

# 머신러닝 예측모델

[ 당뇨병 예측 ]

2024.01.05 (금)

송영달, 송은민, 임창성, 조진우, 함은규

Mean\_30



## I. 프로젝트의 목적

당뇨병 예측모델을 개발

## II. 원본 데이터의 구성확인 (1차 분류)

Datafile : CSV 5종

## III. 원본 데이터 분석 (2차 분류)

1. 원본데이터 분류
2. Feature extraction

## IV. 데이터 전처리

1. Datafile 통합(merge)
2. 결측값(nan) 처리
3. Scaling & Encoding

## V. 학습 모델과 모델 성능평가

1. pycaret 모델링(최적화&앙상블)
2. 검증 데이터 평가(임계값 조정)
3. 테스트 데이터 예측 및 평가

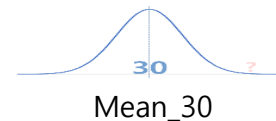
## VI. 결론

## VII. 최종 결과물

## VIII. 프로젝트 한계, 사용기술

## IX. 자료출처

# I. 프로젝트의 목적

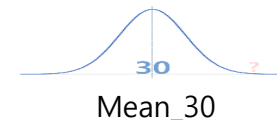


1. 목적 : 머신러닝을 이용한 당뇨병 예측모델 개발

2. 과정

- 1) 적절한 전처리를 통한 필요 데이터 추출
- 2) 다양한 모델과 하이퍼파라미터 조합(비교)을 통한 최적의 예측모델 구축
- 3) AUTOML 사용을 통한 최적화

## II. 원본 데이터의 구성확인 (1차 분류)



### 1. Datafile : CSV 5종

- 1) Family file
  - 2) Household file
  - 3) Person file
  - 4) Sample Child file
  - 5) Sample Adult file
- Datafile summary
  - Datafile layout

[Datafile Feature 설명]

[Datafile Feature 상세 설명 및 질문]

### 2. imputed incomes


[Data파일] \*불필요

### 3. Paradata

[CSV 파일] \*불필요 (설문관련)

### 4. Functioning and Disability

[PDF 파일] \*불필요 (PDF)

 National Health Interview Survey

2018 Data Release

Print

- Readme File. [PDF - 30 KB]
- Notices for Data Users. [PDF - 30 KB]

Data Files

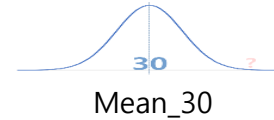
- Family file
  - Variable summary. [PDF - 51 KB]
  - Variable layout. [PDF - 318 KB]
  - Variable frequencies. [PDF - 86 KB]
  - ASCII data. [ZIP - 821 KB]
  - CSV data. [ZIP - 918 KB]
  - Sample SAS statements. [SAS - 29 KB]
  - Sample SPSS statements. [SPS - 26 KB]
  - Sample Stata statements. [DO - 24 KB]
- Household file
  - Variable summary. [PDF - 22 KB]
  - Variable layout. [PDF - 91 KB]
  - Variable frequencies. [PDF - 22 KB]
  - ASCII data. [ZIP - 409 KB]
  - CSV data. [ZIP - 385 KB]
  - Sample SAS statements. [SAS - 7 KB]
  - Sample SPSS statements. [SPS - 4 KB]
  - Sample Stata statements. [DO - 6 KB]
- Person file
  - Variable summary. [PDF - 19 KB]
  - Variable layout. [PDF - 1.9 MB]
  - Variable frequencies. [PDF - 348 KB]
  - ASCII data. [ZIP - 3.1 MB]
  - CSV data. [ZIP - 3.6 MB]
  - Sample SAS statements. [SAS - 111 KB]
  - Sample SPSS statements. [SPS - 156 KB]
  - Sample Stata statements. [DO - 102 KB]
- Sample Child file
  - Variable summary. [PDF - 93 KB]
  - Variable layout. [PDF - 698 KB]
  - Variable frequencies. [PDF - 109 KB]
  - ASCII data. [ZIP - 305 KB]
  - CSV data. [ZIP - 340 KB]
  - Sample SAS statements. [SAS - 41 KB]
  - Sample SPSS statements. [SPS - 46 KB]
  - Sample Stata statements. [DO - 35 KB]
- Sample Adult file
  - Variable summary. [PDF - 244 KB]
  - Variable layout. [PDF - 1.5 MB]
  - Variable frequencies. [PDF - 400 KB]
  - ASCII data. [ZIP - 3 KB]
  - CSV data. [ZIP - 3.4 MB]
  - Sample SAS statements. [SAS - 137 KB]
  - Sample SPSS statements. [SPS - 209 KB]
  - Sample Stata statements. [DO - 124 KB]

Imputed Income Files

Paradata File

Functioning and Disability File

### Ⅲ. 원본 데이터 분석 (2차 분류) (1/2)

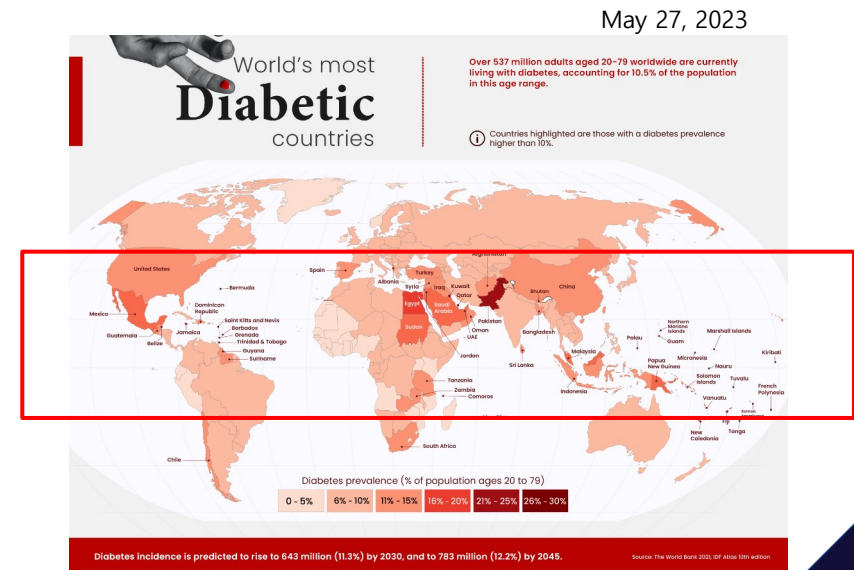


#### 1. Feature 선정기준(1) – 참조자료

논문명	당뇨관련 항목	관련 Feature	선택유무	기타
당뇨병관리 소아에서노인까지 소아당뇨병이란 무엇인가 소아연령에서의 2형당뇨병의 임상적고찰	키	CHGHT_TC	O	-
	체중	CWGHT_TC	O	-
	BMI	BMI_SC	O	-
	다뇨	CINTIL2W	O	-
	다식		X	관련 Feature 없음
	다음		X	관련 Feature 없음

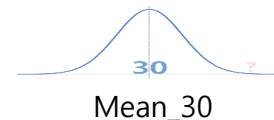
#### 2. Feature 선정기준(2): 인과관계 확인 필요 항목들 추가

- 1) 흡연여부
- 2) 성별, 임신여부(임신성 당뇨)
- 3) 인종별 차이(식습관)



<https://www.visualcapitalist.com/cp/diabetes-rates-by-country/>

# Ⅲ. 원본 데이터 분석 (2차 분류) (2/2)



## 3. 원본 데이터 분류

- 1) 분류 기준 : 당뇨병 관계성(논문 등 참조) 및 결측 값이 적은 항목  
예) 류마티즘 약 복용 등
- 2) 분류 방법 : 데이터 구분(사용, 점검, 참고)

## 4. Feature 필터링 (Feature Selection : Feature extraction)

순	Datafile	columns
1	Family file	1개
2	Household file	불필요
3	Person file	불필요
4	Sample Child file	12개
5	Sample Adult file	33개

동일한 조사 내용  
Feature명 통일

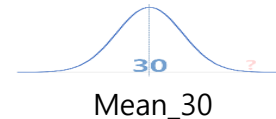
머신러닝 프로젝트 (당뇨병 예측) 2023.12.26~

RS, 컴퓨터 통일

[NHIS 2018] ([https://www.cdc.gov/nchs/nhis/nhis\\_2018\\_data\\_release.htm](https://www.cdc.gov/nchs/nhis/nhis_2018_data_release.htm))

구분	OQ_docs	자료구분	Was_Before	As_Is	변경유무	파일 유형 식별자
1 TARGET 당뇨	Sample Adult	핵심	DIBEV1	DIBEV1	-	당뇨병 진단 받은
2 TARGET 당뇨	Sample Child	핵심	CCONDRR6	DIBEV1	변경	SC는 당뇨병 진단 받은
3 나이	Sample Adult	사용	AGE_P	AGE	변경	나이
4 나이	Sample Child	사용	AGE_P	AGE	변경	나이
5 성별	Sample Adult	사용	SEX	GENDER	변경	성별
6 성별	Sample Child	사용	SEX	GENDER	변경	성별
7 키	Sample Adult	사용	AHEIGHT	HEIGHT	변경	종 키 인치 (백)
8 키	Sample Child	사용	CHGHT_TC	HEIGHT	변경	종 키 인치 (백)
9 체중	Sample Adult	> > 참고	AWEIGHTP	WEIGHT	변경	신발없는 몸무게 (파운드)
10 체중	Sample Child	> > 참고	CWGHT_TC	WEIGHT	변경	몸무게 (파운드)
11 임신	Sample Adult	사용	PREGFLYR	PREGFLYR	-	최근 임신
12 임신	Sample Adult	사용	PREGNOW	PREGNOW	-	현재 임신중
13 예방 자료	Sample Adult	사용	AHCPLRQ1	GP1RQ1	변경	예방적으로 임신/예방 접종을 한다.
14 구토/설사	Sample Adult	> > 참고	AINTL2W	INTIL2W	변경	구토/설사를 동반한 위 문제가 있음 _ 2주
23 구토/설사	Sample Child	> > 참고	CINTIL2W	INTIL2W	변경	구토/설사를 동반한 위장병이 있었음 _ 기간기준 2주
24 알코올	Sample Adult	사용	ALCSTAT	ALCSTAT	-	음주 상태 : 기록
25 담배(흡연)	Sample Adult	사용	SMKSTAT2	SMKSTAT2	-	흡연 상태 : 기록
26 담배(흡연)	Sample Adult	> > 데이터 점검	CIGAREV2	CIGAREV2	-	일반 시가 담배 또는 작은 필터 시가(담배)를 한 번이라도 피워본적 있다
27 담배(흡연)	Sample Adult	> > 데이터 점검	ECIGEV2	ECIGEV2	-	전자 담배를 한 번이라도 사용한 적 있다.

## IV. 데이터 전처리 (1/3)



### 1. Datafile 통합(merge)

- 1) 3개 데이터셋 : Family , Sample Child, Sample Adult
- 2) 이슈 : 파일통합 시, 동일 Feature가 있음에도 Feature이 새로 생성됨
  - 해결방법 : combine\_first 함수 사용

```
# family 데이터프레임에 adult 데이터 프레임 merge
fam_adult = pd.merge(new_fam, new_adult, how='left', on='id')
# combine_first() 이걸로 해결
total = fam_adult.combine_first(new_child)
```

### 2. 전처리/결측치(nan) 처리

2.6 전처리 : AGE 12세 이상

2.7 전처리 : HYPMDEV2(고혈압) yes(1) 외 no(2 or 0)

2.8 전처리 : 결측치가 90퍼센트 이상인 feature 제거

2.9 결측치 처리

2.9.1 결측치 : 결측 Feature 목록화 및 처리

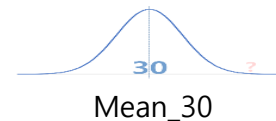
2.9.2 결측치 : PREGNOW / PREGFLYR(임신)

2.9.3 결측치 : 불필요 feature 제거

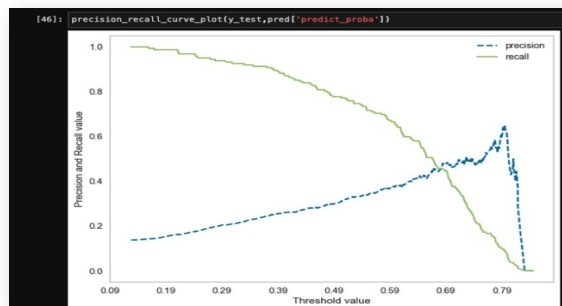
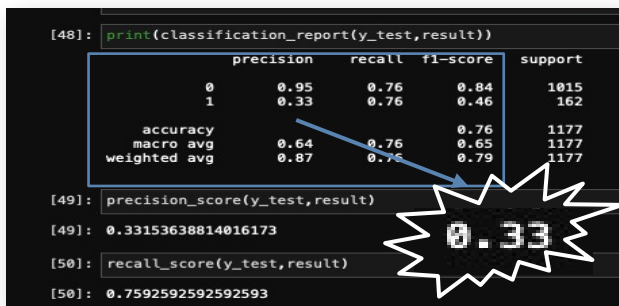
2.10 이상치 : 키, 몸무게 이상값 제거

2.10.1) BMI 재계산을 위해 단위변경[키(인치→cm), 몸무게(파운드→ kg)]

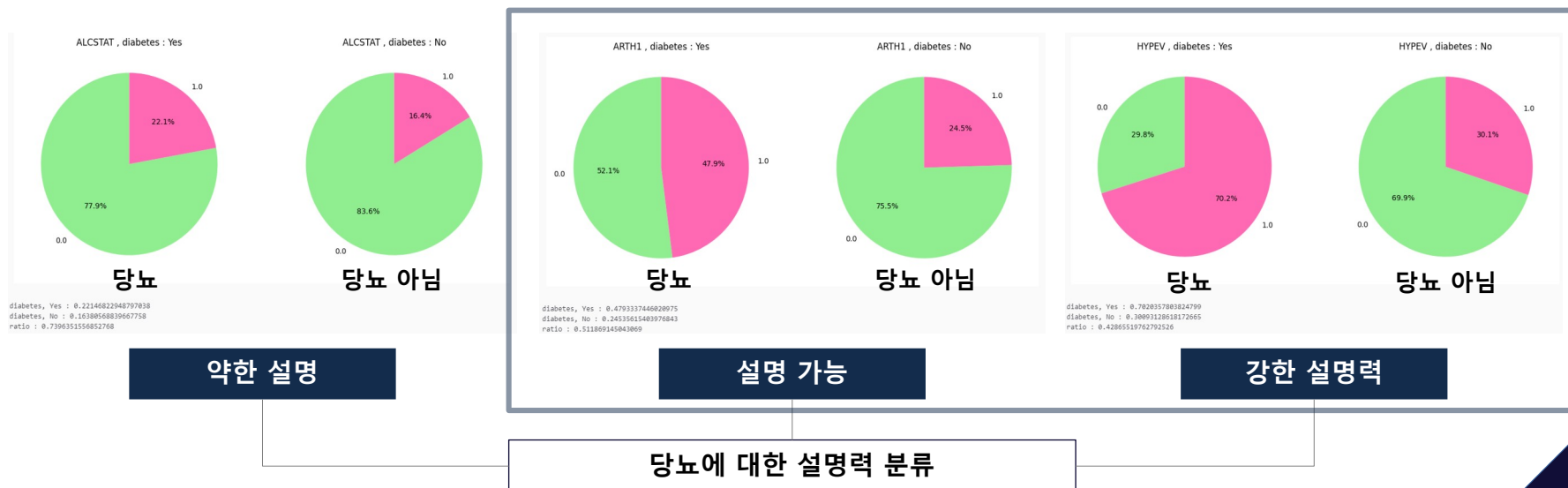
## IV. 데이터 전처리 (2/3)



### 3. 1차 전처리 데이터 학습모델 성능 (preview)

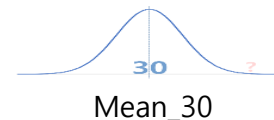


### 4. 2차 전처리 : Feature 추가점검 (당뇨 여부에 미치는 영향 점검)





# IV. 데이터 전처리 (3/3)



## 5. 데이터 전처리 변수 선택

- 당뇨와 상관성 및 RFECV (Recursive Feature Elimination with Cross Validation)

```
[40]: importance
[40]: array([0.00576027, 0.02091774, 0.01638419, 0.00460221, 0.17040429,
0. , 0.02210643, 0. , 0.00808969, 0. ,
0.00785057, 0.05416056, 0.09600868, 0.03507939, 0.04099191,
0.00384618, 0.00653778, 0. , 0.01253758, 0.0254828 ,
0. , 0.00332202, 0.00407399])

[18]: from sklearn.feature_selection import SelectFromModel
sfm = SelectFromModel(lasso, threshold=0.001)
sfm.fit(X_under, y_under)
print(sfm.threshold_)
print(sfm.get_support())
print('선택된 피처:', X_under.columns[sfm.get_support()])

0.001
[ True True True True True False True False True False True True
 True True True True True False True False True True]
선택된 피처: Index(['AGE', 'ALCSTAT', 'ARTHI', 'BMI', 'CHLEV', 'CPLROU', 'FSBALANC',
'HISPAN_I', 'HYPEV', 'HYPMDEV2', 'HYPMED2', 'INTIL2W', 'MRACBP12',
'MRACRP12', 'SMKSTAT2', 'TIRED_1', 'HEIGHT(cm)', 'WEIGHT(kg)'],
dtype='object')

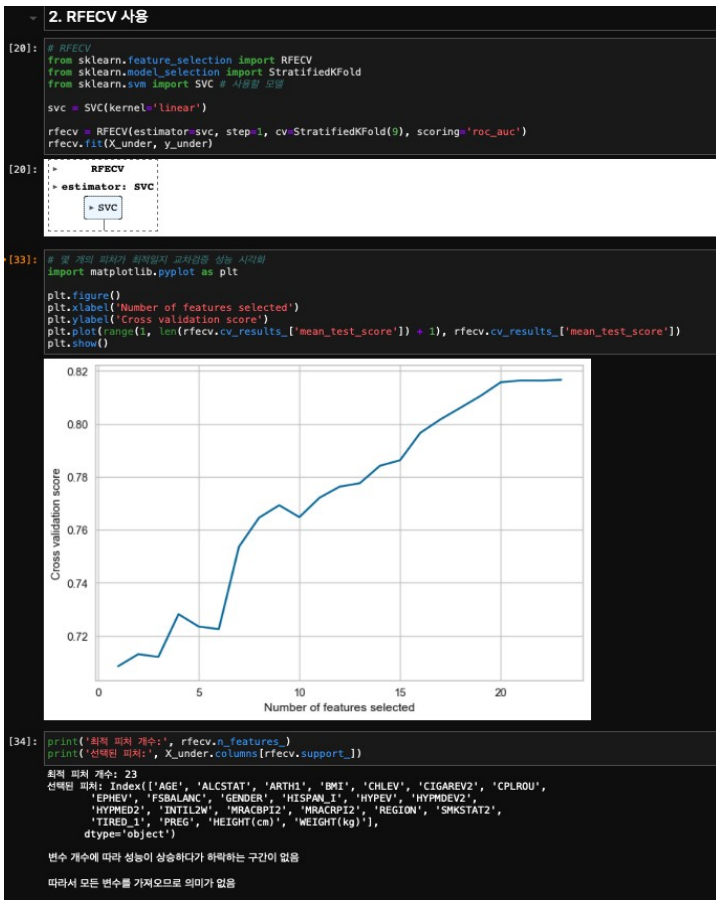
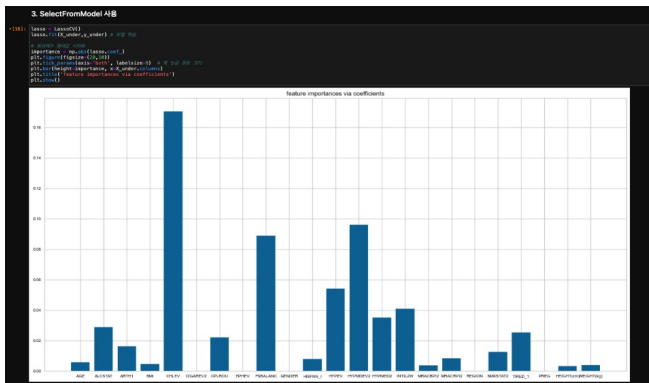
[19]: selected_cols = X_under.columns[sfm.get_support()]

[20]: X_under = X_under[selected_cols]
X_test = X_test[selected_cols]

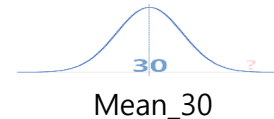
[21]: X_under.columns

[21]: Index(['AGE', 'ALCSTAT', 'ARTHI', 'BMI', 'CHLEV', 'CPLROU', 'FSBALANC',
'HISPAN_I', 'HYPEV', 'HYPMDEV2', 'HYPMED2', 'INTIL2W', 'MRACBP12',
'MRACRP12', 'SMKSTAT2', 'TIRED_1', 'HEIGHT(cm)', 'WEIGHT(kg)'],
dtype='object')

25개 변수 중 18개 변수가 선택됨
해당 변수 기준으로 성능 평가시 개선되지 않음
```



# V. 학습 모델과 모델 성능평가 (1/4)



## 1. import 라이브러리

```
[1]: import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold
from imblearn.under_sampling import RandomUnderSampler
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings(action='ignore')
from sklearn.metrics import roc_auc_score, classification_report, recall_score, f1_score, precision_recall_curve, auc, precision_score, roc_curve, confusion_matrix
import multiprocessing
from pycaret.classification import *
from sklearn.linear_model import LassoCV
import seaborn as sns
sns.set(font="AppleGothic", # Mac 운영체제 경우
        # font="Malgun Gothic", # Windows 운영체제 경우
        rc={"axes.unicode_minus": False}) # 마이너스 폰트 깨지는 문제 해결
```

## 2. imbalanced data 처리

```
[70]: sampler = RandomUnderSampler(random_state=123)

[71]: y_tr.value_counts()

[71]: 0    18871
      1     3080
      Name: DIBEV1, dtype: int64

[72]: X_under, y_under = sampler.fit_resample(X_tr, y_tr)
print(y_under.value_counts())

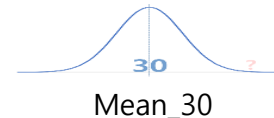
0     3080
1     3080
      Name: DIBEV1, dtype: int64
```

## 3. 수치형 데이터 스케일링

```
[73]: # 스케일링을 적용할 컬럼 선정
num_col = ['AGE', 'BMI', 'HEIGHT(cm)', 'WEIGHT(kg)']

[74]: # 전체 데이터셋의 위 컬럼들에 대해 스케일링 fit 후 각 데이터
for col in num_col:
    ss = StandardScaler()
    X_under[col] = ss.fit_transform(X_under[[col]])
    X_test[col] = ss.transform(X_test[[col]])
```

## V. 학습 모델과 모델 성능평가 (2/4)



### 4. onhotencoding (2진값 外)

```
[75]: # 데이터프레임 전체 컬럼에서 수치를 컬럼 제외
cols = np.setdiff1d(X_under.columns, num_col)

# 위 컬럼에서 고유값 개수가 3개 이상의 컬럼만 추출
# 0,1만 가지는 binary 컬럼은 one-hot 하지 않을 것
nom_col = [col for col in cols if X_under[col].nunique() >= 3 ]

[76]: nom_col

[76]: ['HISPAN_I', 'MRACBP12', 'MRACRP12', 'REGION']

[77]: # 컬럼별 컬럼들에 대한 dummy 데이터 생성(편차인코딩)
train_dummies = [] # 학습용 데이터셋의 컬럼별 컬럼들의 데이터셋을 저장할 리스트
test_dummies = [] # 검증용 데이터셋의 컬럼별 컬럼들의 데이터셋을 저장할 리스트
for col in nom_col:
    train_dummies.append(pd.get_dummies(X_under[col], prefix=col, dtype='int')) # 학습데이터의 각 컬럼들의 데이터셋을 리스트에 저장
    test_dummies.append(pd.get_dummies(X_test[col], prefix=col, dtype='int')) # 검증데이터의 각 컬럼들의 데이터셋을 리스트에 저장

[78]: train_dummies = pd.concat(train_dummies, axis=1) # 학습 데이터의 데이터셋 리스트를 하나로 합침
test_dummies = pd.concat(test_dummies, axis=1) # 검증 데이터의 데이터셋 리스트를 하나로 합침
```

### 5. 전처리 데이터 병합

```
[80]: # 만약 고유값 개수 차이가 있거나 학습셋과 테스트셋의 데이터셋 컬럼 차이가 있다면 컬럼수를 통일
# 학습 및 예측 오류 방지

if train_dummies.columns.nunique() > test_dummies.columns.nunique():
    missing_cols = set(train_dummies.columns) - set(test_dummies.columns)
    for col in missing_cols:
        test_dummies[col] = 0
elif train_dummies.columns.nunique() < test_dummies.columns.nunique():
    missing_cols = set(test_dummies.columns) - set(train_dummies.columns)
    for col in missing_cols:
        train_dummies[col] = 0
else:
    pass

test_dummies = test_dummies[train_dummies.columns]

[81]: train_dummies.shape, test_dummies.shape

[81]: ((6160, 31), (1156, 31))

[82]: # 원본의 학습, 테스트셋에 데이터셋을 합친 후 기존 컬럼 컬럼 제거
X_under = pd.concat([X_under, train_dummies], axis=1).drop(nom_col, axis=1)
X_test = pd.concat([X_test, test_dummies], axis=1).drop(nom_col, axis=1)

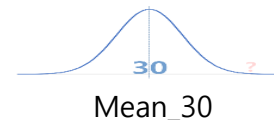
[83]: X_under.shape, X_test.shape

[83]: ((6160, 45), (1156, 45))

[84]: X_under.shape, y_under.shape

[84]: ((6160, 45), (6160,))
```

## V. 학습 모델과 모델 성능평가 (3/4)



### 6. AutoML(Pycaret) 최적화 모델링

```
[86]: skf = StratifiedKFold(n_splits=5, random_state=42, shuffle=True)
      clf = setup(data=X_tr, target=y_tr, preprocess=False, verbose=False, n_jobs=-1, session_id=123) # pycaret AutoML
      best_5 = compare_models(fold=10, sort='auc', verbose=False, n_select=5) # pycaret에서 sort에 성능을 평가하는 기준을 지정
      tuned_models = []
      for model in best_5:
          tuned_model = tune_model(model, optimize='auc', verbose=False, search_library='optuna', fold=skf) #, search_library='optuna'
          tuned_models.append(tuned_model)

      ensemble_model = blend_models(estimator_list=tuned_models, method='auto', optimize='auc', verbose=False)
```



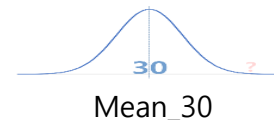
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.7364	0.8183	0.7496	0.7316	0.7402	0.4729	0.4733	0.4570
lda	Linear Discriminant Analysis	0.7376	0.8178	0.7496	0.7332	0.7411	0.4752	0.4756	0.0090
ada	Ada Boost Classifier	0.7283	0.8095	0.7420	0.7233	0.7323	0.4566	0.4572	0.0460
gbc	Gradient Boosting Classifier	0.7321	0.8091	0.7537	0.7237	0.7379	0.4642	0.4653	0.1160
rf	Random Forest Classifier	0.7182	0.7931	0.7200	0.7186	0.7187	0.4363	0.4370	0.1020
lightgbm	Light Gradient Boosting Machine	0.7106	0.7918	0.7316	0.7030	0.7165	0.4212	0.4222	0.0600
xgboost	Extreme Gradient Boosting	0.7054	0.7739	0.7147	0.7020	0.7079	0.4108	0.4114	0.0390
et	Extra Trees Classifier	0.6944	0.7692	0.6928	0.6963	0.6941	0.3888	0.3892	0.0950
nb	Naive Bayes	0.5770	0.7584	0.2371	0.7405	0.3575	0.1541	0.2097	0.0060
knn	K Neighbors Classifier	0.6979	0.7505	0.6783	0.7069	0.6916	0.3958	0.3968	0.0250
dt	Decision Tree Classifier	0.6448	0.6449	0.6348	0.6482	0.6413	0.2897	0.2898	0.0090
qda	Quadratic Discriminant Analysis	0.5100	0.5103	0.3972	0.5050	0.3976	0.0205	0.0198	0.0130
dummy	Dummy Classifier	0.4987	0.5000	0.5000	0.2494	0.3328	0.0000	0.0000	0.0050
svm	SVM - Linear Kernel	0.7237	0.0000	0.7746	0.7085	0.7340	0.4475	0.4598	0.0170
ridge	Ridge Classifier	0.7382	0.0000	0.7513	0.7332	0.7419	0.4764	0.4767	0.0080

### 7. 학습모델 검증

```
[88]: # 실제값과 예측값 비교
      print(classification_report(y_val, pred['prediction_label'], target_names=['정상', '당뇨']))
```

	precision	recall	f1-score	support
정상	0.77	0.74	0.75	616
당뇨	0.75	0.78	0.76	616
accuracy			0.76	1232
macro avg	0.76	0.76	0.76	1232
weighted avg	0.76	0.76	0.76	1232

## V. 학습 모델과 모델 성능평가 (4/4)



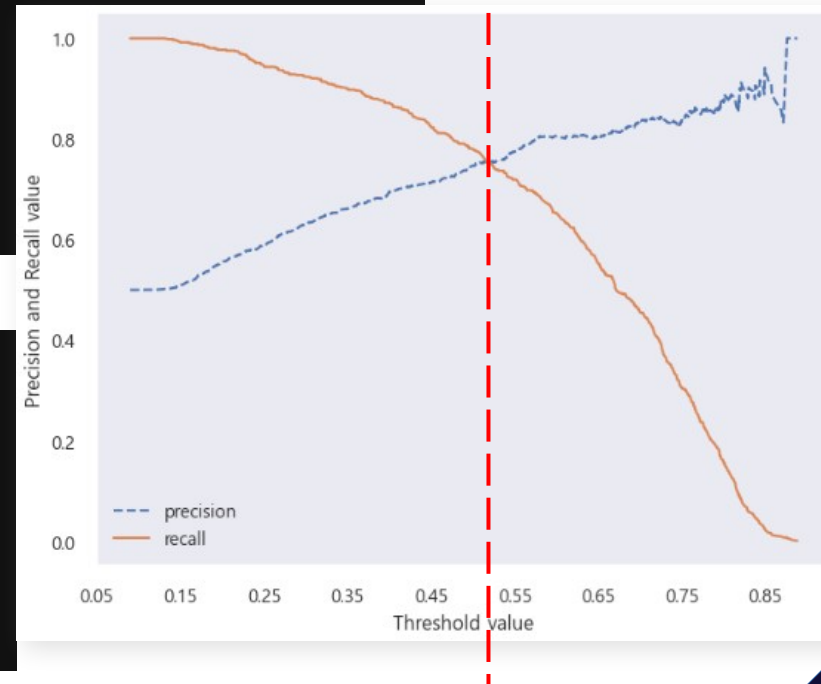
### 8. Threshold 값 추정

```
[94]: f1_score_list = []
threshold_list = []
for j in range(10):
    j = 0.45+j*0.01
    # j = 0.485
    result = [0 if i<=j else 1 for i in pred['predict_proba']]
    score = f1_score(y_val,result)
    f1_score_list.append(score)
    threshold_list.append(j)
print(f'임계값:{j} / f1_score : {round(score,3)} / recall : {round(recall_score(y_val,result),3)} / precision : {round(precision_score(y_val,result),3)}')
```

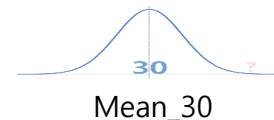
```
임계값:0.45 / f1_score : 0.765 / recall : 0.823 / precision : 0.714
임계값:0.46 / f1_score : 0.762 / recall : 0.812 / precision : 0.717
임계값:0.47000000000000003 / f1_score : 0.763 / recall : 0.807 / precision : 0.723
임계값:0.48 / f1_score : 0.761 / recall : 0.795 / precision : 0.73
임계값:0.49 / f1_score : 0.761 / recall : 0.787 / precision : 0.737
임계값:0.5 / f1_score : 0.763 / recall : 0.779 / precision : 0.747
임계값:0.51 / f1_score : 0.759 / recall : 0.766 / precision : 0.753
임계값:0.52 / f1_score : 0.752 / recall : 0.75 / precision : 0.755
임계값:0.53 / f1_score : 0.748 / recall : 0.739 / precision : 0.757
임계값:0.54 / f1_score : 0.744 / recall : 0.726 / precision : 0.763
```

### 9. Test Value : 모델 성능 측정

	precision	recall	f1-score	support
정상	0.95	0.72	0.82	994
당뇨	0.30	0.75	0.43	162
accuracy			0.72	1156
macro avg	0.62	0.73	0.62	1156
weighted avg	0.86	0.72	0.76	1156



# 참조1\_변수 중요도 확인



## ※ 학습 후 변수 중요도 확인

```
[40]: importance

[40]: array([0.00576027, 0.02891774, 0.01638419, 0.00460221, 0.17040429,
            0.          , 0.02210643, 0.          , 0.08880969, 0.          ,
            0.00785057, 0.05416056, 0.09600868, 0.03507939, 0.04099191,
            0.00384618, 0.00853778, 0.          , 0.01253758, 0.0254828 ,
            0.          , 0.00332202, 0.00407399])

[18]: from sklearn.feature_selection import SelectFromModel
      sfm = SelectFromModel(lasso, threshold=0.001)
      sfm.fit(X_under, y_under)

      print(sfm.threshold_)
      print(sfm.get_support())
      print('선택된 피처:', X_under.columns[sfm.get_support()])

      0.001
      [ True  True  True  True  True  True  True  True  True  True  True  True  True  True
        True  True  True  True  True  True  True  True  True  True  True  True  True  True]
      선택된 피처: Index(['AGE', 'ALCSTAT', 'ARTH1', 'BMI', 'CHLEV', 'CPLROU', 'FSBALANC',
                      'HISPAN_I', 'HYPERV', 'HYPMDEV2', 'HYPMED2', 'INTIL2W', 'MRACBPI2',
                      'MRACRPI2', 'SMKSTAT2', 'TIRED_1', 'HEIGHT(cm)', 'WEIGHT(kg)'],
                      dtype='object')
```

```
[19]: selected_cols = X_under.columns[sfm.get_support()]
```

```
[20]: X_under = X_under[selected_cols]
      X_test = X_test[selected_cols]
```

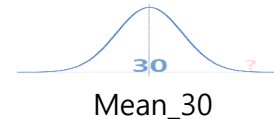
```
[21]: X_under.columns
```

```
[21]: Index(['AGE', 'ALCSTAT', 'ARTH1', 'BMI', 'CHLEV', 'CPLROU', 'FSBALANC',
          'HISPAN_I', 'HYPERV', 'HYPMDEV2', 'HYPMED2', 'INTIL2W', 'MRACBPI2',
          'MRACRPI2', 'SMKSTAT2', 'TIRED_1', 'HEIGHT(cm)', 'WEIGHT(kg)'],
          dtype='object')
```

25개 변수 중 18개 변수가 선택 됨

해당 변수 기준으로 성능 평가시 개선되지 않음

## 참조2\_개별 모델 성능 확인



※ Ensemble Model에 포함된 개별 모델 성능 확인

```
[104]: for model in tuned_models:
        pred = model.predict(X_test)
        print(str(model.get_params()).find('of')*3+str(model.get_params().find('{}'))
        print(classification_report(y_test,pred))
```

LogisticRegression

	precision	recall	f1-score	support
0	0.95	0.74	0.83	1015
1	0.32	0.77	0.45	162
accuracy			0.74	1177
macro avg	0.63	0.75	0.64	1177
weighted avg	0.86	0.74	0.78	1177

LinearDiscriminantAnalysis

	precision	recall	f1-score	support
0	0.95	0.74	0.83	1015
1	0.32	0.76	0.45	162
accuracy			0.74	1177
macro avg	0.63	0.75	0.64	1177
weighted avg	0.86	0.74	0.78	1177

GradientBoostingClassifier

	precision	recall	f1-score	support
0	0.96	0.72	0.82	1015
1	0.31	0.79	0.45	162
accuracy			0.73	1177
macro avg	0.63	0.76	0.64	1177
weighted avg	0.87	0.73	0.77	1177

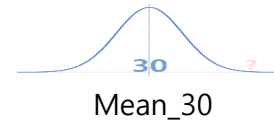
AdaBoostClassifier

	precision	recall	f1-score	support
0	0.95	0.73	0.83	1015
1	0.32	0.78	0.45	162
accuracy			0.74	1177
macro avg	0.64	0.76	0.64	1177
weighted avg	0.87	0.74	0.78	1177

[LightGBM] [Warning] bagging\_freq is set=4, subsample\_freq=0 will be ignored. Current value: bagging\_freq=4  
[LightGBM] [Warning] feature\_fraction is set=0.9, colsample\_bytree=1.0 will be ignored. Current value: feature\_fraction=0.9  
[LightGBM] [Warning] bagging\_fraction is set=0.6, subsample=1.0 will be ignored. Current value: bagging\_fraction=0.6

LGBMClassifier

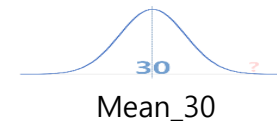
	precision	recall	f1-score	support
0	0.96	0.73	0.83	1015
1	0.32	0.80	0.46	162
accuracy			0.74	1177
macro avg	0.64	0.76	0.64	1177
weighted avg	0.87	0.74	0.78	1177



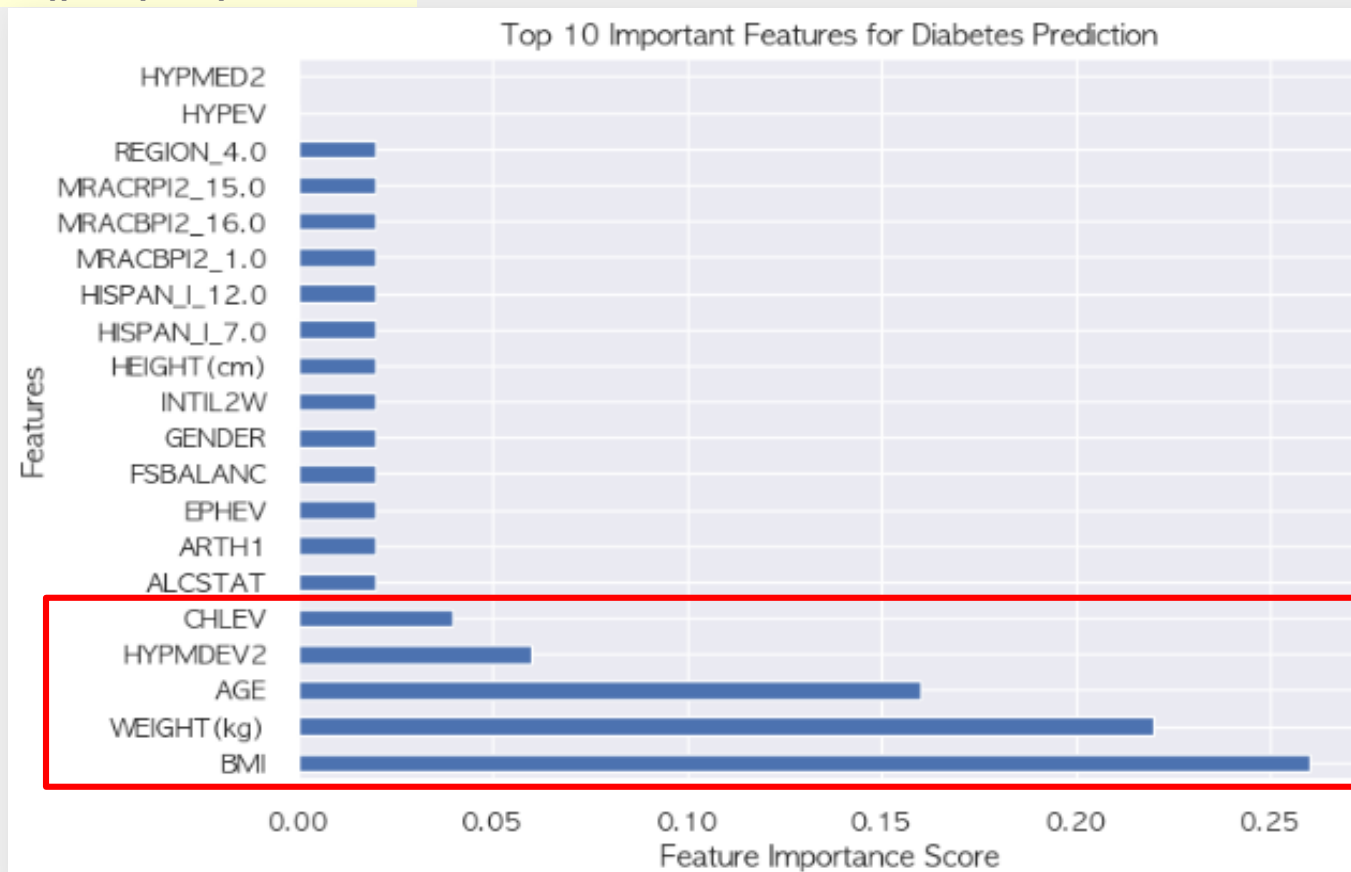
HTML 코드 문서 참조



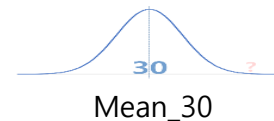
## VI. 결론(1/3)



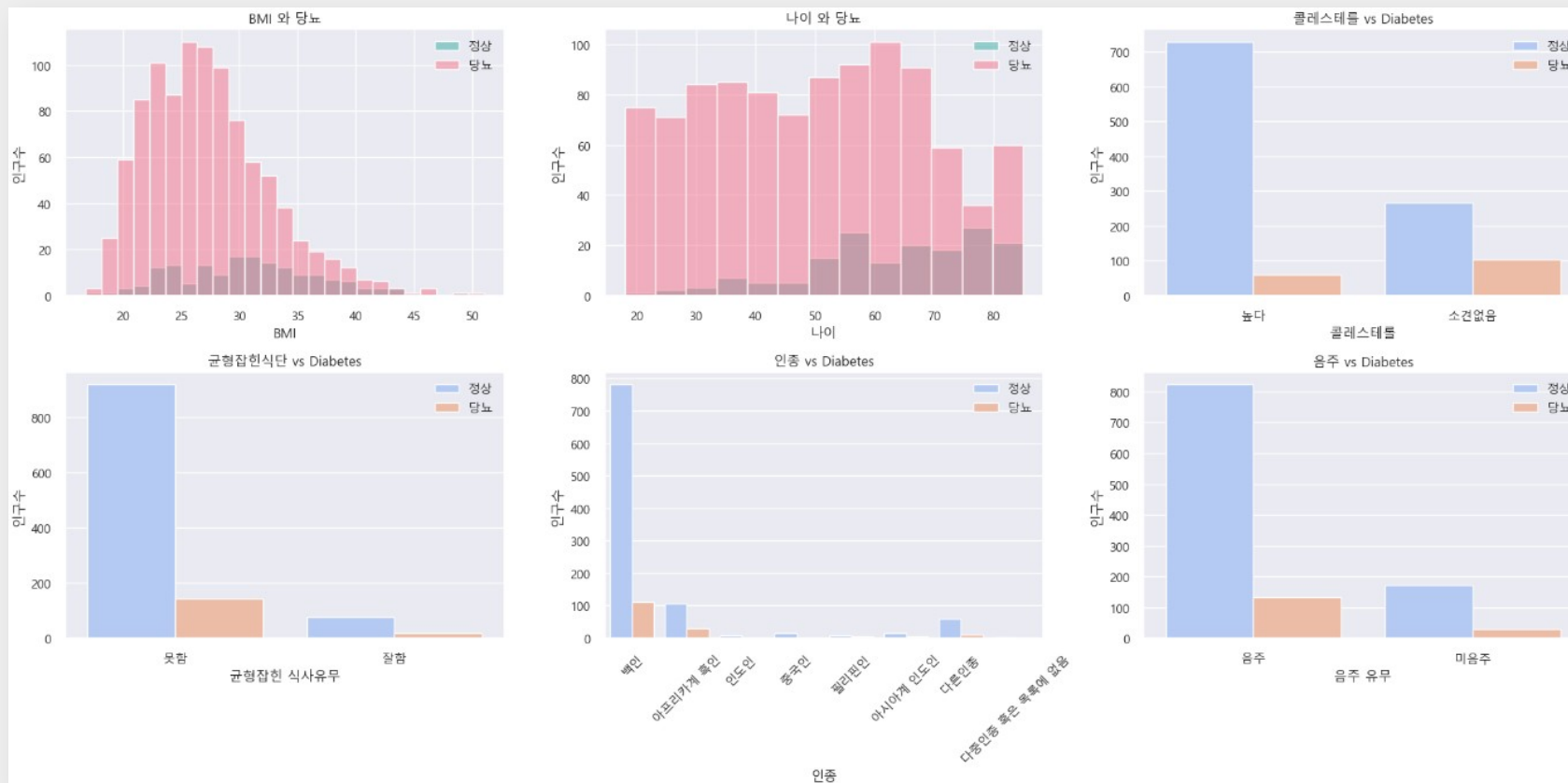
### 1. 당뇨병을 유발하는 주요 원인들



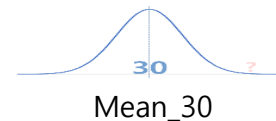
# VI. 결론(2/3)



## 2. 주요 원인별 당뇨 유무 그래프 분석



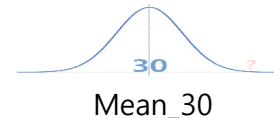
## VI. 결론(3/3)



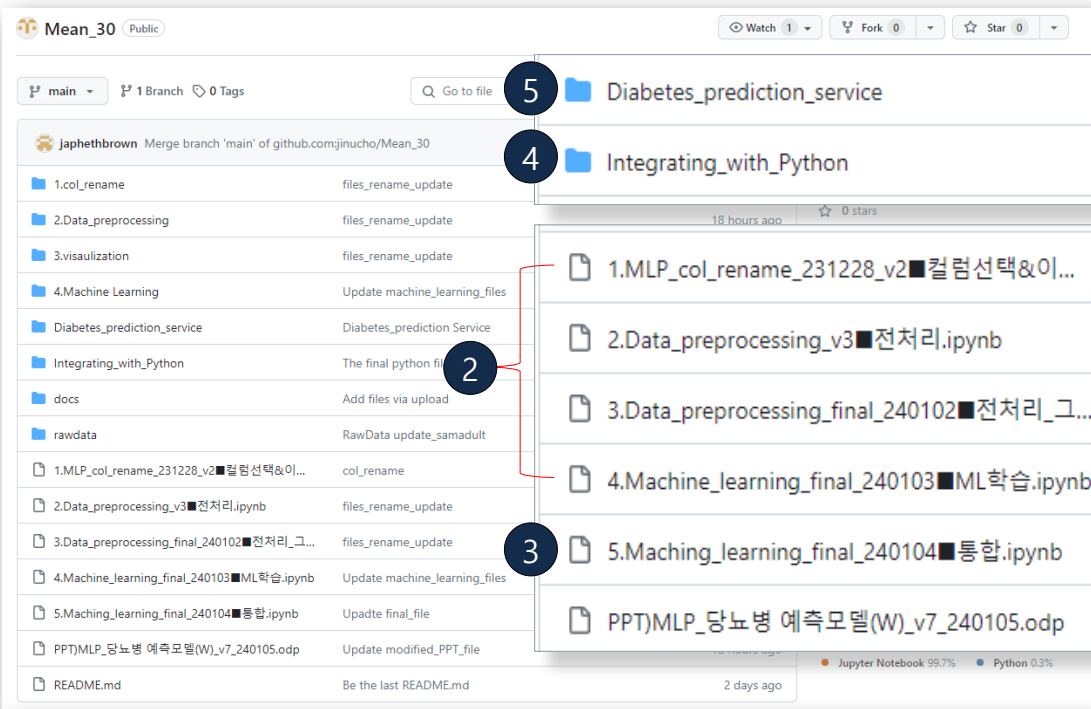
### 3. 당뇨병에 영향을 주는 주요 원인

- 1) BMI : 건강 상태를 나타내는 주요 인자
- 2) 나이 : 노화로 인한 신진대사 감소로 당뇨등의 대사 관련된 질병이 생길 확률이 높다.
- 3) 고혈압 : 당뇨와의 합병증으로 주로 발생
- 4) 콜레스테롤 수치 : BMI와 마찬가지로 건강상태를 나타내는 주요 인자
- 5) 기타 음주 및 평소 식단에 따라 건강 상태에 영향을 주므로 당뇨에 영향이 있다.

## VII. 최종 결과물



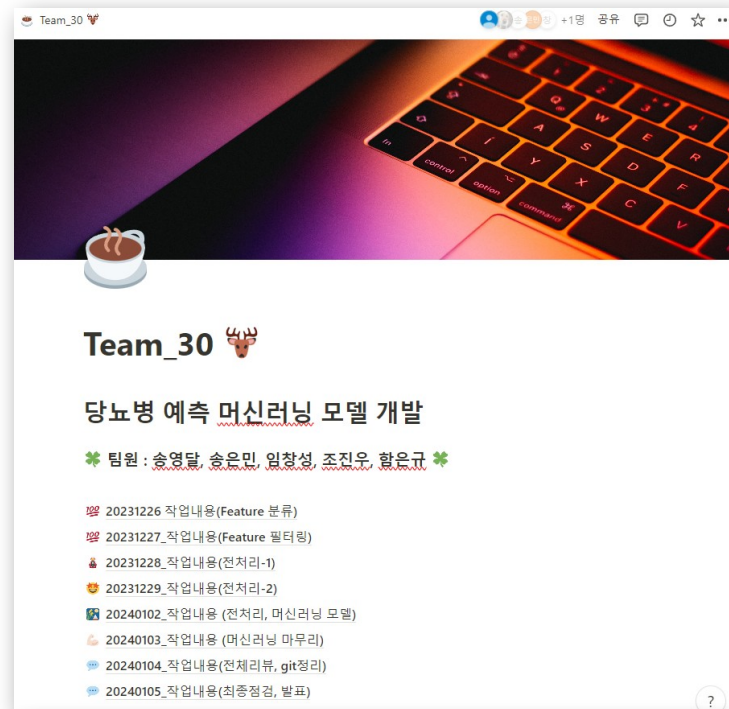
### Git hub



### Git Hub

1. 기초 데이터
2. 개발 내역 (ipynb)
3. 개발 내역 통합본 (ipynb)
4. 자동화 (python → 전처리 파일, 모델링pickle)
5. 서비스화 (설문지+결과 출력)

### Notion



### Notion

1. 개발 계획 및 인원별 업무 분담
2. 진행사항 및 특이사항 공유
3. 개발 이력관리

### 1. 최적화의 한계 : precision 0.32 max

- 1) 정밀도(Precision 값)이 더 이상 높아지지 않음
- 2) 시도 : Validation까지 0.7~0.8로 준수 → 최종 Test에서 0.30~0.32로 형성
  - 컬럼 재정리, 모델 변경과 하이퍼파라미터를 변경과 샘플링 방법 변경 등
- 3) 결과 : 결과값에 변동이 없음
- 4) 대안검토
  - Feature select부터 다시 Building 필요 (의학적 Domain)
  - 원본 데이터의 한계 : 다른 참조문서에서 Recall이 0.11 수준

### 2. 추가 서비스화 단계 필요

- 1) 프로젝트 과정이 전처리, 모델링, 성능검증에 학습에 집중됨
- 2) 실제 설문지를 통한 예측모델 결과 출력을 위해 프로그램 재설계가 필요

### 3. 마무리

- 1) 전처리, 그래프, 그리고 머신러닝까지 아우를 수 있는 좋을 기회였음
- 2) 기술을 공유하며 많이 배울 수 있는 기회였으며 업무 스케줄링 및 진행사항 공유, 보고서 작성 등 실무를 간접적으로 접할 수 있는 좋은 기회였음

### 4. 사용기술

- python, pandas, numpy, matplotlib, sklearn and pycaret etc.

### 5. 자료출처

#### 1) 기초데이터

- [https://www.cdc.gov/nchs/nhis/nhis\\_2018\\_data\\_release.htm](https://www.cdc.gov/nchs/nhis/nhis_2018_data_release.htm)

#### 2) 기술자료

- 월간당뇨 11월호 : 소아 당뇨병이란 무엇인가?
- 연세대학교 의과대학 소아과학교실 : 소아연령에서의 2형 당뇨병의 임상적 고찰

#### 3) 세계 당뇨지도

- <https://www.visualcapitalist.com/cp/diabetes-rates-by-country>

# The E.O.D

# Thanks

Mean\_30

