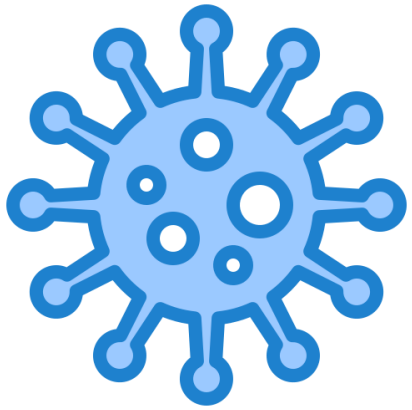
Diabetes Mellitus
Prediction Project

프로젝트 팀원 : 김대영, 박성호

contents

프로젝트 개요

프로젝트 개요



코로나 바이러스

→

실외 활동 감소, 실내 활동 증가

→ 일상 생활 활동량 감소

운동 빈도 감소

→ 체중 증가

배달 음식 및 간편식 소비 증가

→ 영향 불균형

→

당뇨
유병률
증가

프로젝트 개요

당뇨?

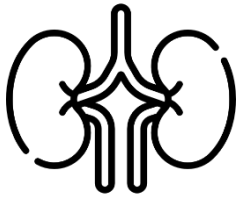
우리 몸이 섭취한 음식물을 적절하게 사용하지 못해
혈당 수치가 정상인보다 훨씬 높은 상태
포도당이 소변으로 빠져 나간다 하여 이름 붙여진 병

지속적인 고혈당 → 당뇨병 만성합병증 →


망막질환


심근경색


뇌졸중


신장질환


동맥경화


신경질환

데이터 소개

데이터

당뇨병 데이터

2438개의 rows

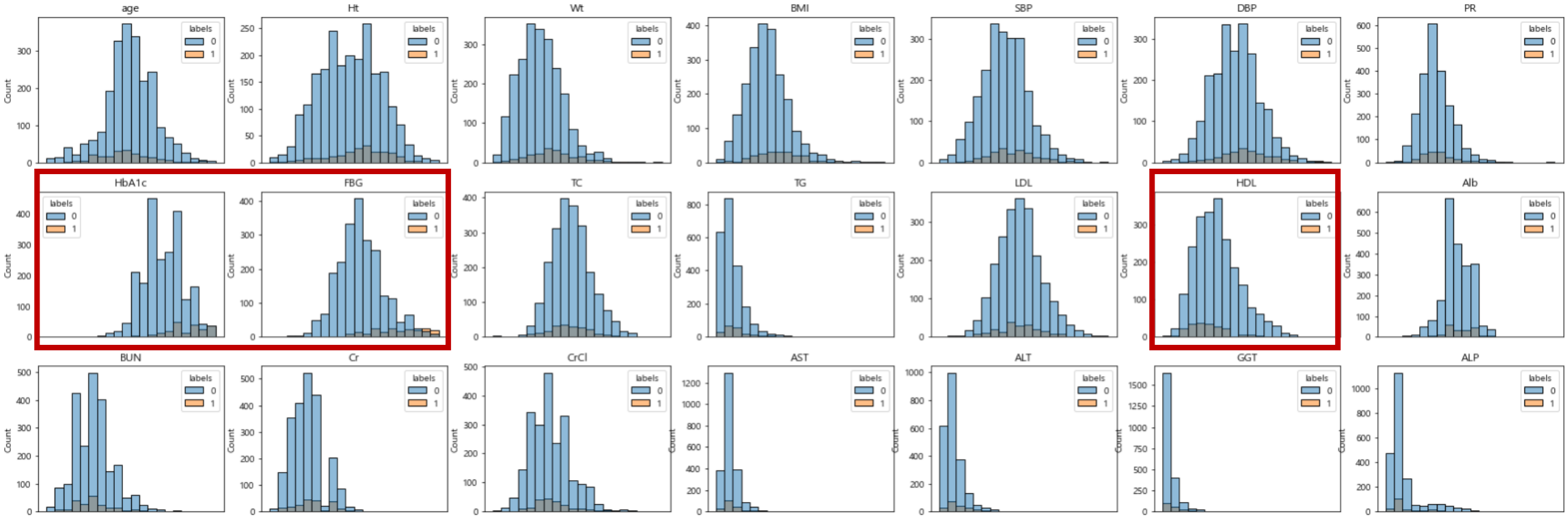
26개의 columns

환자 기본 정보
혈압 수치
콜레스테롤 수치
간수치
신장 수치
당뇨병 여부

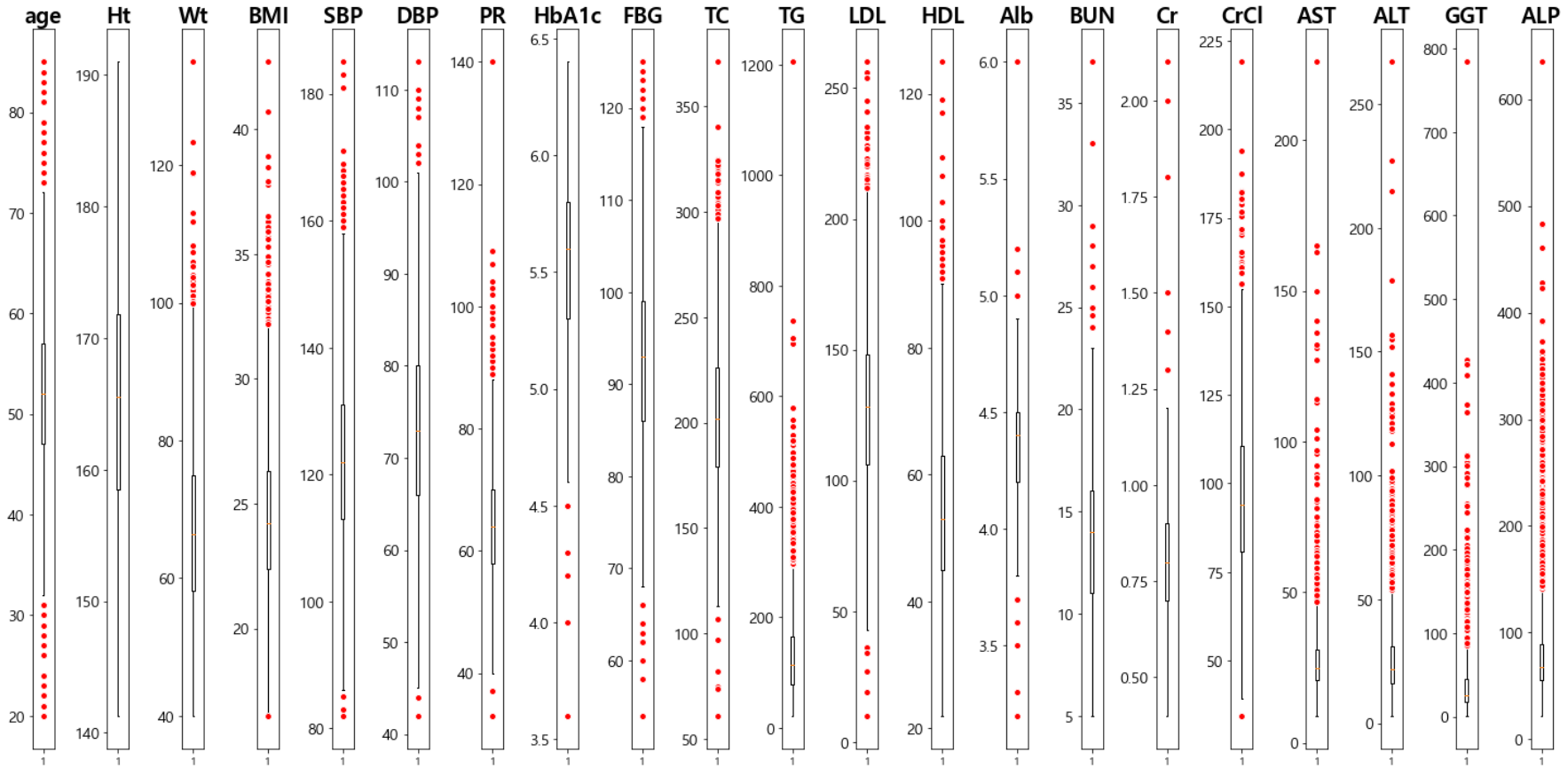
2438개 중 당뇨병이 206개인 불균형 데이터

EDA 및 전처리

변수 분포도 확인


age
Ht
Wt
BMI
SBP
DBP
PR
HbA1c
FBG
TC
TG
LDL
HDL
Alb
BUN
Cr
CrCl
AST
ALT
GGT
ALP
labels
0
1
Count

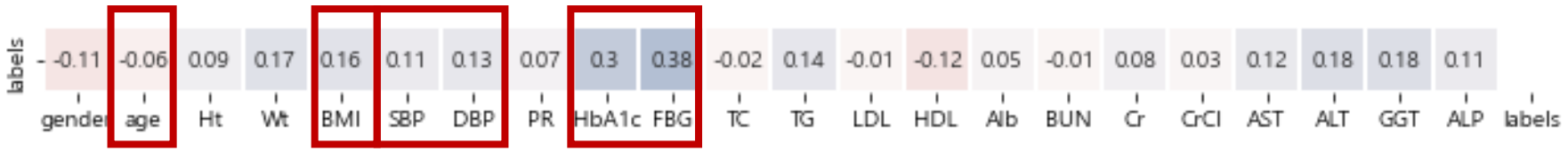
변수 분포도 확인



각 수치들 중 **이상치**가 존재했지만 의학적으로 의미 있는 데이터

EDA 및 전처리
상관관계 확인

변수 상관관계 확인



당뇨병 위험도 측정표

질문	문항		점수
1. 당신의 나이는?	35세 미만		0점
	35~44세		2점
	45세 이상		3점
2. 당신의 부모형제 중 한 명이라도 당뇨병이 있습니까?	아니오		0점
	예		1점
3. 당신은 현재 혈압약을 복용하고 있거나 혈압이 140/90 mm Hg 이상인가요?	아니오		0점
	예		1점
4. 당신의 허리둘레는 얼마인가요?	남자	84 cm 미만	0점
		84 ~ 89.9 cm	2점
		90 cm 이상	3점
	여자	77 cm 미만	0점
		77 ~ 83.9 cm	2점
		84 cm 이상	3점
5. 당신은 현재 담배를 피나요?	아니오		0점
	예		1점
6. 당신의 음주량은 하루 평균 몇 잔 인가요? (술 종류 관계없이)	하루 1잔 미만		0점
	하루 1~4.9잔		1점
	하루 5잔 이상		2점
총점			

※결과해석 : 점수가 높을수록 당뇨병 위험이 높아진다. 8~9점은 5~7점보다 당뇨병 발생 위험이 2배, 10점 이상일 경우 3배 이상 높아진다.
총점이 5점 이상일 경우 당뇨병이 있을 위험이 높으므로 혈당검사(공복혈당 또는 식후혈당)가 권고된다.

AGE(나이)

나이와 약한 상관 관계

BMI(체질량 지수)

혈압과 약한 양의 상관관계

SBP(수축기혈압), DBP(이완기 혈압)

혈압과 약한 양의 상관관계

HbA1c(당화혈색소), FBG(공복 혈당)

혈압과 상관 관계가 존재

데이터 전처리

데이터 범주/그룹화

```
한국인의 체질량 지수에 따른 비만 판정 기준
- <18.5: 저체중
- 18.5-22.9: 정상
- 23.0-24.9: 비만 전단계(과체중)
- 25.0-29.9: 1단계 비만
- 30.0-34.9: 2단계 비만
- 35.0 이상: 3단계 비만
```

실제 사용하는 검사 수치에 대한
정상 및 그 외 진단 기준에 맞춰 범주화

데이터 NULL값 처리

	PR	TG	LDL	HDL	Alb	ALP	labels
184	NaN	138.0	114.0	46.0	4.7	81.0	0
254	NaN	NaN	NaN	NaN	NaN	NaN	0
657	NaN	201.0	119.0	41.0	4.2	62.0	0
1649	NaN	41.0	105.0	81.0	4.7	61.0	0
2002	66.0	NaN	83.0	47.0	4.3	84.0	0
2068	NaN	94.0	121.0	51.0	4.3	253.0	0

PR(맥박)은 나이대별 중앙값을 이용해 처리
TG(중성지방)은 아래 식을 이용해 처리

$$TC = HDL + LDL + TC/5$$

254번 index는 삭제

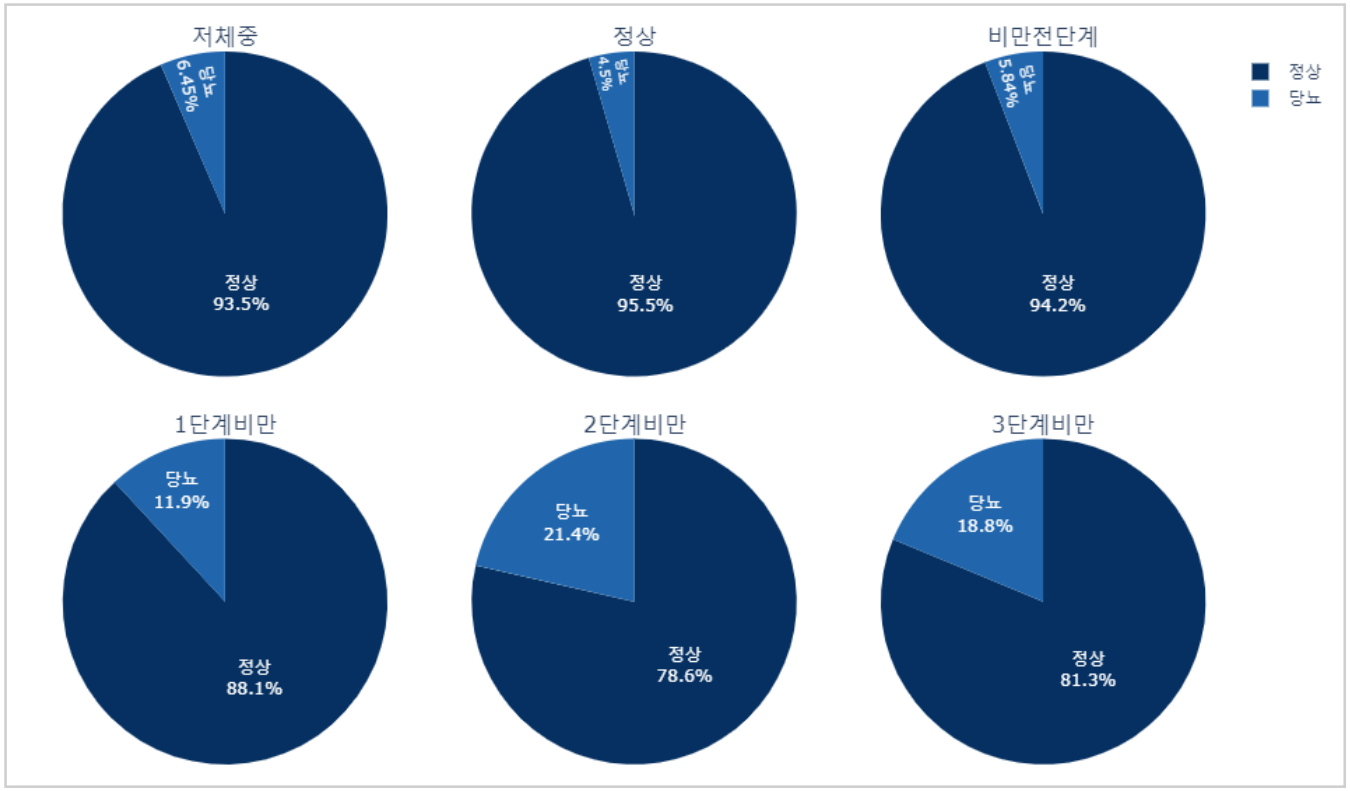
파생변수 생성

- TC 그룹(240기준 이상/미만)
- 글루코스 그룹
- 맥압: 수축기와 이완기의 혈압 차이
- BUN/Cr ratio: 신장 질환의 여부 파악
- AST/ALT: 간 기능의 데미지를 파악

당뇨병과 연관된 질병들의 수치를 파악해
관련 값들을 산식을 이용해 파생 변수 생성

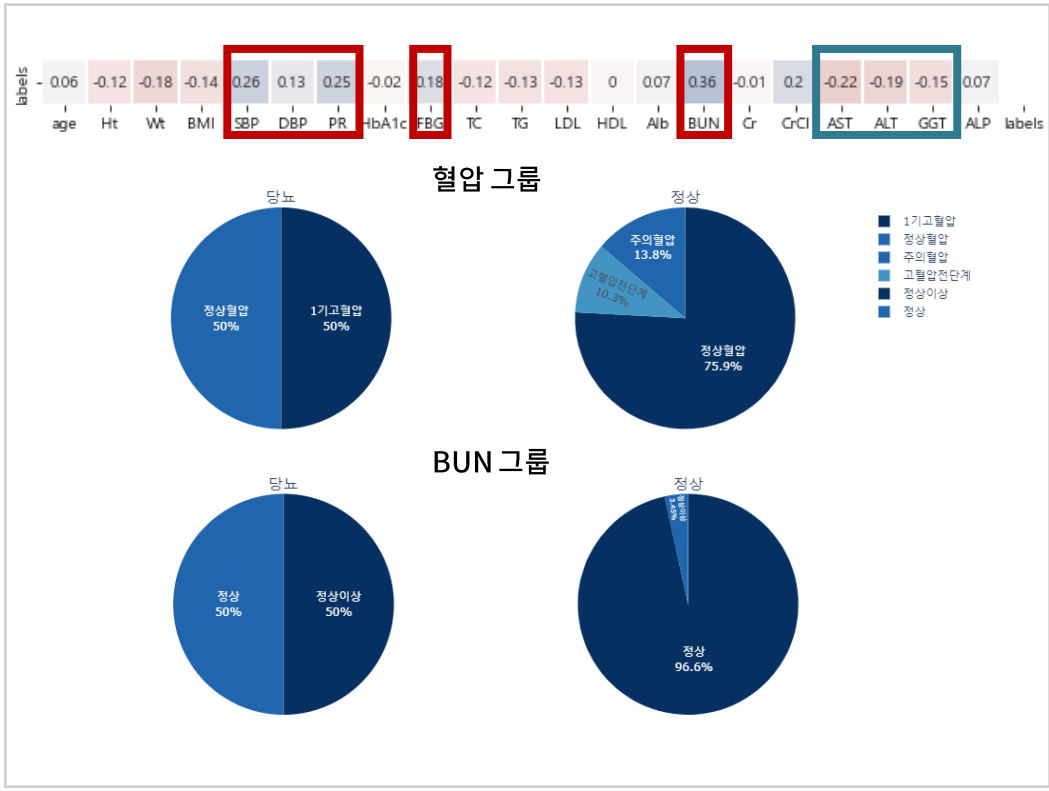
EDA 및 전처리
탐색적 데이터 분석

BMI 그룹 분포



저체중에서 당뇨 비율이 높은 편
비만율이 높을수록 당뇨병의 위험율 증가

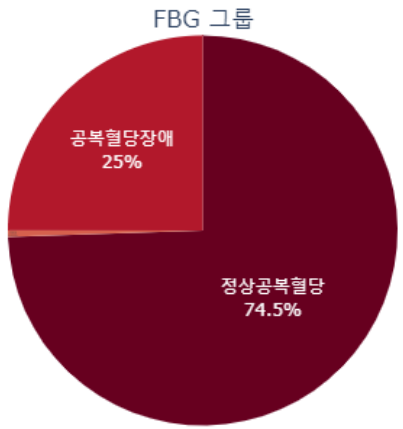
저체중 데이터 분포

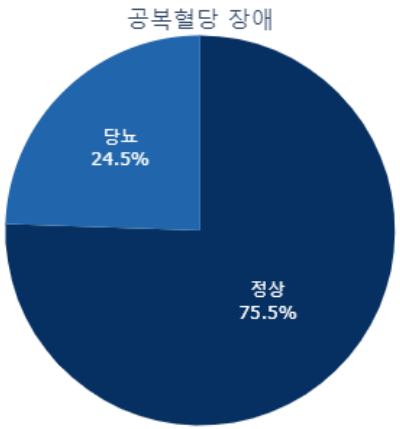


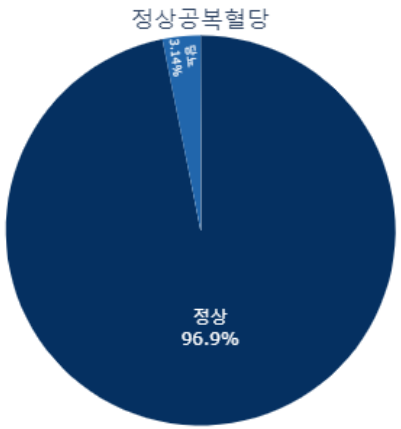
저체중 인원 31명 중 2명이 당뇨병
수치들을 확인 했을 때 뚜렷한 연관성 없음

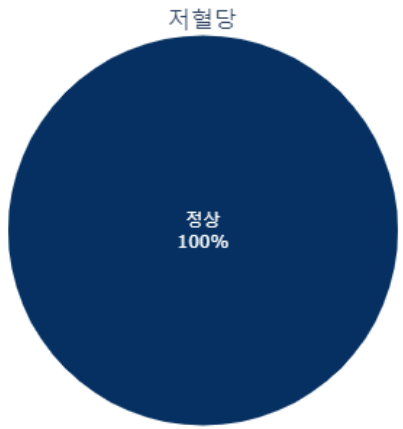
EDA 및 전처리
탐색적 데이터 분석

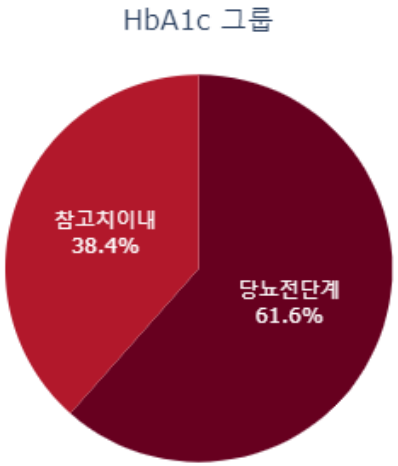
혈당 그룹 분포

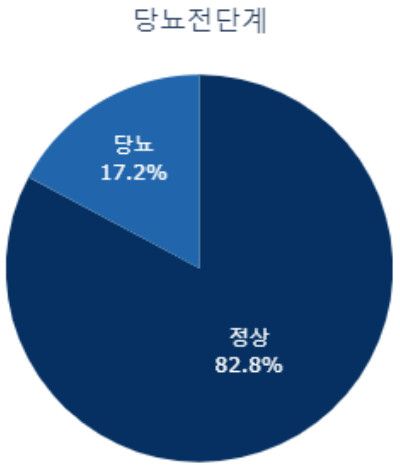


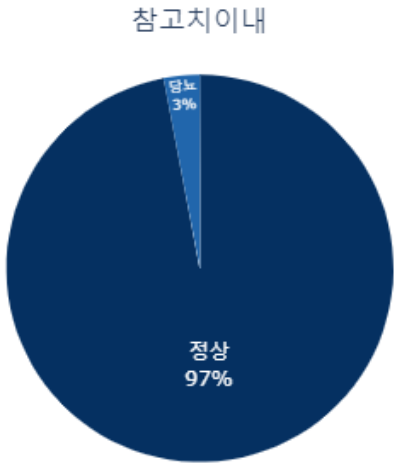












당뇨 진단 기준

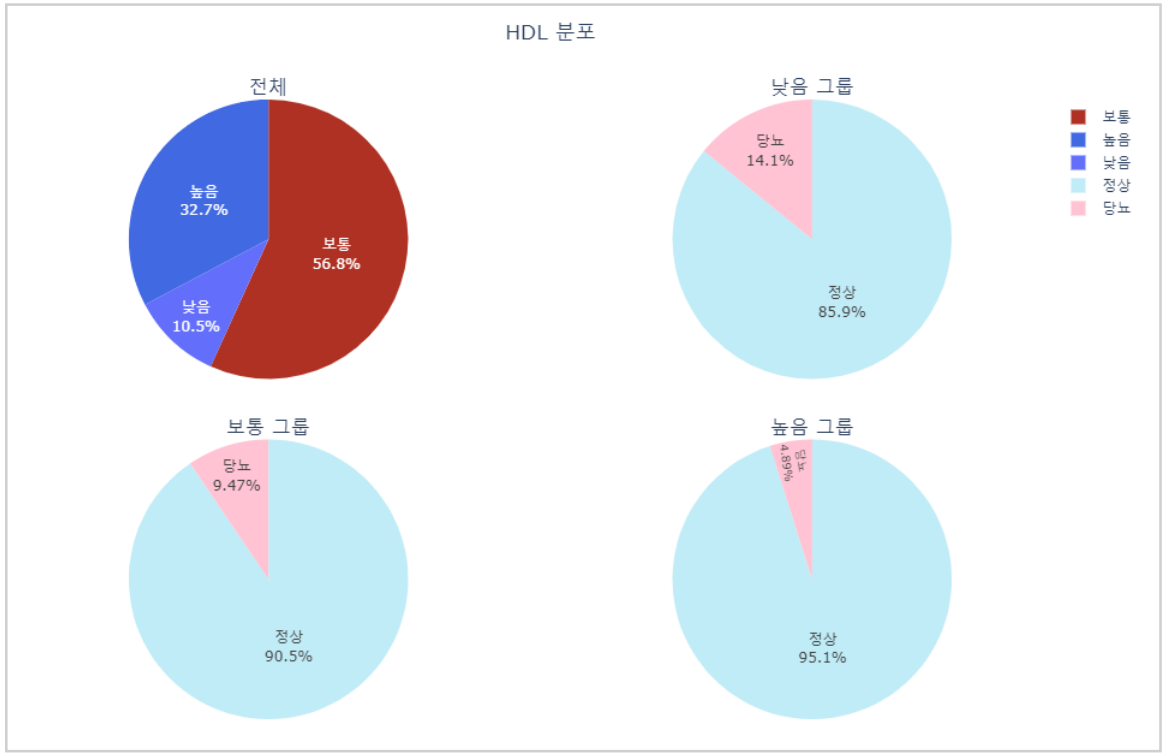
HbA1c(당화혈색소) ≥ **6.5%**: 당뇨
FBG(공복 혈당) ≥ **126**: 당뇨

당뇨로 진단 되는 데이터는 존재하지 않음

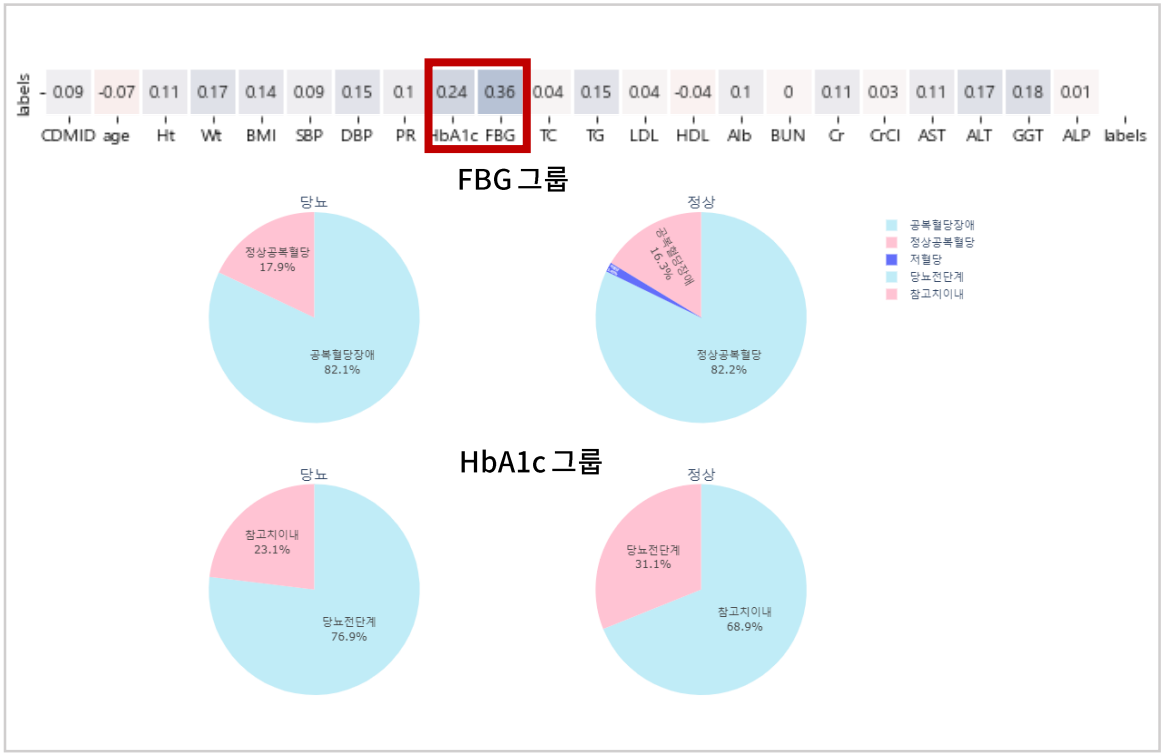
저혈당 그룹에선 당뇨병이 존재하지 않음

EDA 및 전처리
탐색적 데이터 분석

HDL 그룹 분포



HDL 높음 데이터 분포



HDL 높음 그룹은 당뇨가 없을 줄 알았지만
그 안에서 **혈당 수치**가 높은 데이터가 존재

모델링

모델링 과정

데이터
전처리

오버
샘플링

파라미터
튜닝

데이터 처리 기법 및 사용 분류 모델

인코딩/스케일링	오버 샘플링	예측 모델
Label Encoding	Random Oversampling	Logistic Regression
One-Hot Encoding	SMOTE	Gradient Boosting
Standard Scaling	SMOTE-NC	Decision Tree
	SMOTE-N	Random Forest
	ADASYN	XGBoost
		KNN
		SVC
		MLP
		LightGBM

해당 **기법**과 **모델**들을 사용해 당뇨병 예측

모델링 과정

1. 데이터 전처리

라벨 인코딩 - 각 수준에 따라서 값들을 수치화

	BMI_g	BP_g	PP_g	PR_g	HbA1c_g	FBG_g	TC_g	TG_g	LDL_g	HDL_g
0	1	0	1	1	1	1	2	0	4	2
1	1	2	1	1	1	2	2	0	4	1
2	1	0	1	1	1	1	2	0	4	1
3	1	0	1	1	0	1	0	0	0	1
4	2	0	0	1	1	1	2	2	3	1

스탠다드 스케일링 - 수치형 데이터를 정규화

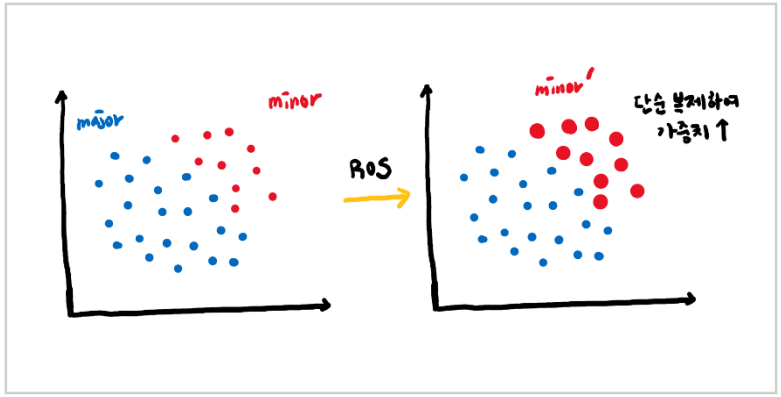
	age	Ht	Wt	BMI	SBP	DBP	PR	HbA1c	FBG	TC
0	59	159.0	56.0	22.2	99	59	60.0	5.8	94	269
1	51	160.0	56.0	21.9	140	85	73.0	6.4	111	272
2	36	162.0	56.0	21.2	100	55	60.0	6.0	91	287
3	53	150.0	47.2	21.0	105	62	78.0	5.5	86	164
4	61	168.1	66.1	23.4	104	67	88.0	5.7	94	252

원핫 인코딩 – 각 수준의 개수만큼 컬럼을 만들어 해당 값은 1 / 아닌 값은 0으로 수치화

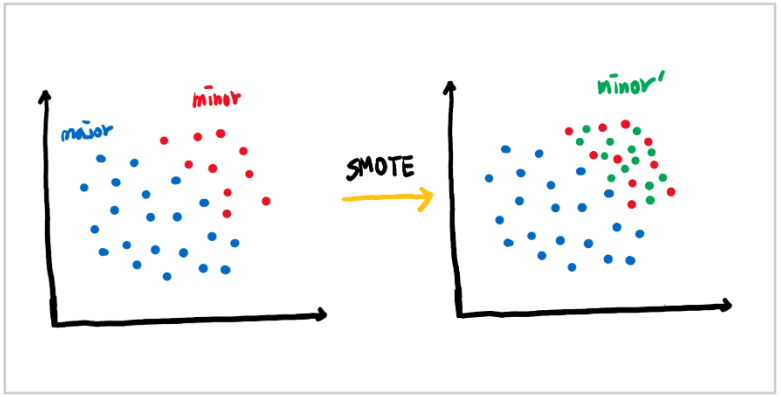
	gender_F	gender_M	age_g_30-44세	age_g_35세미만	age_g_45세이상	BMI_g_1단계비만	BMI_g_2단계비만	BMI_g_3단계비만	BMI_g_비만전단계	BMI_g_저체중
0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
4	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0

2. 오버 샘플링

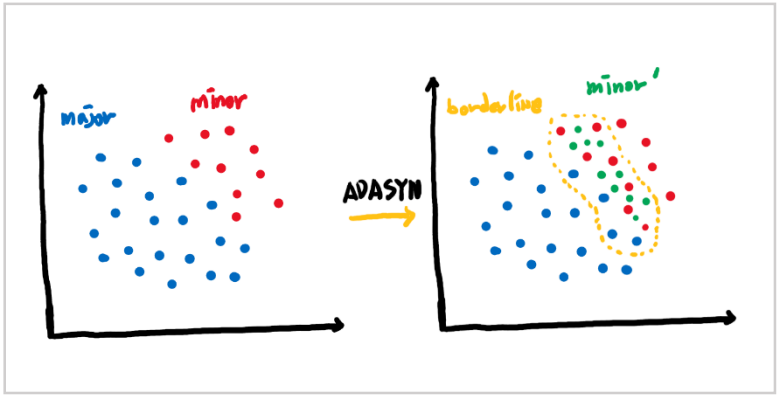
Random Sampling



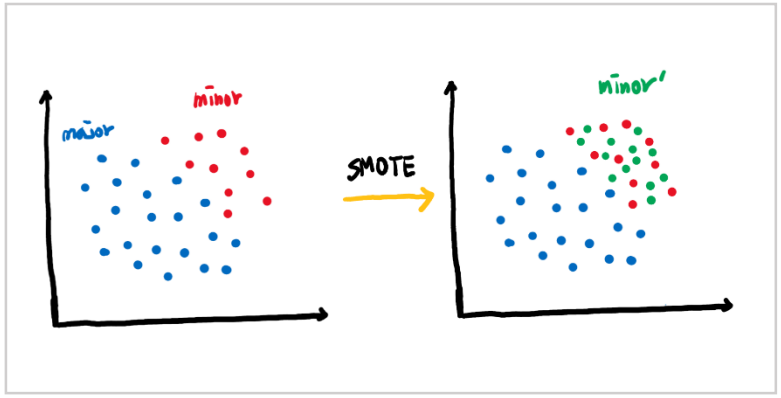
SMOTE



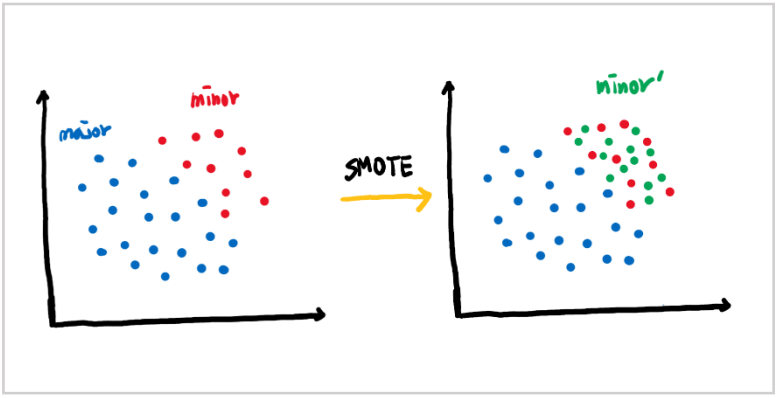
ADASYN



SMOTE-NC



SMOTE-N



3. Parameter tuning

```python
# 분류기의 성능을 return하는 함수 작성
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
def get_clf_eval(y_test, pred):
    acc = accuracy_score(y_test, pred)
    pre = precision_score(y_test, pred)
    re = recall_score(y_test, pred)
    f1 = f1_score(y_test, pred)
    auc = roc_auc_score(y_test, pred)
    return acc, pre, re, f1, auc

# 모델과 데이터를 주면 성능을 출력하는 함수
def get_result(model, X_train, y_train, X_test, y_test):
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    return get_clf_eval(y_test, pred)

# 다수의 모델의 성능을 정리해서 DataFrame으로 반환하는 함수
def get_result_pd(models, model_names, X_train, y_train, X_test, y_test):
    col_names = ['accuracy', 'precision', 'recall', 'f1', 'roc_auc']
    tmp =[]
    for model in models:
        tmp.append(get_result(model, X_train, y_train, X_test, y_test))
    return pd.DataFrame(tmp, columns=col_names, index=model_names)
```

3. Parameter tuning

```
dt_clf = DecisionTreeClassifier(random_state=13, max_depth=4)
knn = KNeighborsClassifier(n_neighbors=5)
rf_clf = RandomForestClassifier(random_state=13, max_depth=4, n_jobs=-1, n_estimators=1000)
lr_clf = LogisticRegression(max_iter=400, random_state=13, C=1, solver='lbfgs')
lgbm_clf = LGBMClassifier(random_state=13, n_estimators=1000, n_jobs=-1)
svc = SVC(kernel='linear', random_state=13)
gb_clf = GradientBoostingClassifier(random_state=13, learning_rate=0.1, n_estimators=200)
xgb = XGBClassifier(random_state=13, n_estimators=400, learning_rate=0.1, n_jobs=-1, use_label_encoder=False)
mlp = MLPClassifier(hidden_layer_sizes=(11,11,11), activation='relu', max_iter=500, batch_size=32, learning_rate_init=0.1, random_state=13)
```

```
models = [dt_clf, knn, rf_clf, lr_clf, svc, lgbm_clf, gb_clf, xgb, mlp]
model_names = ['DecisionTree', 'KNN', 'RandomForest', 'LogisticReg', 'SVC', 'LightGBM', 'GradientBoost' ,'XGBoost', 'MLP']

start_time = time.time()
results = get_result_pd(models, model_names, X_train_oversampled, y_train_oversampled, X_test, y_test)

print('Fit time : ', time.time() - start_time)
results
```

3. Parameter tuning

```
s_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=13)
```

```
# random forest
param_rf = {'n_estimators': [200,600,1000], 'max_depth': [2, 4, 6, 10], 'min_samples_leaf': [1,2,3]}
grid_rf = GridSearchCV(estimator=RandomForestClassifier(random_state=13), param_grid=param_rf, scoring='recall', cv=s_kfold, n_jobs=-1)

# logistic regression
param_lg = {'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000], 'max_iter':[200,400,600], 'solver':['newton-cg', 'lbfgs', 'liblinear']}
grid_lg = GridSearchCV(estimator=LogisticRegression(random_state=13), param_grid=param_lg, scoring='recall', cv=s_kfold, n_jobs=-1)

# support vector machine
param_svm = {'C': [0.01,0.1,1,10,100,1000], 'gamma': [1,0.1,0.01,0.001,0.0001],'kernel': ['rbf','poly','sigmoid','linear']}
grid_svm = GridSearchCV(estimator=SVC(random_state=13), param_grid=param_svm, scoring='recall', cv=s_kfold, n_jobs=-1)

# multilayer perceptrons
param_mlp = {'max_iter':(200,500,800,1000), 'batch_size':[16,32,64,128], 'learning_rate_init':[0.001,0.01,0.1,1]}
grid_mlp = GridSearchCV(estimator=MLPClassifier(hidden_layer_sizes=(11,11,11), activation='relu', random_state=13), param_grid=param_mlp, scoring='recall', cv=s_kfold, n_jobs=-1)
```

```
grid_models = [grid_rf, grid_lg, grid_svm, grid_mlp]
grid_model_names = ['RandomForest', 'LogisticReg', 'SVC', 'MLP']
# grid_search.fit(X_train_oversampled, y_train_oversampled)
# grid_search.predict(X_train)
pm_tuning_result = get_result_pd(grid_models, grid_model_names, X_train_oversampled, y_train_oversampled, X_test, y_test)
pm_tuning_result
```

3. Parameter tuning

```
from sklearn.model_selection import GridSearchCV, KFold, StratifiedKFold, cross_val_score
from sklearn.metrics import make_scorer
import pprint

# Random Forst 모델 하이퍼파라미터 튜닝
param_grid = {'n_estimators': [200,600,1000], 'max_depth': [2, 4, 6, 10, 15, 20], 'min_samples_leaf': [1,2,3]}
s_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=13)
scoring = ['accuracy','precision','recall']
grid_search = GridSearchCV(estimator=RandomForestClassifier(),
                           param_grid=param_grid,
                           cv=s_kfold, n_jobs=-1, #scoring='f1')
                           scoring=scoring, refit='accuracy', return_train_score=True)

pm_tuning_result = get_result_pd([grid_search], ['random_forest'], X_train_oversampled, y_train_oversampled, X_test, y_test)
pm_tuning_result
```

모델링 결과

모델링 결과

범주형+라벨 인코딩

	accuracy	precision	recall	f1	rou_auc
DecisionTree	0.241803	0.093827	0.926829	0.170404	0.552900
KNN	0.415984	0.103896	0.780488	0.183381	0.581519
RandomForest	0.084016	0.084016	1.000000	0.155009	0.500000
LogisticReg	0.223361	0.097619	1.000000	0.177874	0.576063
SVC	0.202869	0.093458	0.975610	0.170576	0.553800
LightGBM	0.084016	0.084016	1.000000	0.155009	0.500000
GradientBoost	0.084016	0.084016	1.000000	0.155009	0.500000
XGBoost	0.846311	0.282051	0.536585	0.369748	0.705653
MLP	0.663934	0.163934	0.731707	0.267857	0.694713

범주형+원핫 인코딩

	accuracy	precision	recall	f1	roc_auc
DecisionTree	0.754098	0.207407	0.682927	0.318182	0.721777
KNN	0.788934	0.196078	0.487805	0.279720	0.652180
RandomForest	0.829918	0.298077	0.756098	0.427586	0.796393
LogisticReg	0.807377	0.280992	0.829268	0.419753	0.817319
SVC	0.786885	0.248000	0.756098	0.373494	0.772903
LightGBM	0.875000	0.222222	0.195122	0.207792	0.566241
GradientBoost	0.844262	0.307692	0.682927	0.424242	0.770994
XGBoost	0.875000	0.272727	0.292683	0.282353	0.610547
MLP	0.795082	0.229358	0.609756	0.333333	0.710918

연속형+스탠다드 스케일링

	accuracy	precision	recall	f1	roc_auc
DecisionTree	0.803279	0.256637	0.707317	0.376623	0.759699
KNN	0.795082	0.234234	0.634146	0.342105	0.721995
RandomForest	0.827869	0.287129	0.707317	0.408451	0.773122
LogisticReg	0.819672	0.284404	0.756098	0.413333	0.790800
SVC	0.801230	0.262712	0.756098	0.389937	0.780733
LightGBM	0.928279	0.588235	0.487805	0.533333	0.728242
GradientBoost	0.893443	0.396226	0.512195	0.446809	0.720303
XGBoost	0.907787	0.454545	0.487805	0.470588	0.717057
MLP	0.846311	0.326531	0.780488	0.460432	0.816418

모델링 결과

범주형+원핫 인코딩 모델링 결과

SMOTE-NC 결과

	accuracy	precision	recall	f1	roc_auc
DecisionTree	0.805102	0.256402	0.654936	0.366050	0.736926
KNN	0.798127	0.242523	0.650523	0.353203	0.731138
RandomForest	0.846952	0.312032	0.673868	0.425199	0.768356
LogisticReg	0.832582	0.307966	0.790708	0.443049	0.813555
SVM	0.813707	0.283143	0.786063	0.416229	0.801144
LightGBM	0.924084	0.568782	0.417305	0.479897	0.694084
GradientBoost	0.906030	0.451110	0.475494	0.461802	0.710626
XGBoost	0.918752	0.518253	0.460976	0.486695	0.710990
MLP	0.725561	0.230137	0.751568	0.342647	0.737251

Random Oversampling 결과

	accuracy	precision	recall	f1	roc_auc
DecisionTree	0.801818	0.250029	0.669454	0.363370	0.741721
KNN	0.844070	0.257884	0.451220	0.328126	0.665770
RandomForest	0.851055	0.323042	0.693496	0.439684	0.779516
LogisticReg	0.826014	0.300182	0.795819	0.435692	0.812298
SVM	0.822326	0.296800	0.805459	0.433585	0.814653
LightGBM	0.924493	0.593333	0.324971	0.419290	0.652400
GradientBoost	0.890444	0.397148	0.567828	0.466402	0.744022
XGBoost	0.925730	0.581969	0.422184	0.488748	0.697196
MLP	0.647375	0.158408	0.649710	0.240145	0.648428

모델링 결과

연속형+스탠다드 스케일링 모델링 결과

SMOTE-NC 결과

	accuracy	precision	recall	f1	roc_auc
DecisionTree	0.815343	0.263421	0.654472	0.374164	0.742288
KNN	0.805919	0.252803	0.660046	0.365516	0.739709
RandomForest	0.850235	0.319801	0.678862	0.433220	0.772420
LogisticReg	0.835452	0.308617	0.771080	0.440197	0.806203
SVM	0.819448	0.288177	0.776074	0.419961	0.799732
LightGBM	0.926137	0.588372	0.412427	0.484294	0.692991
GradientBoost	0.908081	0.459526	0.490012	0.473180	0.718334
XGBoost	0.915471	0.496591	0.432056	0.460305	0.696081
MLP	0.807120	0.265762	0.693961	0.381621	0.755767

Random Oversampling 결과

	accuracy	precision	recall	f1	roc_auc
DecisionTree	0.791983	0.244483	0.684088	0.358684	0.742999
KNN	0.849001	0.271513	0.465505	0.342689	0.674935
RandomForest	0.850240	0.323558	0.698606	0.441416	0.781399
LogisticReg	0.827653	0.302487	0.795819	0.438115	0.813193
SVM	0.822736	0.299434	0.819977	0.438390	0.821465
LightGBM	0.926959	0.629316	0.329849	0.430269	0.655961
GradientBoost	0.891265	0.399382	0.562718	0.466051	0.742138
XGBoost	0.921625	0.544293	0.412660	0.468768	0.690641
MLP	0.732515	0.213181	0.766551	0.331439	0.747902

연속형+스탠다드 스케일링 모델링 결과

SMOTE-NC 결과

	accuracy	precision	recall	f1	roc_auc
DecisionTree	0.804273	0.255323	0.684088	0.371141	0.749706
KNN	0.802635	0.244865	0.640534	0.354161	0.729055
RandomForest	0.851050	0.320889	0.683740	0.436027	0.775085
LogisticReg	0.849807	0.339561	0.815331	0.479230	0.834156
SVM	0.838323	0.322704	0.824855	0.463689	0.832196
LightGBM	0.924906	0.571968	0.427294	0.486781	0.699079
GradientBoost	0.915062	0.495540	0.504646	0.499091	0.728791
XGBoost	0.923675	0.554589	0.480488	0.511331	0.722537
MLP	0.811259	0.283294	0.699187	0.394827	0.760415

Random Oversampling 결과

	accuracy	precision	recall	f1	roc_auc
DecisionTree	0.782126	0.237562	0.703600	0.353835	0.746477
KNN	0.850635	0.272778	0.450987	0.339497	0.669246
RandomForest	0.855980	0.334881	0.708130	0.453687	0.788850
LogisticReg	0.838732	0.323084	0.825087	0.464090	0.832533
SVM	0.830528	0.310820	0.820209	0.450451	0.825838
LightGBM	0.931060	0.659448	0.388037	0.486916	0.684606
GradientBoost	0.901107	0.432560	0.567596	0.489062	0.749735
XGBoost	0.924494	0.564828	0.431823	0.487712	0.700897
MLP	0.677393	0.209973	0.883624	0.334608	0.771018

한계 및 과제

한계 및 과제

한계

데이터 부족

변수 설명 부재

도메인 지식 부족

분류 범위 기준 비획일화

한계 및 과제

과제

다른 분류 범위 기준 적용해 모델 성능 확인

불균형 및 연속/범주형 혼합 데이터 처리 기법 연구

참조

Reference

https://www.e-jkd.org/upload/pdf/jkd-2020-21-1-27.pdf
http://www.lecturernews.com
https://www.koreascience.or.kr/article/JAKO200560537773551.pdf
http://www.monews.co.kr/news/articleView.html?idxno=203842
http://www.docdocdoc.co.kr
https://ko.wikipedia.org/wiki/%EB%8B%B9%EB%87%A8%EB%B3%91
https://smtmap.com/%EA%B0%84%EC%88%98%EC%B9%98/
http://guro.kumc.or.kr/dept/main/index.do?DP_CODE=GRCP&MENU_ID=003036050045
https://www.koreascience.or.kr/article/JAKO201354840931827.pdf
https://www.schlab.org/guide/item/261/
https://m.khan.co.kr/life/health/article/201511101537195#c2b
http://seoulnim.com/news/lecture_v.asp?srno=7628&page=70&gubun=&keyword=
https://blog.naver.com/PostView.naver?blogId=i-doctor&logNo=221450543662
https://www.joongang.co.kr/article/21657194#home
https://www.ibric.org/myboard/view.php?Board=review0&id=529&filename=bc0602.pdf&fidx=2&mode=down
https://blog.naver.com/hyouncho2/60170417299
https://www.paik.ac.kr/busan/medicine/disease_info_view.asp?p_sid=1040&p_cate=A
https://www.cheric.org/PDF/PIC/PC19/PC19-2-0085.pdf
https://labtestsonline.kr/tests
https://m.amc.seoul.kr/asan/mobile/healthinfo/
https://amc.seoul.kr/asan/healthinfo/
https://kormedi.com/
http://drug.co.kr/abbreviation/
https://m.blog.naver.com/sorak123/222052778113
http://guro.kumc.or.kr/
https://medgongbu.tistory.com/92
https://wyatt37.tistory.com/10
http://www.koreanhypertension.org/