보고서

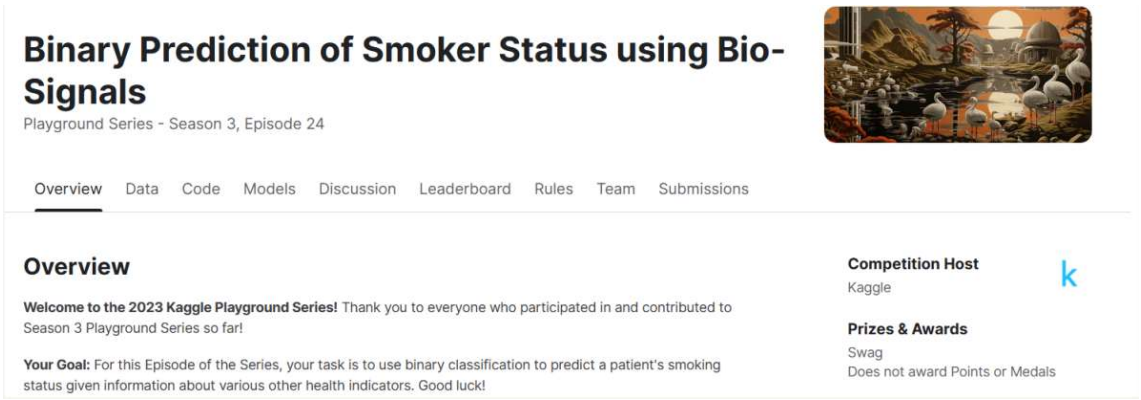
생체신호를 이용한 흡연 상태 이진 예측

과목명 : AI를 위한 머신러닝
담당교수명 : 곽일엽 교수님
팀명 : 팀3_SMOKE
팀원 : 강하람, 박지후, 유지원, 이주영, 정서현

[MLforAI] Kaggle Competition (Binary Prediction of Smoker Status using Bio-Signals)

Contents

1.Introduction



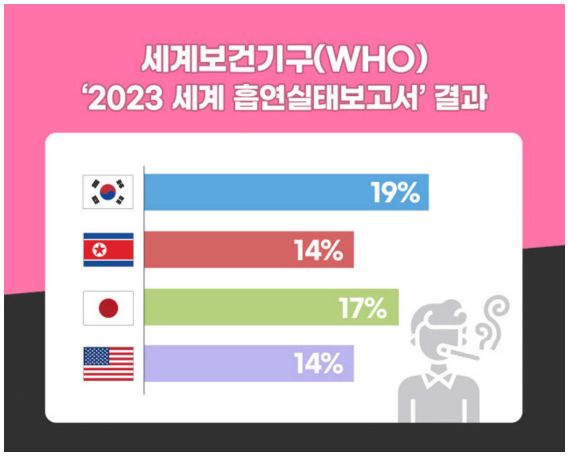
- 대회목표

건강 상태에 관련된 데이터들을 보고 해당 인원의 흡연 여부를 판별하는 알고리즘 제작하는 프로젝트

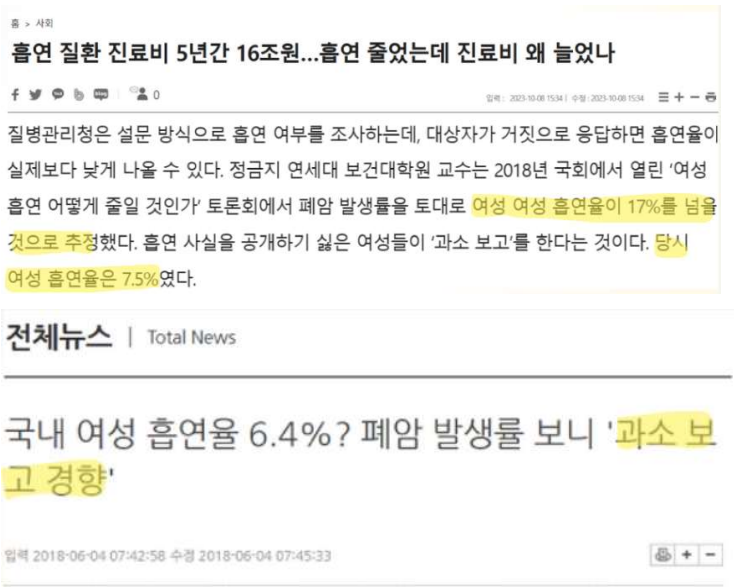
- 평가

area under the ROC curve를 통해 평가.

- 건강 상태에 관련된 데이터를 통해 흡연 여부 판별을 해야 하는 이유 (대회 참여 계기)



2023년 세계보건기구(WHO)의 세계 흡연 현황 보고서에 따르면, 대한민국 성인의 흡연율은 19%로 다른 나라들에 비해 비교적 높은 수치를 보이고 있습니다. 이는 대한민국에서 흡연이 여전히 개선이 필요한 중요한 문제임을 나타냅니다. 흡연 상태를 정확하게 예측하는 것은 흡연과 관련된 다양한 건강 및 경제적 문제를 해결하는 데 중요합니다.



홈 > 사회

흡연 질환 진료비 5년간 16조원...흡연 줄었는데 진료비 왜 늘었나

질병관리청은 설문 방식으로 흡연 여부를 조사하는데, 대상자가 거짓으로 응답하면 흡연율이 실제보다 낮게 나올 수 있다. 정금지 연세대 보건대학원 교수는 2018년 국회에서 열린 '여성 흡연 어떻게 줄일 것인가' 토론회에서 폐암 발생률을 토대로 여성 여성 흡연율이 17%를 넘을 것으로 추정했다. 흡연 사실을 공개하기 싫은 여성들이 '과소 보고'를 한다는 것이다. 당시 여성 흡연율은 7.5%였다.

전체뉴스 | Total News

국내 여성 흡연율 6.4%? 폐암 발생률 보니 '과소 보고 경향'

입력 2018-06-04 07:42:58 수정 2018-06-04 07:45:33

물론, 병원에서 실시하는 건강 체크리스트를 통해 사람들의 흡연 상태를 예측할 수 있습니다. 하지만 이 방법은 **자발적인 정보 제공**에 의존하기 때문에, **흡연률이 실제보다 낮게 보고되는 경향**이 있습니다. 예를 들어, 대한민국에서 공식적으로 보고된 여성 흡연율은 6.4%였지만, 폐암 발생률 분석을 통해 실제 흡연율은 7.3%로 추정되었습니다. 이러한 '과소보고'는 **흡연율의 정확한 측정을 복잡하게 하고 흡연과 직접적으로 연관된 질병의 부담을 증가**시킵니다.

지난 5년간 흡연으로 인한 건강보험 진료비는 16조 3982억원에 달하며, 이는 국가 건강보험 재정에 큰 부담을 주고 있습니다. 또한, **흡연 관련 질병이 증가**하고, **금연 치료 이수율이 감소**하는 추세를 보이고 있으며, 이는 **사회경제적 비용 증가**로 이어집니다.

따라서, 우리 팀은 흡연과 관련된 질병 부담을 해결하기 위해 개인의 건강 상태 데이터를 기반으로 흡연 상태를 결정하는 알고리즘을 만들었습니다.

2. Method

(1) Pycaret (autoML)

a) Pycaret이란?

PyCaret은 적은 코드로 머신러닝 워크 플로우를 자동화하는 오픈 소스 라이브러리입니다. 머신러닝 모델 개발시 많은 시간을 소요했던 코딩, 전처리, 모델 선택, 파라미터 튜닝 작업을 자동화해주어 쉽고, 높은 생산성의 작업을 가능하게 합니다.

b) Setting

우선 다루고자 하는 모델의 특성에 맞는 모듈을 import해줄 수 있는데 여기서는 분류 모델을 사용합니다.

```
from pycaret.classification import *
```

다음으로 모델을 돌릴 환경을 설정합니다.

```
model_ = setup(train, target = 'smoking', session_id = 42,train_size= 0.8,remove_multicollinearity= True,
               data_split_shuffle= True, data_split_stratify= True)
```

- **data** : 모델링에 사용할 데이터를 지정합니다. (여기서는 train을 데이터로 지정)
- **target** : data 내에서 타겟 값으로 사용할 피처를 지정합니다.
- **train_size** : 데이터에서 학습에 사용할 데이터의 비율을 지정합니다.
- **remove_multicollinearity** : 서로 상관관계가 높은 특성을 제거하는 데 사용합니다.
- **data_split_shuffle** : 데이터 세트를 학습 세트와 테스트 세트로 분할하기 전에 섞을지 여부를 제어합니다.
- **data_split_stratify** : 데이터 세트를 훈련 세트와 테스트 세트로 분할할 때 사용합니다.

	Description	Value
0	Session id	42
1	Target	smoking
2	Target type	Binary
3	Original data shape	(192723, 377)
4	Transformed data shape	(192723, 357)
5	Transformed train set shape	(154178, 357)
6	Transformed test set shape	(38545, 357)
7	Numeric features	376
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Remove multicollinearity	True
13	Multicollinearity threshold	0.900000
14	Fold Generator	StratifiedKFold
15	Fold Number	10
16	CPU Jobs	-1
17	Use GPU	False
18	Log Experiment	False
19	Experiment Name	clf-default-name
20	USI	0a9b

ID	Name	Reference	Turbo
lr	Logistic Regression	sklearn.linear_model._logistic.LogisticRegression	True
knn	K Neighbors Classifier	sklearn.neighbors._classification.KNeighborsCl...	True
nb	Naive Bayes	sklearn.naive_bayes.GaussianNB	True
dt	Decision Tree Classifier	sklearn.tree._classes.DecisionTreeClassifier	True
svm	SVM - Linear Kernel	sklearn.linear_model._stochastic_gradient.SGDC...	True
rbfsvm	SVM - Radial Kernel	sklearn.svm._classes.SVC	False
gpc	Gaussian Process Classifier	sklearn.gaussian_process._gpc.GaussianProcessC...	False
mlp	MLP Classifier	sklearn.neural_network._multilayer_perceptron....	False
ridge	Ridge Classifier	sklearn.linear_model._ridge.RidgeClassifier	True
rf	Random Forest Classifier	sklearn.ensemble._forest.RandomForestClassifier	True
qda	Quadratic Discriminant Analysis	sklearn.discriminant_analysis.QuadraticDiscrim...	True
ada	Ada Boost Classifier	sklearn.ensemble._weight_boosting.AdaBoostClas...	True
gbc	Gradient Boosting Classifier	sklearn.ensemble._gb.GradientBoostingClassifier	True
lda	Linear Discriminant Analysis	sklearn.discriminant_analysis.LinearDiscrimina...	True
et	Extra Trees Classifier	sklearn.ensemble._forest.ExtraTreesClassifier	True
xgboost	Extreme Gradient Boosting	xgboost.sklearn.XGBClassifier	True
lightgbm	Light Gradient Boosting Machine	lightgbm.sklearn.LGBMClassifier	True
catboost	CatBoost Classifier	catboost.core.CatBoostClassifier	True
dummy	Dummy Classifier	sklearn.dummy.DummyClassifier	True

compare_models() 함수로 지정한 모델들의 성능을 비교할 수 있습니다.

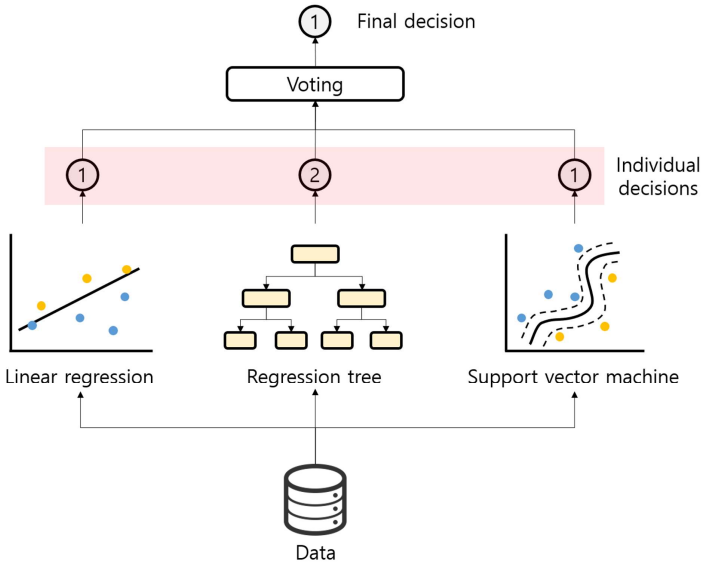
```
compare_models(sort = 'AUC',include = ['et', 'xgboost', 'lightgbm','catboost','gbc','rf', 'ada','dt'])
```

타겟 값이 0 일때와 1일때의 평가 지표, 그리고 두 평가 지표의 평균과 표준 편차를 출력합니다.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
catboost	CatBoost Classifier	0.7860	0.8710	0.7923	0.7281	0.7588	0.5670	0.5687	11.5290
lightgbm	Light Gradient Boosting Machine	0.7814	0.8676	0.7928	0.7208	0.7551	0.5585	0.5606	5.9210
xgboost	Extreme Gradient Boosting	0.7797	0.8652	0.7833	0.7220	0.7514	0.5542	0.5557	5.5920
gbc	Gradient Boosting Classifier	0.7729	0.8599	0.7913	0.7085	0.7476	0.5424	0.5451	20.0700
rf	Random Forest Classifier	0.7703	0.8544	0.7801	0.7087	0.7427	0.5361	0.5381	8.9370
ada	Ada Boost Classifier	0.7665	0.8518	0.7756	0.7047	0.7384	0.5284	0.5304	7.0350
et	Extra Trees Classifier	0.7667	0.8511	0.7786	0.7038	0.7393	0.5291	0.5313	10.2460
dt	Decision Tree Classifier	0.6949	0.6878	0.6402	0.6413	0.6407	0.3756	0.3756	5.4550

(2) Emsemble Learning

a) Emsemble Learning이란?



앙상블 기법이란 여러 개의 개별 모델을 조합하여 최적의 모델로 일반화하는 방법입니다. weak classifier들을 결합하여 strong classifier를 만드는 것입니다. decision tree에서 overfitting되는 문제를 앙상블에서는 감소시킨다는 장점이 있습니다. 앙상블 기법에는 voting, bagging, boosting, stacking이 있는데 저희 조는 각각 다른 알고리즘을 이용한 분류기를 결합하는 방식으로 최종 예측 값을 투표하는 방식인 Voting 기법을 사용하였습니다.

b) Emsemble Learning에 어떤 모델들을 사용하였는가?

1. XGBoost

XGBoost는 고성능 그래디언트 부스팅 알고리즘입니다. 이는 반복적으로 새로운 트리를 구축하여 이전 트리의 예측 오류를 보완하는 방법입니다. 각 트리는 이전 트리의 예측 오류를 줄이기 위해 훈련됩니다

2. CatBoost

CatBoost는 범주형 변수를 자동으로 처리할 수 있는 기능을 갖춘 그래디언트 부스팅 라이브러리입니다. 특히 범주형 데이터가 많은 경우에 효과적이며, 복잡한 데이터 전처리 과정 없이도 높은 성능을 발휘합니다. 과적합에 강하고 일반화 성능이 뛰어나며, 사용하기 쉬운 인터페이스를 제공합니다. 이로 인해 '흡연 상태'와 같은 이산 데이터를 예측하는 데 적합합니다.

3. LightGBM

LightGBM은 대용량 데이터를 빠르게 처리할 수 있는 경량 그래디언트 부스팅 프레임워크입니다. 이는 메모리 효율성과 학습 속도 면에서 우수하며, Leaf-wise 방식의 분할을 통해 효율적인 학습이 가능합니다. 대규모 데이터셋에 적합하며, 다양한 맞춤 설정 옵션을 통해 세밀한 모델 튜닝이 가능합니다. 그러나 과적합에 주의해야 합니다.

4. Logistic Regression

로지스틱 회귀는 분류 문제를 해결하기 위한 기본적이면서도 강력한 알고리즘입니다. 이는 주로 이진 분류 문제에 사용되며, 결과를 확률로 해석할 수 있는 장점을 가지고 있습니다. 간단하고 해석하기 쉬운 모델 구조를 가지고 있지만, 데이터의 복잡한 패턴이나 비선형 관계를 모델링하는 데에는 한계가 있습니다.

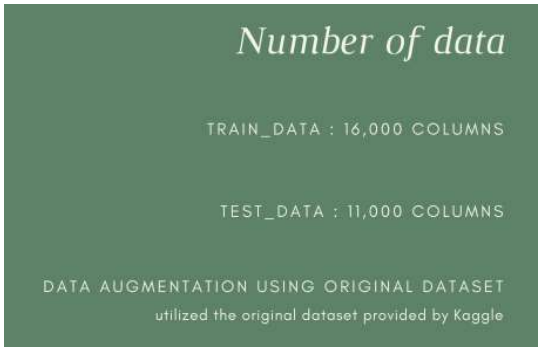
5. Decision Tree

의사결정나무는 데이터를 분류하거나 예측하는 데 사용되는 직관적인 모델입니다. 데이터를 여러 기준에 따라 분할하여 결과를 예측하며, 트리 구조를 통해 결정 과정을 쉽게 시각화할 수 있습니다. 의사결정나무는 해석이 용이하지만, 과적합에 취약하고, 작은 데이터 변화에도 민감하게 반응할 수 있어 안정성이 다소 떨어지는 단점이 있습니다.

3. Experimental Setup & Results

(1) Data EDA

저희 데이터는 23개의 칼럼으로 이루어져있습니다.
나이, 키, 몸무게, 허리둘레와 같은 내용부터 시각, 청력, 혈압, 혈당 등의 내용, 또한 피 검사로 알 수 있는 콜레스테롤과 헤모글로빈 같은 내용들이 담겨 있습니다.



데이터 양이 부족하여 학습의 정확도를 높이기 위해, 데이터 양을 증가시킬 필요가 있었습니다. 이를 위해 캐글에서 제공하는 원본 데이터셋을 찾아서 활용했습니다. 원본 데이터셋에서 중복된 값을 제거한 후, 기존 데이터와 합쳐 총 데이터 양을 늘렸습니다. 이러한 데이터 증강 과정은 모델의 학습 효율과 정확도 향상에 도움이 되었습니다.

Missing Value and Basic Statistics

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159256 entries, 0 to 159255
Data columns (total 24 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   id                   159256 non-null  int64
 1   age                  159256 non-null  int64
 2   height(cm)           159256 non-null  int64
 3   weight(kg)           159256 non-null  int64
 4   waist(cm)            159256 non-null  float64
 5   eyesight(left)       159256 non-null  float64
 6   eyesight(right)      159256 non-null  float64
 7   hearing(left)        159256 non-null  int64
 8   hearing(right)       159256 non-null  int64
 9   systolic             159256 non-null  int64
 10  relaxation           159256 non-null  int64
 11  fasting blood sugar  159256 non-null  int64
 12  Cholesterol          159256 non-null  int64
 13  triglyceride         159256 non-null  int64
 14  HDL                  159256 non-null  int64
 15  LDL                  159256 non-null  int64
 16  hemoglobin           159256 non-null  float64
 17  Urine protein        159256 non-null  int64
 18  serum creatinine     159256 non-null  float64
 19  AST                  159256 non-null  int64
 20  ALT                  159256 non-null  int64
 21  Gtp                  159256 non-null  int64
 22  dental caries        159256 non-null  int64
 23  smoking              159256 non-null  int64
dtypes: float64(5), int64(19)
memory usage: 29.2 MB
```

	count	mean	std	min	25%	50%	75%	max
id	159256.0	79627.500000	45973.391572	0.0	39813.75	79627.5	119441.25	159255.0
age	159256.0	44.306626	11.842286	20.0	40.00	40.0	55.00	85.0
height(cm)	159256.0	165.266929	8.818970	135.0	160.00	165.0	170.00	190.0
weight(kg)	159256.0	67.143662	12.586198	30.0	60.00	65.0	75.00	130.0
waist(cm)	159256.0	83.001990	8.957937	51.0	77.00	83.0	89.00	127.0
eyesight(left)	159256.0	1.005798	0.402113	0.1	0.80	1.0	1.20	9.9
eyesight(right)	159256.0	1.000989	0.392299	0.1	0.80	1.0	1.20	9.9
hearing(left)	159256.0	1.023974	0.152969	1.0	1.00	1.0	1.00	2.0
hearing(right)	159256.0	1.023421	0.151238	1.0	1.00	1.0	1.00	2.0
systolic	159256.0	122.503648	12.729315	77.0	114.00	121.0	130.00	213.0
relaxation	159256.0	76.874071	8.994642	44.0	70.00	78.0	82.00	133.0
fasting blood sugar	159256.0	98.352552	15.329740	46.0	90.00	96.0	103.00	375.0
Cholesterol	159256.0	195.796165	28.396959	77.0	175.00	196.0	217.00	393.0
triglyceride	159256.0	127.616046	66.188989	8.0	77.00	115.0	165.00	766.0
HDL	159256.0	55.852684	13.964141	9.0	45.00	54.0	64.00	136.0
LDL	159256.0	114.607682	28.158931	1.0	95.00	114.0	133.00	1860.0
hemoglobin	159256.0	14.796965	1.431213	4.9	13.80	15.0	15.80	21.0
Urine protein	159256.0	1.074233	0.347856	1.0	1.00	1.0	1.00	6.0
serum creatinine	159256.0	0.892764	0.179346	0.1	0.80	0.9	1.00	9.9
AST	159256.0	25.516853	9.464882	6.0	20.00	24.0	29.00	778.0
ALT	159256.0	26.550296	17.753070	1.0	16.00	22.0	32.00	2914.0
Gtp	159256.0	36.216004	31.204643	2.0	18.00	27.0	44.00	999.0
dental caries	159256.0	0.197996	0.398490	0.0	0.00	0.0	0.00	1.0
smoking	159256.0	0.437365	0.496063	0.0	0.00	0.0	1.00	1.0

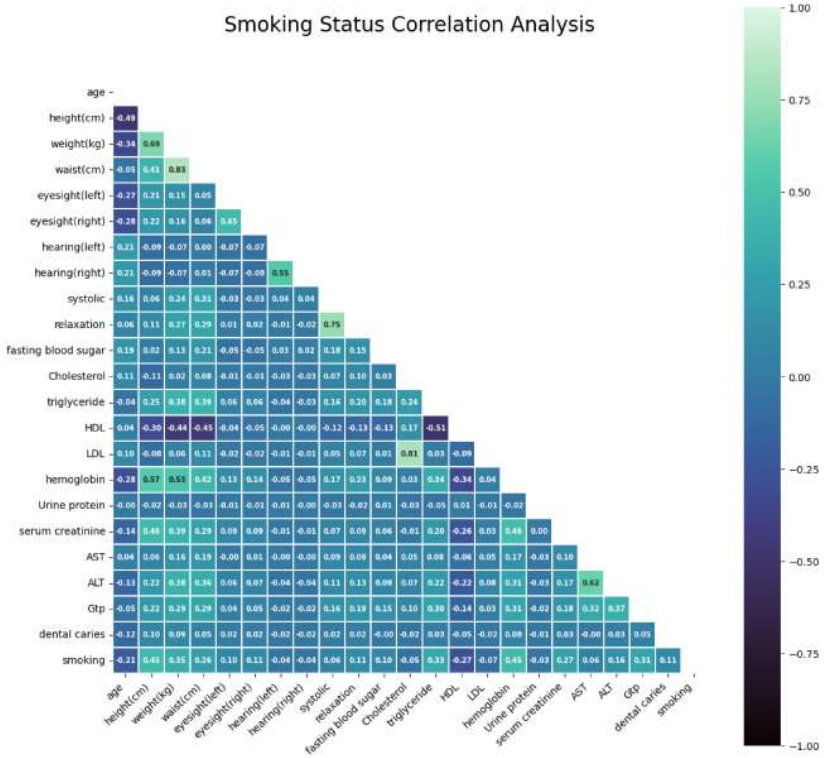
증강된 데이터는 EDA를 진행한 결과, 결측치는 없는 것으로 확인되었습니다. 기초통계량 표를 보면 알 수 있듯이, 시력이 MAX 값이 9.9가 나온것으로 볼 때, 이상치가 있는 것으로 파악할 수 있었습니다. 또한 수축기 혈압의 max 값이 233인 것도 이상치가 있는 것으로 의심할 수 있습니다. 이 외에도 AST, ALT, 감마 GTP같은 수치도 평균에 비해 너무 높은 max 값을 가지고 있어서 이상치를 가지고 있다고 생각할 수 있습니다.

SMOKING =1	TRIGLYCERIDE	LDL	AST	ALT	GTP
MAX	766	1220	656	2914	999
MEAN	152.537952	26.154451	26.154451	29.832714	47.030523

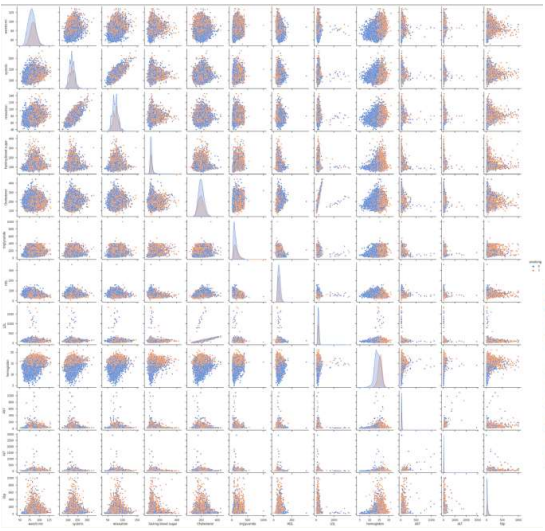
SMOKING =0	TRIGLYCERIDE	LDL	AST	ALT	GTP
MAX	466	1860	778	745	926
MEAN	108.242972	116.402308	25.021216	23.998705	27.809326

comparison between smokers and non-smokers

또한 흡연자 그룹과 비흡연자 그룹을 비교한 결과, Triglyceride(중성지방)', 'ALT(간수치 ALT)', 'Gtp(감마 Gtp)'는 max와 mean 모두 흡연자가 비흡연자보다 높았고 'LDL(저밀도 지단백)', 'AST(간수치 AST)'는 max와 mean 모두 비흡연자가 흡연자보다 높았습니다.



Correlation Matrix를 보면, 헤모글로빈만 유의미하게 수치가 높은 것을 확인할 수 있습니다. 헤모글로빈은 남성이 여성보다 일반적으로 더 높은 수치를 지니는데, 우리 Dataset은 성별과 관련된 컬럼이 존재하지 않습니다. 이러한 상황에서 흡연자의 혈색소 수치가 높게 나타나는 것은 남성 흡연자가 많다는 가정 하에 이해할 수 있으며, 따라서 혈색소와 흡연 사이에 높은 상관관계가 있음을 해석할 수 있습니다.



seaborn 라이브러리를 사용하여 원본 데이터프레임에서 선택한 수치형 열에 대한 페어 플롯(pair plot)을 생성하여보니 확실히 헤모글로빈은 흡연여부를 판별하기에 좋은 변수인 것으로 생각됩니다.

(2) Data Engineering

모델링 전, 다음과 같은 데이터 엔지니어링이 수행되었습니다. 기존의 캐글 데이터를 사용하여 데이터 양을 증가시켰으며, 누락값과 중복값을 확인했습니다. 또한, 이상치 제거, 스케일링, 원-핫 인코딩, 로그 변환 등을 수행했습니다.

(a) Remove Outlier (이상치 제거)

Limit range of data:

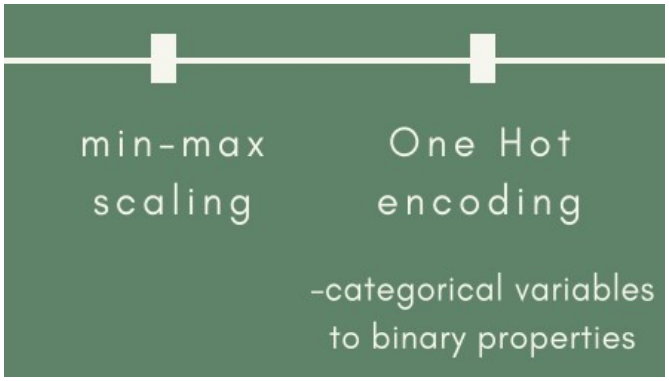
```
df['Gtp'] = np.clip(df['Gtp'], 0, 300)
df['HDL'] = np.clip(df['HDL'], 0, 110)
df['LDL'] = np.clip(df['LDL'], 0, 200)
df['ALT'] = np.clip(df['ALT'], 0, 150)
df['AST'] = np.clip(df['AST'], 0, 100)
df['serum creatinine'] = np.clip(df['serum creatinine'], 0, 3)
```

Remove outlier data:

```
df['eyesight(left)']
  = np.where(df['eyesight(left)'] > 9, 0, df['eyesight(left)'])
df['eyesight(right)']
  = np.where(df['eyesight(right)'] > 9, 0, df['eyesight(right)'])
```

앞서 확인한 바와 같이, 특정 변수들은 비정상적인 값이 등장하지 않도록 범위를 제한했습니다. 또한, 다른 값들에 비해 너무 큰 값을 가진 경우, 그 값을 전체적으로 제거하는 작업을 수행했습니다. 청력 데이터의 경우, 좌우 청력 중 더 좋은 쪽을 좌측 청력으로, 더 나쁜 쪽을 우측 청력으로 정리합니다. 이때 현재 1 또는 2로 표현된 값을 0 또는 1로 정규화합니다. 시력 데이터는 9 이상의 이상치 값을 0으로 처리하고, 좌우 시력 중 더 좋은 쪽을 좌측 시력으로, 더 나쁜 쪽을 우측 시력으로 정렬합니다. 그리고 Gtp는 0과 300 사이, HDL은 0과 110 사이, LDL은 0과 200 사이, ALT는 0과 150 사이, AST는 0과 100 사이, 혈청 크레아티닌은 0과 3 사이의 값으로 제한하여 데이터를 보다 효과적으로 정규화하였습니다.

(b) Scaling (정규화)



이후 데이터셋에 스케일링을 적용하여 이상치들을 효과적으로 처리하고자 합니다. 이를 위해 우선, 특정 열들에 대해 **최소-최대 정규화(Min-Max Scaling)**를 적용하여, train과 test 데이터의 값을 0과 1 사이로 조정하려고 합니다. 이 과정에서는 train과 test 데이터셋 양쪽에서 나타나는 최소값(min_val)과 최대값(max_val)을 기준으로 정규화 범위를 설정합니다. 이는 이상치를 보다 효과적으로 관리할 수 있도록 돕습니다.

One Hot Encoding(OHE) 기법을 사용하여 범주형 변수를 이진 특성으로 변환합니다. 이 과정은 먼저 train과 test 데이터를 결합한 후, 지정된 열들에 대해 OHE를 실시합니다. 이때,

가장 드문 범주는 중복으로 간주되어 제외됩니다. 변환된 데이터는 이후 다시 train과 test 데이터셋으로 분리됩니다.

Root Mean Squared Error(RMSE) 계산을 통해 예측값과 실제값 간의 차이를 평가합니다. 이 방법은 모델의 예측 정확도를 객관적으로 측정하는 데 사용됩니다. 또한, 결측치가 있는 행들은 특정 피처를 기준으로 기록됩니다.

CatBoost 알고리즘을 사용해 결측치를 반복적으로 채우는 작업을 진행합니다. 초기 단계에서는 결측치가 있는 피처를 찾아 그 평균값으로 채웁니다. 그 후 각 반복 과정에서 해당 피처를 대상으로 한 랜덤 포레스트 모델을 학습시키고, 이를 통해 얻은 예측값으로 결측치를 업데이트합니다. 이 과정은 RMSE를 최소화하면서 모델의 정확도를 높이는 데 기여합니다.

이 4가지의 Scaling 과정을 통해 이상치를 효과적으로 처리하여 모델의 성능을 높이고자 하였습니다.

(c) Numerical Transformation

Feature	Initial ROC_AUC	Transformation	Tranformed ROC_AUC
age	0.6153010625151015	age	0.6153010625151015
height(cm)	0.7531335297614603	height(cm)	0.7531335297614603
weight(kg)	0.7079867509018658	weight(kg)	0.7079867509018658
waist(cm)	0.6508305863480158	waist(cm)	0.6508305863480158
eyesight(left)	0.579464517903229	eyesight(left)	0.579464517903229
eyesight(right)	0.5810586555846605	eyesight(right)	0.5810586555846605
systolic	0.5388086994036667	systolic	0.5388086994036667
relaxation	0.5664235515547296	relaxation	0.5664235515547296
fasting blood sugar	0.5651136235653957	fasting blood sugar	0.5651136235653957
Cholesterol	0.5269492239020683	Cholesterol	0.5269492239020683
triglyceride	0.6949508131085066	triglyceride	0.6949508131085066
HDL	0.6524546024017777	HDL	0.6524546024017777
LDL	0.540517619041145	LDL	0.540517619041145
hemoglobin	0.7634366421450545	hemoglobin	0.7634366421450545
Urine protein	0.5056717990735431	Urine protein	0.5056717990735431
serum creatinine	0.6649061870598099	serum creatinine	0.6649061870598099
AST	0.5463943062940902	AST	0.5463943062940902
ALT	0.6312732257229434	ALT	0.6312732257229434
Gtp	0.7416016920440514	Gtp	0.7416016920440514

overall best CV ROC AUC score: 0.7634366421450545

앞서 **최소-최대 정규화(Min-Max Scaling)**를 사용하여 연속형 변수들을 0과 1사이의 값으로 스케일링한 뒤, 각 변수에 대해 로그 변환, 제곱근 변환, Box-Cox 변환, Yeo-Johnson 변환, 두 가지 지수(power) 변환, 그리고 로그와 제곱근의 조합 변환 등 다양한 변환을 적용합니다. 이러한 변환된 값들과 원래의 값들을 포함하여 주성분 분석(PCA)을 통해 차원을 축소합니다. K-폴드 교차 검증을 사용하여 각 변환된 변수의 성능을 로지스틱 회귀 모델로 평가하고, 가장 성능이 좋은 변수를 선택합니다. 이 과정을 통해, 최적의 수치 변환을 적용한 변수들을 선별하고, 모델의 성능을 향상시키려고 하였습니다.

(d) Discrete Feature to Categorical

Feature	Encoded Feature	ROC AUC Score
cat_age	cat_age_count	0.6193448851503595
cat_height(cm)	cat_height(cm)_count	0.7541779636690394
cat_weight(kg)	cat_weight(kg)_count	0.7079529922079254
cat_waist(cm)	cat_waist(cm)_count	0.6412649991792173
cat_eyesight(left)	cat_eyesight(left)_count	0.5793485408100109
cat_eyesight(right)	cat_eyesight(right)_count	0.5812127021315897
cat_systolic	cat_systolic_count	0.5740708436445809
cat_relaxation	cat_relaxation_count	0.5882576921865621
cat_fasting blood sugar	cat_fasting blood sugar_count_label	0.5696070443363543
cat_Cholesterol	cat_Cholesterol_count	0.5526887901239161
cat_triglyceride	cat_triglyceride_count_label	0.6900615890277451
cat_HDL	cat_HDL_count	0.6528891288948875
cat_LDL	cat_LDL_count_label	0.5610395537461915
cat_hemoglobin	cat_hemoglobin_count_label	0.7601189445748036
cat_Urine protein	cat_Urine protein_count	0.5054805678521632
cat_serum creatinine	cat_serum creatinine_count	0.672028387983824
cat_AST	cat_AST_count	0.5507340033457089
cat_ALT	cat_ALT_count_label	0.6339031115330023
cat_Gtp	cat_Gtp_count	0.740756318525521

데이터 분석 과정에서, 특히 이산형 데이터를 다룰 때, 데이터의 유용성과 처리 효율성을 높이기 위해 범주형으로의 변환이 필요하게 됩니다. 이를 위해 각 이산형 변수를 범주형으로 변환하고자 하였는데, 이 과정에서 고빈도 값에 대해서는 **One-Hot Encoding**을 적용하고, 낮은 빈도의 범주들은 특정 값으로 대체하여 차원을 축소하는 방식을 택했습니다. 추가적으로, 각 범주별 타겟 변수의 평균값을 기준으로 순위를 매겨 새로운 특성을 만들고, 이 특성의 빈도수에 따라 순위를 다시 매겨 추가 특성을 생성합니다. 이런 변환 과정을 통해 생성된 특성들은 **HistGradientBoostingClassifier**를 사용하여 평가되며, 여러 특성 중에서 가장 좋은 성능을 보이는 특성을 선택합니다. 이 전체 과정은 모델의 예측 정확도를 향상시키기 위해, 이산형 데이터를 더 효과적으로 범주형 데이터로 변환하고 활용하기 위한 목적으로 진행됩니다.

(e) Numerical Clustering

Clustered Feature	ROC AUC (CV-TRAIN)
age_unimp_cluster_WOE	0.6189801815069776
height(cm)_unimp_cluster_WOE	0.7542023608361975
weight(kg)_unimp_cluster_WOE	0.7079974583437237
waist(cm)_unimp_cluster_WOE	0.6490101880556234
eyesight(left)_unimp_cluster_WOE	0.579402520835678
eyesight(right)_unimp_cluster_WOE	0.5810626314711035
systolic_unimp_cluster_WOE	0.5528058707913498
relaxation_unimp_cluster_WOE	0.5692608912500543
fasting blood sugar_unimp_cluster_WOE	0.5635042254246605
Cholesterol_unimp_cluster_WOE	0.5393871674104371
triglyceride_unimp_cluster_WOE	0.6938429517631054
HDL_unimp_cluster_WOE	0.6500857027222247
LDL_unimp_cluster_WOE	0.5411670266954329
hemoglobin_unimp_cluster_WOE	0.7578915393839374
Urine protein_unimp_cluster_WOE	0.5054805678521632
serum creatinine_unimp_cluster_WOE	0.671855037369254
AST_unimp_cluster_WOE	0.5453442108624339
ALT_unimp_cluster_WOE	0.628304925552096
Gtp_unimp_cluster_WOE	0.736330690513397

overall best CV score: 0.7634366421450545

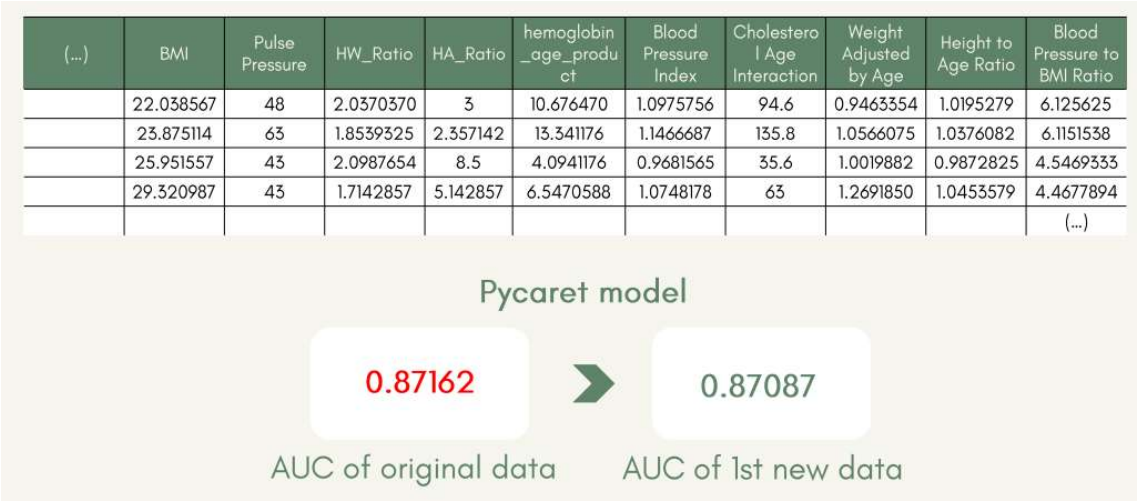
또한 모델의 성능을 향상시키기 위한 전략으로, 연속형 변수들을 클러스터링하여 새로운 특성을 생성하는 방법을 설계했습니다. 이 과정의 첫 단계로, 연속형 변수들 중에서 상대적으로 중요도가 낮은(unimportant) 특성들을 식별하여 서브셋을 형성합니다. 이 서브셋은 먼저 표준화(StandardScaler) 과정을 거친 후, **Kmeans 클러스터링**을 적용하여 각 데이터 포인

트를 10개의 클러스터 중 하나에 할당합니다. 이렇게 할당된 클러스터 레이블들은 Weight of Evidence(WOE) 형태의 새로운 특성으로 변환되어 기존 데이터셋에 추가됩니다. 그 다음으로, 이 새로운 WOE 특성을 활용해 **HistGradientBoostingClassifier 모델**을 훈련시키고, 5-폴드 교차 검증을 통해 각 특성의 ROC AUC 성능을 평가합니다. 이러한 접근은 **상대적으로 덜 중요한 것으로 판단된 특성들에서 숨겨진 유용한 패턴을 발견**하고, 이를 활용하여 모델의 전반적인 예측 정확도를 개선하는 것을 목표로 합니다.

(f) 1st derivative variables

EDA 과정에서 'Hemoglobin' 변수가 '흡연'과 높은 상관관계가 있음을 발견했습니다. 더불어, '흡연' 변수를 제외한 다른 변수들 간에도 서로 강한 상관관계가 있는 것을 확인할 수 있었습니다. 이를 바탕으로, 상관관계가 높은 변수들을 활용하여 새로운 파생 변수들을 생성하기로 하였습니다. 첫 번째 시도에서는 특히 '흡연' 여부를 판별하는 데 영향을 미칠 것으로 예상되는 파생변수들에 집중하였습니다. 생성된 파생변수는 다음과 같습니다.

: BMI, Pulse Pressure, HW_Ratio, HA_Ratio, Hemoglobin_age_product, Blood Pressure Index, Cholestrol Age Interaction, Weight Adjusted by Age, Height to Age Ratio, Blood Pressure to BMI Ratio

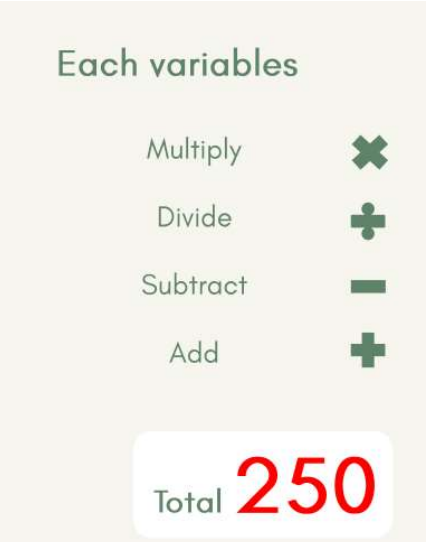


이렇게 생성한 파생 변수들을 원래 데이터 세트에 추가되어 새로운 데이터 세트를 생성했습니다. Pycaret을 통해 돌린 CatBoost 모델에서 이러한 파생 특성을 사용하여 모델을 실행했을 때, 흡연 상태 예측을 위한 **AUC는 0.87087**이었습니다. 그러나 이 수치는 원래 변수만

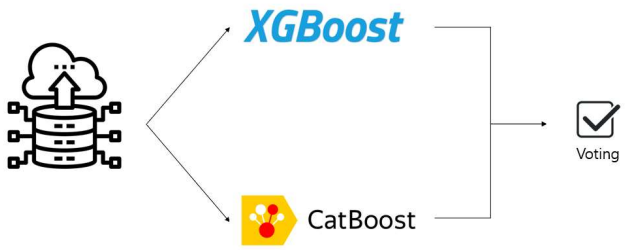
을 사용한 모델의 성능인 0.87162보다 낮았습니다.

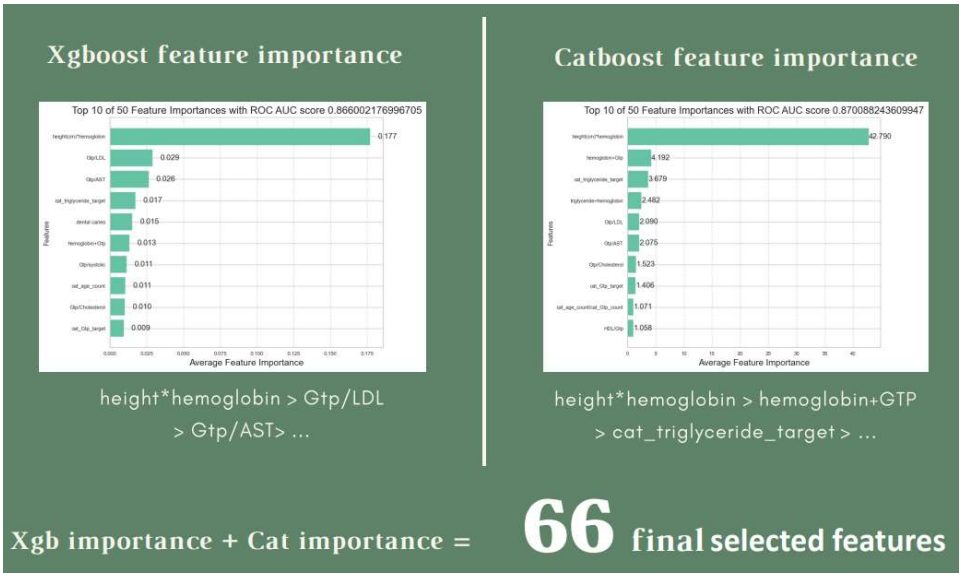
(g) 2nd derivative variables

```
for j in range(i+1, len(cols)):
            col2 = cols[j]
            # Multiply
            temp_df[col1 + '*' + col2] = train[col1] * train[col2]
            temp_df_test[col1 + '*' + col2] = test[col1] * test[col2]

            # Divide (col1 / col2)
            temp_df[col1 + '/' + col2] = train[col1] / (train[col2] + 1e-5)
            temp_df_test[col1 + '/' + col2] = test[col1] / (test[col2] + 1e-5)

            # Divide (col2 / col1)
            temp_df[col2 + '/' + col1] = train[col2] / (train[col1] + 1e-5)
            temp_df_test[col2 + '/' + col1] = test[col2] / (test[col1] + 1e-5)

            # Subtract
            temp_df[col1 + '-' + col2] = train[col1] - train[col2]
            temp_df_test[col1 + '-' + col2] = test[col1] - test[col2]

            # Add
            temp_df[col1 + '+' + col2] = train[col1] + train[col2]
            temp_df_test[col1 + '+' + col2] = test[col1] + test[col2]
```



따라서, 우리는 또 다른 파생 변수를 생성했습니다. 모든 변수들을 곱하고, 나누고, 빼고, 더하는 방식으로 총 250개의 파생 변수를 다시 생성했습니다. 이러한 재생성된 파생 변수들은 앙상블 모델에서 사용되었습니다.





파생변수의 수가 상당히 많아진 상황에서, 이를 효율적으로 활용하기 위해 XGBoost와 CatBoost 모델을 앙상블 방식으로 사용하게 되었습니다. 이 두 모델의 특성 중요도를 계산함으로써, 흡연 상태 예측에 더 유리한 변수들을 식별할 수 있었습니다. 특성 중요도 분석 결과, '키*헤모글로빈' 변수가 두 모델에서 모두 가장 중요한 요소로 나타났습니다. 더불어, EDA 과정에서 '헤모글로빈'과 'GTP'와 같이 흡연과 높은 상관관계를 보였던 여러 파생 변수들이 상위 10위 안에 다수 포함되었습니다. 최종적으로, 이러한 중요도를 바탕으로 총 66개의 변수가 선택되었고, 이들을 기반으로 앙상블 모델을 구축하였습니다.

(3) Data Modeling (Emsemble)

다음과 같이 각 모델 별 파라미터를 설정하고 앙상블 모델을 돌렸습니다.

Model	Parameter
XGBoost	`xgb_params = { 'n_estimators': self.n_estimators, 'learning_rate': 0.1, 'max_depth': 4, 'subsample': 0.8, 'colsample_bytree': 0.1, 'n_jobs': -1, 'eval_metric': 'logloss', 'objective': 'binary:logistic', 'tree_method': 'hist', 'verbosity': 0, 'random_state': self.random_state, } if self.device == 'gpu': xgb_params['tree_method'] = 'gpu_hist' xgb_params['predictor'] = 'gpu_predictor'`
	`xgb_params2=xgb_params.copy() xgb_params2['subsample']= 0.3 xgb_params2['max_depth']=8 xgb_params2['learning_rate']=0.005 xgb_params2['colsample_bytree']=0.9`
	`xgb_params3=xgb_params.copy() xgb_params3['subsample']= 0.6 xgb_params3['max_depth']=6 xgb_params3['learning_rate']=0.02 xgb_params3['colsample_bytree']=0.7`
LightGBM	`lgb_params = { 'n_estimators': self.n_estimators, 'max_depth': 8, 'learning_rate': 0.02, 'subsample': 0.20, 'colsample_bytree': 0.56, 'reg_alpha': 0.25, 'reg_lambda': 5e-08, 'objective': 'binary', 'boosting_type': 'gbdt', 'device': self.device, 'random_state': self.random_state, }`
	`lgb_params2 = { 'n_estimators': self.n_estimators, 'max_depth': 6, 'learning_rate': 0.05, 'subsample': 0.20, 'colsample_bytree': 0.56, 'reg_alpha': 0.25, 'reg_lambda': 5e-08, 'objective': 'binary', 'boosting_type': 'gbdt', 'device': self.device, 'random_state': self.random_state, }`
	`lgb_params3=lgb_params.copy() lgb_params3['subsample']=0.9 lgb_params3['reg_lambda']=0.3461495211744402 lgb_params3['reg_alpha']=0.3095626288582237 lgb_params3['max_depth']=8 lgb_params3['learning_rate']=0.007 lgb_params3['colsample_bytree']=0.5`
	`lgb_params4=lgb_params2.copy() lgb_params4['subsample']=0.7 lgb_params4['reg_lambda']=0.1 lgb_params4['reg_alpha']=0.2 lgb_params4['max_depth']=10 lgb_params4['learning_rate']=0.007 lgb_params4['colsample_bytree']=0.5`

<table>
<tr><td rowspan="3">Catboost</td><td>

```
cb_params = {
    'iterations': self.n_estimators,
    'depth': 6,
    'learning_rate': 0.1,
    'l2_leaf_reg': 0.7,
    'random_strength': 0.2,
    'max_bin': 200,
    'od_wait': 65,
    'one_hot_max_size': 120,
    'grow_policy': 'Depthwise',
    'bootstrap_type': 'Bayesian',
    'od_type': 'Iter',
    'eval_metric': 'AUC',
    'loss_function': 'Logloss',
    'task_type': self.device.upper(),
    'random_state': self.random_state,
}
cb_sym_params = cb_params.copy()
cb_sym_params['grow_policy'] = 'SymmetricTree'
cb_loss_params = cb_params.copy()
cb_loss_params['grow_policy'] = 'Lossguide'
```

</td></tr>
<tr><td>

```
cb_params2=  cb_params.copy()
cb_params2['learning_rate']=0.01
cb_params2['depth']=8
```

</td></tr>
<tr><td>

```
cb_params3={
    'iterations': self.n_estimators,
    'random_strength': 0.1,
    'one_hot_max_size': 70,
    'max_bin': 100,
    'learning_rate': 0.008,
    'l2_leaf_reg': 0.3,
    'grow_policy': 'Depthwise',
    'depth': 10,
    'max_bin': 200,
    'od_wait': 65,
    'bootstrap_type': 'Bayesian',
    'od_type': 'Iter',
    'eval_metric': 'AUC',
    'loss_function': 'Logloss',
    'task_type': self.device.upper(),
    'random_state': self.random_state,
}
cb_params4=  cb_params.copy()
cb_params4['learning_rate']=0.01
cb_params4['depth']=12
dt_params= {'min_samples_split': 30, 'min_samples_leaf': 10, 'max_depth': 8, 'criterion': 'gini'}
```

</td></tr>
<tr><td>Logistic Regression</td><td>

```
    'lr': LogisticRegression(),
```

</td></tr>
<tr><td>Decision Tree</td><td>

```
'dt' :DecisionTreeClassifier(**dt_params,random_state=self.random_state),
```

</td></tr>
<tr><td>ANN</td><td>

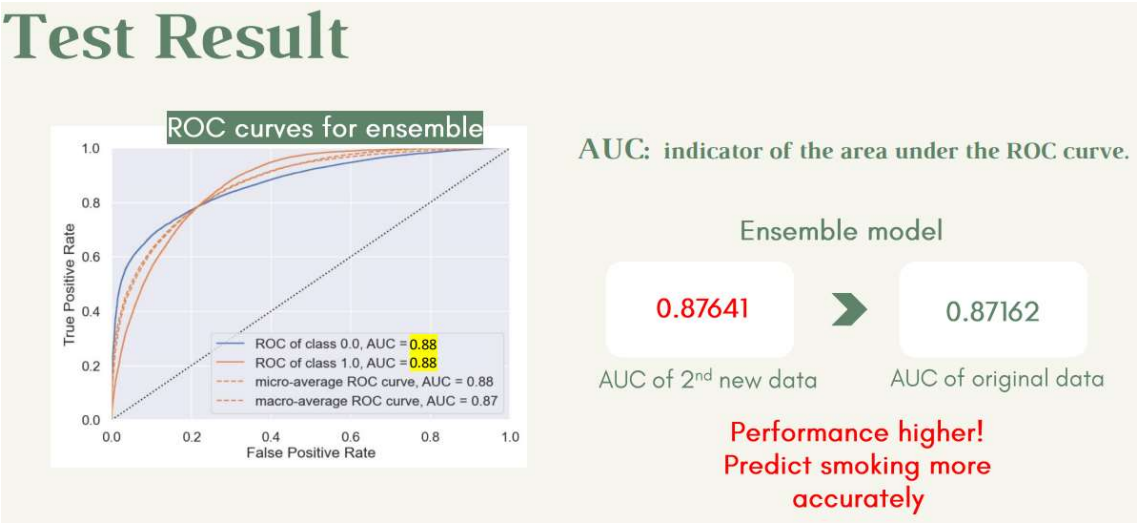
```
ann = Sequential()
ann.add(Dense(64, input_dim=X_train.shape[1], kernel_initializer='he_uniform', activation=lrelu))
ann.add(Dropout(0.1))
ann.add(Dense(16,  kernel_initializer='he_uniform', activation=lrelu))
ann.add(Dropout(0.1))
ann.add(Dense(4,  kernel_initializer='he_uniform', activation='relu'))
ann.add(Dropout(0.1))

ann.add(Dense(1,  kernel_initializer='he_uniform', activation='sigmoid'))
ann.compile(loss="binary_crossentropy", optimizer=sgd,metrics=['accuracy'])
```

</td></tr>
</table>

4. Conclusion

(1) Result

모델의 성능은 **AUC** 값으로 평가되었습니다. AUC는 ROC 곡선 아래 영역의 지표로 모델이 클래스를 얼마나 잘 구별할 수 있는지를 알려줍니다. 이 영역이 클수록 모델의 정확도가 더 높다는 것을 의미하며 예측이 100% 정확할 경우, AUC는 1.0입니다.



초기에는 흡연 여부를 판별하는 데 필요하다고 생각했던 파생변수들만을 사용하여 PyCaret을 통해 CatBoost 모델을 구축했습니다. 그러나 이러한 접근은 기대와 달리 흡연 여부 판별에 큰 도움이 되지 않았고, 실제로 성능이 기존의 0.87162에서 0.87087로 약간 저하되었습니다. 이에 따라, 저희 팀은 변수들 간의 추가적인 관계를 탐색하기 위해 각 변수 간의 덧셈(+), 뺄셈(-), 곱셈(x), 나눗셈(/) 등을 적용하여 새로운 관계를 구축하였습니다. 이후 XGBoost와 CatBoost 모델의 특성 중요도를 분석하여 총 66개의 가장 유의미한 변수를 선정하고, 이를 기반으로 앙상블 모델을 재구성했습니다. 이러한 조정으로, 모델의 AUC는 0.87641로 향상되었으며, 이는 1에 가까운 값으로 예측의 정확성이 높음을 나타냅니다.

또한, 캐글 대회 제출 결과에서는 280위대에서 64위로 크게 상승했습니다. 결론적으로, 원래 변수에서 많은 파생 변수를 만들고 이러한 변수들의 중요성을 분석함으로써 흡연 예측을 더 정확하게 만드는 데 성공했습니다.

(2) Expectation (기대 효과)

생체 정보는 조작할 수 없기 때문에 정확한 흡연 여부 판별이 가능하며 이는 과소 보고 현상을 없앨 수 있습니다. 또한 흡연과 관련된 건강 변화를 명확하게 비교하여 흡연의 부정적 영향을 보여주는 도구로 활용될 수 있으며, 흡연 관련 건강 문제에 대한 인식을 제고할 뿐만 아니라 공중 보건 기관의 흡연 예방 및 중단 노력에도 도움이 될 수 있습니다.

(3) Limitation (한계점)

하지만 이러한 분석은 한계점이 존재합니다. 첫번째로, 머신러닝 자체의 문제입니다. 훈련 데이터와 다른 형식의 입력값은 흡연 여부를 정확히 추정할 수 없습니다. 이를 위해 데이터를 모델에 맞게 전처리하는 과정은 시간과 비용 측면에서 비효율적일 수 있습니다.
두번째로, 모델이 악용될 우려가 있습니다. 보험사가 의료 정보를 활용해 개인의 보험 가입을 제한하는 등 건강정보 데이터로 모델을 악용할 가능성이 존재합니다. 따라서 이를 예방할 수 있는 방안이 추가로 마련되어야 할 것입니다.