1_HYST		AS1_INS0	
	AS1_HYSTOVARYW		AS1_PRT_U	
	AS1_OVARYW		AS1_CREATINE_U	
	AS1_OBGYOP		AS1_CA_U	
	AS1_BRCA		AS1_NA_U	
			AS1_K_U	
			AS1_TSH	
			AS1_WAIST	
			AS1_HEIGHT	
			AS1_WEIGHT	
			AS1_BMI	

2. 종속변수 정의

```
dataset.dropna(subset=['AS1_DT', 'AS1_MT'], inplace=True)
dataset['OP'] = np.where((dataset['AS1_DT'] <= -2.5) | (dataset['AS1_MT'] <= -2.5), 1,
0)
```

'AS1_DT' <= -2.5 또는 'AS1_MT' <= -2.5 이면 1 아니면 0

3. 데이터 가공

3.1 질환유무 가공

```
disease_list = np.array(['HT', 'DM', 'UL', 'AL', 'MI', 'TH', 'CH', 'CD',
                         'LP', 'AS', 'CL', 'PV', 'KD', 'CV', 'HN', 'GT',
                         'TOTCA1'])
for x in disease_list:
    dataset['AS1_'+x] = np.where((dataset['AS1_PD'+x] == 2) | (dataset['AS1_TRT'+x] ==
2), 1, 0)

dataset['AS1_ARRM'] = np.where((dataset['AS1_JOAR'] == 2) | (dataset['AS1_JORM'] == 2)
| (dataset['AS1_TRT'+x] == 2), 1, 0)
```

'AS1_병이름'. 코드북 참고

_PD병이름 == 2 또는 _TRT병이름 == 2 이면 1 아니면 0

3.2 약물력 가공

```
# 약물력 가공
drug_list = np.array(['ST', 'CP', 'INS', 'HT', 'AR', 'UL',
                      'TH', 'FH', 'OS', 'DM', 'STK', 'AS', 'LP'])  # CP, HP 제외
for x in drug_list:
    dataset['AS1_DR'+x] = np.where((dataset['AS1_DRUG'+x] == 2) |
(dataset['AS1_DRUG'+x+'CU'] == 2), 1, 0)
```

'AS1_DR약물이름'. 코드북 참고

_DRUG병이름 == 2 또는 _DRUG병이름CU == 2이면 1 아니면 0

3.3 가족력 가공

```
disease_dict = {
    'HT': 4,
    'DM': 4,
    'CVA': 3,
    'HE': 2,
    'OS': 2,
    'CVB': 2,
    'CD': 2,
    'PV': 2,
    'LP': 2,
    'GT': 2,
    'CH': 1
}

for disease, num_family in disease_dict.items():
    dataset[f'AS1_FM{disease}REL_N'] = np.all(dataset[[f'AS1_FM{disease}REL{i}A' for i
in range(1, num_family+1)]] == 99999, axis=1).astype(int)
    dataset[f'AS1_FM{disease}REL_P'] = np.any(dataset[[f'AS1_FM{disease}REL{i}A' for i
in range(1, num_family+1)]] <= 2, axis=1).astype(int)
    dataset[f'AS1_FM{disease}REL_S'] = np.any(dataset[[f'AS1_FM{disease}REL{i}A' for i
in range(1, num_family+1)]] == 3, axis=1).astype(int)
    dataset[f'AS1_FM{disease}REL_O'] = np.any(dataset[[f'AS1_FM{disease}REL{i}A' for i
in range(1, num_family+1)]] == 4, axis=1).astype(int)
```

- 'AS1_FM병이름REL_N'
    - 가족력 없음
    - 각 병이름 가족력 있는 사람 아무도 없으면 1 있으면 0
- 'AS1_FM병이름REL_P'
    - 가족력 부모
    - 각 병이름 가족력 중 1 또는 2 있으면 1 없으면 0
- 'AS1_FM병이름REL_S'
    - 가족력 형제자매
    - 각 병이름 가족력 중 3 있으면 1 없으면 0
- 'AS1_FM병이름REL_O'
    - 가족력 기타
    - 각 병이름 가족력 중 4 있으면 1 없으면 0

3.4 여성력 가공

```
for x in ['CP', 'FH']:
    dataset['AS1_DR'+x] = np.where(dataset['AS1_SEX'] == '1',
                                   0,
                                   np.where((dataset['AS1_DRUG'+x] == 2) |
(dataset['AS1_DRUG'+x+'CU'] == 2), 2, 1))

women = np.array(['AS1_PREG', 'AS1_HYST', 'AS1_HYSTOVARYW', 'AS1_OVARYW',
                  'AS1_OBGYOP', 'AS1_BRCA', 'AS1_DRCP', 'AS1_DRFH'])

for x in women:
    dataset[x] = np.where((dataset['AS1_SEX'] == 1), 0, dataset[x])
```

- 남성: 0
- 여성 1 또는 2
    - 병 있음 / 약물 있음: 1
    - 없음: 2

4. 결측값 대치

```
raw_binary = raw_binary.fillna(0)
raw_cath0 = raw_cath0.fillna(0)
raw_cath1 = raw_cath1.fillna(0)
raw_cnt = raw_cnt.fillna(0)
raw_tumor_name = raw_tumor_name.fillna(0)

raw_label = raw_label.fillna(0)
```

모든 변수 결측값 0으로 대체

5. 데이터 표준화

Standardization, OneHotEncoding

```
binary = pd.DataFrame(scaler.transform(raw_binary), index=dataset.index,
columns=raw_binary.columns).astype('float')
cath0 = pd.get_dummies(raw_cath0, columns=raw_cath0.columns, drop_first=True,
dtype='float64') # one-hot-encoding
cath1 = pd.DataFrame(scaler.transform(raw_cath1), index=dataset.index,
columns=raw_cath1.columns).astype('float')
cnt = pd.DataFrame(scaler.transform(raw_cnt), index=dataset.index,
columns=raw_cnt.columns).astype('float')
tumor_name = pd.get_dummies(raw_tumor_name, columns=raw_tumor_name.columns,
drop_first=True, dtype='float64') # one-hot-encoding

label = pd.DataFrame(normalize(raw_label, norm='l2'), index=dataset.index,
columns=raw_label.columns).astype('float')
```

- sklearn.preprocessing.StandardScaler
  - binary, cath1, cnt, label에 사용
  - 가우시안 분포
- pandas.get_dummies
  - cath0, tumor_name에 사용
  - one-hot-encoding
    - 범주형 데이터를 기계가 이해할 수 있게 숫자 형식으로 바꾸는 것.
    - 0 또는 1 이진 벡터로 변환

모델 선정 및 학습

사용할 알고리즘 목록

- Linear Regression Model
- KNN
- Decison Tree
- LGBM
- XGBoost
- MLP
- SAINT

Classic ML

- Linear Regression Model(선형 회귀)
    - 데이터를 가장 잘 나타내는 선형 방정식 (y = ax + b)을 찾는 것.
    - 예측 값과 실제 값 사이의 오차가 가장 적은 선을 찾는 것.
    - 다중 선형회귀도 가능하나, 과적합
- KNN (K-최근접 이웃)
    1. 교육 데이터에서 새 데이터 포인트에 대한 k개의 가장 가까운 이웃 찾기
    2. 가장 가까운 이웃의 레이블을 기반으로 새 데이터 포인트의 레이블을 예측
- Decision Tree (결정트리)
    - 입력 변수의 값을 기준으로 데이터를 하위 집합으로 재귀적으로 분할
    - 하위 집합이 대상 변수에 대해 분산이 낮거나 유사성이 높도록 하는 지도 학습 알고리즘입니다.
        - GINI
    - 트리가 너무 복잡해지면, 과적합 되기 쉬움

GBDT

Gradient-boosted decision trees

- 정의
    - Decision Tree의 앙상블
    - Decision Tree를 학습한 후 경사 하강법 알고리즘을 적용하여 잘못 분류된 데이터 샘플의 가중치를 업데이트
        - 경사하강법: gradient descent
        - 예측값과 실제값의 차이(비용함수)를 최소화하여 모델 개선
        - 비용함수를 조금씩(learning late) 이동(기울기 변화)하면서 비용함수의 값을 최소화
    - 업데이트된 데이터로 다른 tree 생성. (앙상블)
    - 모델 성능이 더 좋아지지 않을 때까지 반복

XGBoost

eXtreme Gradient Boosting

- 2014, Tianqi Chen
- 각 후속 트리가 이전 트리의 오류 수정을 시도하는 일련의 결정 트리를 구축합니다. 각 단계에서 알고리즘은 음의 기울기에 가장 잘 맞는 결정 트리를 구성하는 데 사용되는 손실 함수의 기울기와 헤시안을 계산합니다. 이 프로세스는 원하는 트리 수에 도달하거나 모델이 과적합되기 시작할 때까지 반복됩니다.

LGBM

Light Gradient Boosting Machine

- 2017, Microsoft
- GOSS(Gradient-based One-Side Sampling) 및 EFB(Exclusive Feature Bundling)라는 기술을 기반으로 의사 결정 트리를 구축하기 위해 다른 전략을 사용합니다. GOSS는 손실 함수의 기울기를 기반으로 훈련 데이터를 샘플링하는 기법으로, 정확도를 희생하지 않으면서 훈련에 사용되는 샘플 수를 줄이는 데 도움이 됩니다. EFB는 학습에 사용되는 기능의 수를 줄이기 위해 유사한 기능을 함께 묶어 오버피팅을 줄이고 속도를 향상시키는 데 도움이 되는 기술입니다.

DNN

Deep Neural Networks

- 정의
    - 인간 두뇌(뉴런)를 모방
        - 퍼셉트론
    - 은닉층에 많은 노드(뉴런)를 넣고, 성능을 최적화하기 위해 훈련 중에 뉴런의 가중치와 편향이 조정
    - Deep: 깊다. 은닉층이 많다.

MLP

- 가장 기초적인 DNN
- 퍼셉트론을 여러겹 쌓음
- **역전파**로 가중치와 편향을 조정하여, 오류가 적음

    - 순전파는 입력층에서 은닉층을 거쳐 출력층으로
    - 역전파는 출력층에서 은닉층을 거쳐... 반대로.
        - 오차를 줄이기 위해 가중치 조정
        - 은닉층의 각 가중치를 오차에 대하여 미분.
        - 체인 법칙으로 식을 풀어서 미분 풀어씀.
- 과적합을 방지하기 위해, 파라미터를 세심하게 조정해야 함.
