그만피시조

생체 지표를 통한 흡연 분류

Figure out with body signal of smoking

1

Project Overview

2

About Data

3

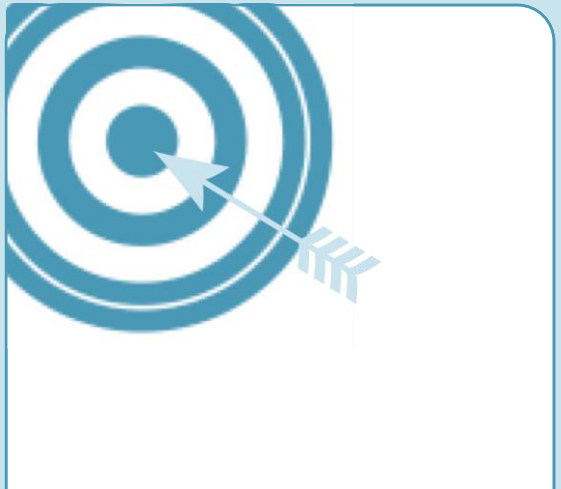
Modeling

4

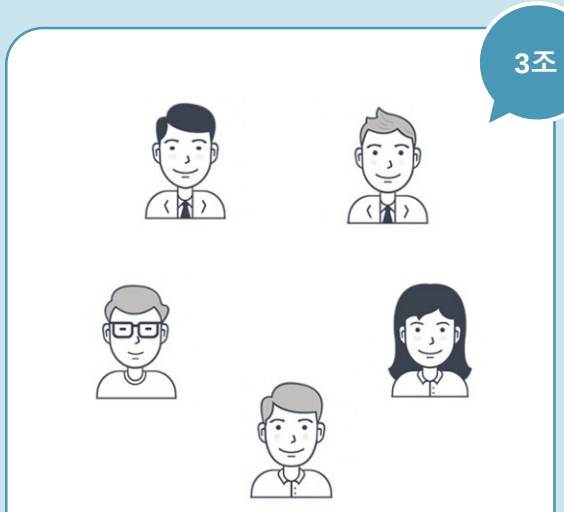
최종 결론

1.Project overview

프로젝트 배경



프로젝트 목표



조원 소개

Figure out with body signal of smoking

1. Project overview

- 캐글에서 제공하는 데이터셋
- 기본적인 신체 정보를 이용한 흡연 유무 분석
- 55692 rows, 27 features
- https://www.kaggle.com/datasets/kukuroo3/body-signal-of-smoking/code?datasetId=2157551&sortBy=voteCount

gender
hearing(left)
hearing(right)
age
height(cm)
weight(kg)
waist(cm)
eyesight(left)
eyesight(right)

systolic
relaxation
fasting blood sugar
Cholesterol
triglyceride
HDL
LDL
hemoglobin

Urine protein
serum creatinine

AST
ALT
Gtp

oral
dental caries
tartar

Figure out with body signal of smoking

1. Project overview

프로젝트 목표

실제로 흡연 여부를 예측하는 데에 어떤 생체 지표가 영향을 미치는지 분석하고, 흡연 여부를 예측하여 실제 흡연자를 판단하기 위함.

Decision Tree
Random forest
XGBoost

Logistic Regression
Support Vector Machine

Figure out with body signal of smoking

1. Project overview

조원 소개

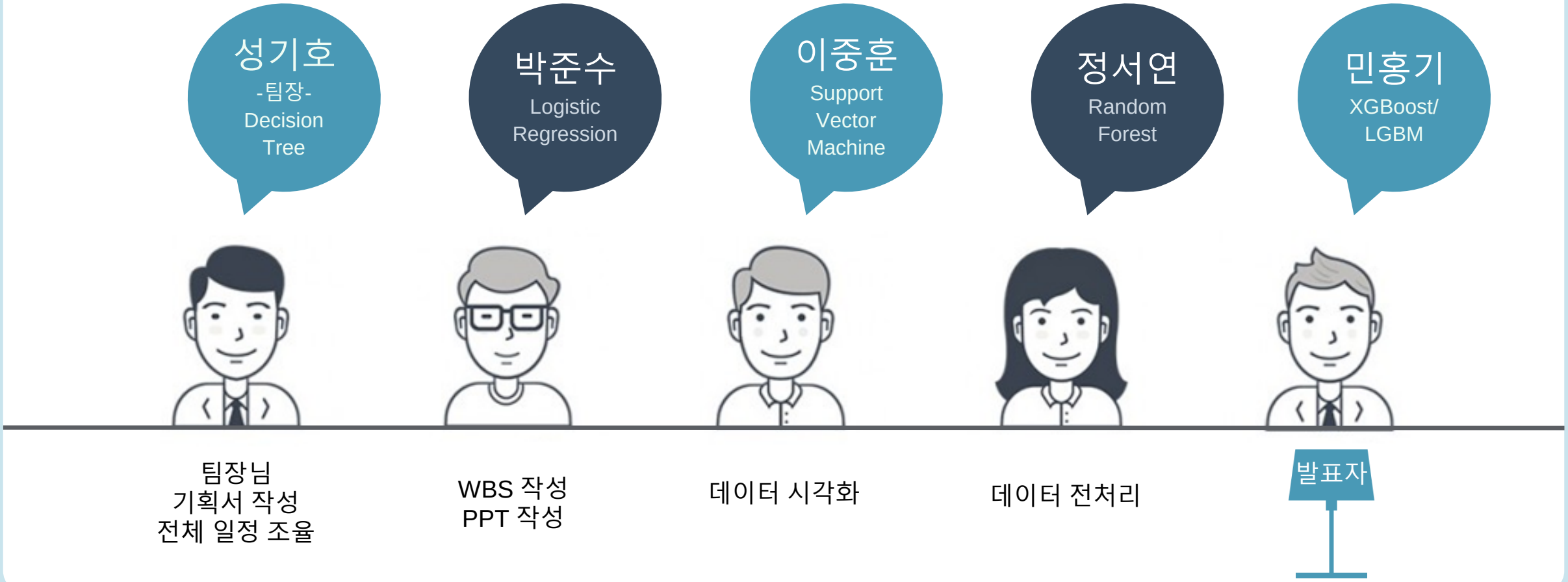
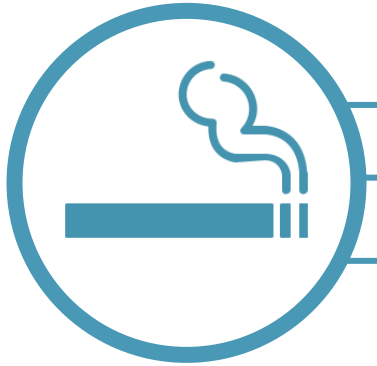


Figure out with body signal of smoking

2. About Data

2. About Data - Data Domain



특성	type	비고
smoking	int 64	흡연여부 1/0

기본 특성

심혈관 및 호흡

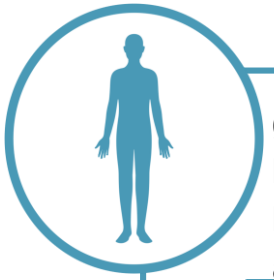
신장

간

구강상태

Figure out with body signal of smoking

2. About Data - Domain Data



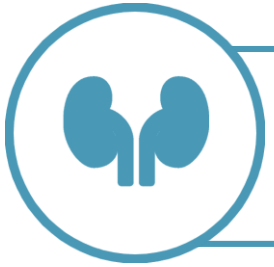
gender	성별
hearing(left)	청력(왼쪽)
hearing(right)	청력(오른쪽)
age	연령
height(cm)	**키**
weight(kg)	**몸무게**
waist(cm)	**허리둘레**
eyesight(left)	시력(왼쪽)
eyesight(right)	시력(오른쪽)

type	단위	
int64	키(cm)	WWI(비만지수) wwi = waist / sqrt(weight) BMI bmi = kg/m**2 과거 흡연자는 다른 두 군에 비해 체질량지수와 허리둘레가 의미 있게 컸다. 비만과 관련된 다양한 요소들을 보정했을 때, 47세 이상에서 현재 흡연자는 비만에 대한 위험도가 비흡연자에 비해 의미 있게 감소하지만(P < 0.001), 과거흡연자에서는 비흡연자와 차이가 없었다. 이기헌, et al. 한국 남성 비만과 흡연의 관련성: 제 3 차 및 4 차 국민건강영양조사 자료 분석. 대한금연학회지, 2010, 1.2: 115-123.
int64	무게(kg)	
float64	허리둘레(cm)	

systolic	수축기(혈압)
relaxation	이완기(혈압)
fasting blood sugar	공복혈당
Cholesterol	**콜레스테롤**
triglyceride	**중성지방**
hemoglobin	**헤모글로빈**
HDL	고단백 콜레스테롤
LDL	저단백 콜레스테롤

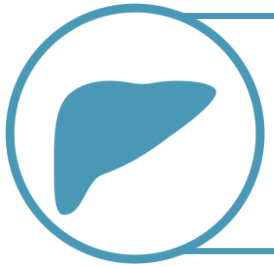
type	비고
float64	적정 수치는 200 미만
float64	중성지방 적정 수치는 150 미만
float64	남성 : 13.5~17.5g/dL 여성 : 12.5-15.5g/dL - 정상수치가 낮은 경우 : 빈혈, 호흡곤란, 피로감 야기 - 정상수치보다 높은 경우 : 두통, 붉은 반점, 심근경색, 뇌졸중 야기

2. About Data - Domain Data



Urine protein 요단백
serum creatinine 혈청 크레아티닌

type	비고	
float64	[Dipstick검사] 0 : 미검출, 1 : 정상 수치, 2~5 : 이상수치	흡연여부에 따라 현재 흡연자와 과거 흡연자의 요중 크레아티닌 농도가 보정전과 보정 후 모두 유의한 차이를 보임 크레아티닌은 근육에서 생성되는 노폐물로 대부분 신장을 통해 배출되기 때문에 신장기능 검사의 좋은 지표
float64	[정상범위] 0.5-1.4mg/dL 남성 : 0.7-1.2ml/dL 여성 : 0.5-1.0ml/dL	



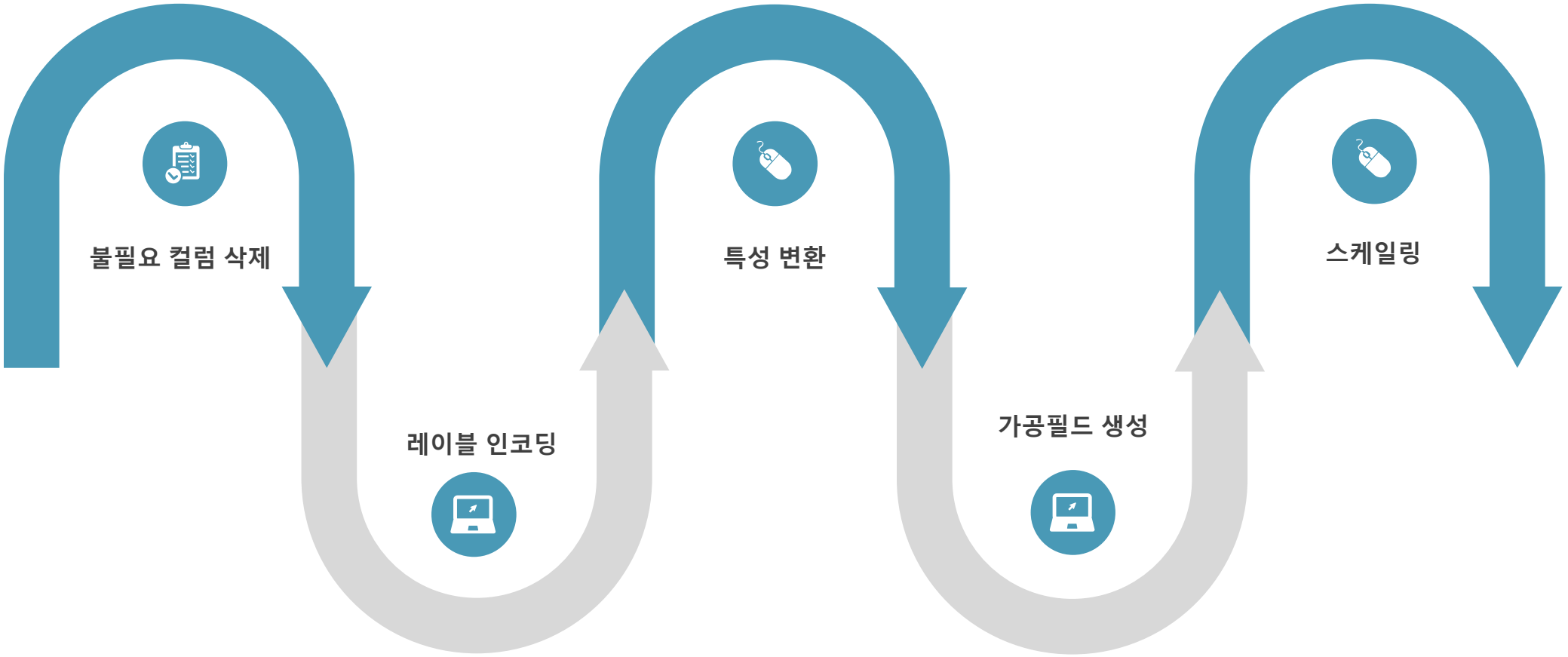
Gtp 감마 지티피
AST 아스파르테이트 아미노 전달효소
ALT 알라닌 아미노 전달효소

type	비고
float64	감마지티피 남성 : 11 ~ 63 IU/L, 여성 : 8 ~ 35 IU/L
float64	아스파르테이트아미노전달효소 적혈구, 골격근(뼈대 근육, skeletal muscle) 등에 분포하는 효소로 세포가 괴사, 파괴되면 혈중으로 유출
float64	알라닌아미노전달효소 간세포 내에 많이 함유되어 있는 효소로 간이 장애를 입으면 혈중 ALT 활성

oral 구강검사 유무
dental caries 충치 유무
tartar 치석 유무

type	비고
object	구강검사상태(Y/N)
int64	충치 (1/0)
object	치석 (Y/N)

2. About Data - Preprocessing



2. About Data - Preprocessing

Step. 1

▶ 불필요 컬럼 제거

- ID : 고유한 id가 아닌 단순 연번
- oral : 모든 데이터값이 같은 값

```
# 불필요한 컬럼 제거
if 'ID' in df_scaling:
    df_scaling = df_scaling.drop("ID", axis = 1)
    if len(df_scaling.columns) == 1:
        return df_scaling
    df_scaling = df_scaling.drop('oral', axis = 1)
```

Step. 2

▶ 레이블 인코딩

- gender, tartar : object를 float으로 변환
- LabelEncoder()사용

```
# 범주형 피처 레이블 인코딩
cate_features = df_scaling[['gender','tartar']]

lbe = LabelEncoder()
lbe.fit_transform(df_scaling["gender"])
df_scaling["gender"] = lbe.fit_transform(df_scaling["gender"])

lbe = LabelEncoder()
lbe.fit_transform(df_scaling["tartar"])
df_scaling["tartar"] = lbe.fit_transform(df_scaling["tartar"])
```

2. About Data - Preprocessing

Step. 3

특성 변환

- 시력 데이터 범주화
  - 0.1~0.9 나쁨(1)
  - 1.0~1.5 보통(2)
  - 1.6~2.0 좋음(3)
  - 9.9 실명(4)
- 청력 데이터 재편집
  - 1, 2로 구성된 데이터를 0, 1로 변환
- Urine protein 범주화
  - 6단계의 데이터를 0~2 단계로 변환

```
# 시력(eyesight) 데이터 범주화
def func(x):
    if x < 1.0 :
        return 1
    elif x < 1.6 :
        return 2
    elif x <= 2.0 :
        return 3
    else :
        return 4

df_scaling['eyesight(left)'] = df_scaling['eyesight(left)'].apply(lambda x:func(x))
df_scaling['eyesight(right)'] = df_scaling['eyesight(right)'].apply(lambda x:func(x))
# hearing 피처 1, 2 => 1, 0으로 변환
df_scaling['hearing(left)'] = df_scaling['hearing(left)'].apply(lambda x: x-2 if x ==2.0 el
df_scaling['hearing(right)'] = df_scaling['hearing(right)'].apply(lambda x: x-2 if x ==2.0

# Urine protein categorizing
x = df_scaling['Urine protein']
for i in range(len(x)):
    if(x[i] == 1.0):
        x[i] = 0
    elif(x[i] == 2.0):
        x[i] = 1
    else:
        x[i] = 2
df_scaling['Urine protein'] = x
```

2. About Data - Preprocessing

Step. 4

▶ 가공필드 생성

- BMI (체질량 지수)
  - bmi = kg/m**2
- WWI (비만 지수)
  - 지방과 근육량을 분리해 비만도를 측정할 수 있는 지수
  - wwi = waist(cm)/np.sqrt(weight(kg))

```
# BMI 지수 계산
# bmi = kg/m^2
df_scaling['bmi'] = df_scaling['weight(kg)']/((df_scaling['height(cm)']*0.01)**2)
# wwi(비만 지수) 지수 계산
df_scaling['wwi'] = df_scaling['waist(cm)']/(df_scaling['weight(kg)'].apply(np.sqrt))
```

2. About Data - Preprocessing

Step. 5

▶ 스케일러 생성

- 연속형 데이터에 각기 다른 스케일러를 비교적용
  - Standard Scaler
  - Min Max Scaler
  - Robust Scaler
  - log Scaler

```
if scaled_form == 'StandardScaler()':
    # Standard scaler
    scaler = StandardScaler()
    scaler.fit(tr_scaled_features) # 훈련 데이터에 fit() 적용

    # 훈련 데이터와 테스트 데이터에 transform()을 통해 변환
    tr_scaled = scaler.transform(tr_scaled_features)
    ts_scaled = scaler.transform(ts_scaled_features)
elif scaled_form == 'RobustScaler()':
    # Robust scaler
    scaler = RobustScaler()
    scaler.fit(tr_scaled_features) # 훈련 데이터에 fit() 적용

    # 훈련 데이터와 테스트 데이터에 transform()을 통해 변환
    tr_scaled = scaler.transform(tr_scaled_features)
    ts_scaled = scaler.transform(ts_scaled_features)
elif scaled_form == 'logScaler' :
    tr_scaled = np.log1p(tr_scaled_features)
    ts_scaled = np.log1p(ts_scaled_features)
     # 데이터 프레임 형태로 변환
    train_log_scaled = pd.DataFrame(tr_scaled, columns = tr_scaled_features.columns)
    train_log_scaled[tr_cate_features.columns] = tr_cate_features
else:
    # MinMax scaler
    scaler = MinMaxScaler()
    scaler.fit(tr_scaled_features) # 훈련 데이터에 fit() 적용

    # 훈련 데이터와 테스트 데이터에 transform()을 통해 변환
    tr_scaled = scaler.transform(tr_scaled_features)
    ts_scaled = scaler.transform(ts_scaled_features)
```

2. About Data - Data Visualization

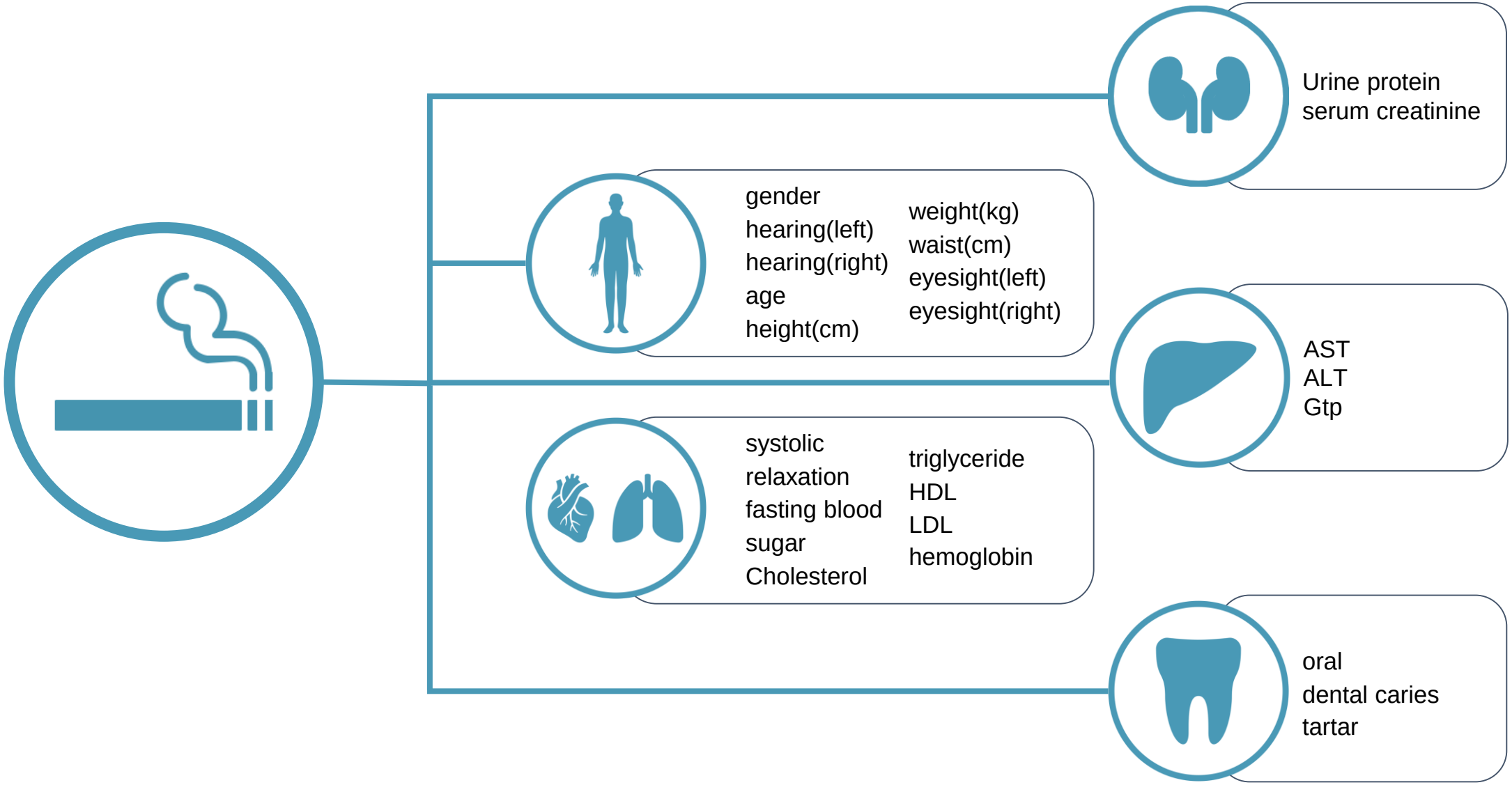
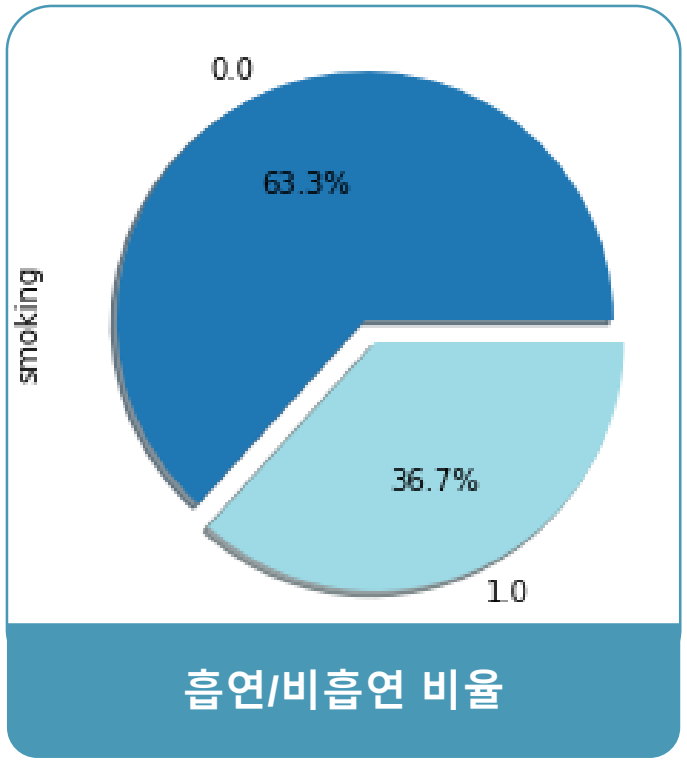


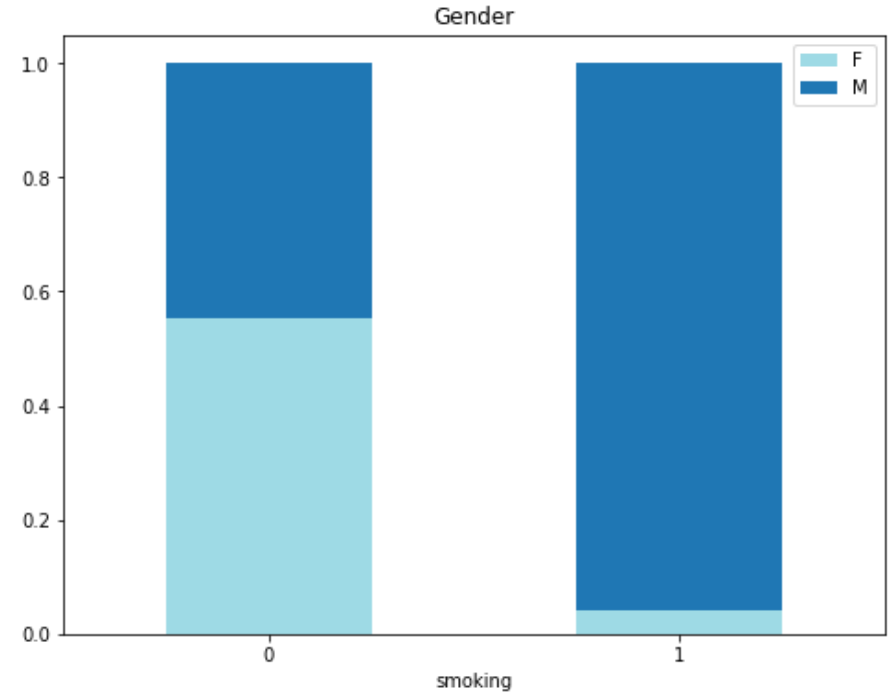
Figure out with body signal of smoking

2. About Data - Data Visualization



흡연/비흡연 비율

- 흡연자 36.7%, 비흡연자 63.3%



gender	F	M
smoking		
0	0.551466	0.448534
1	0.041995	0.958005

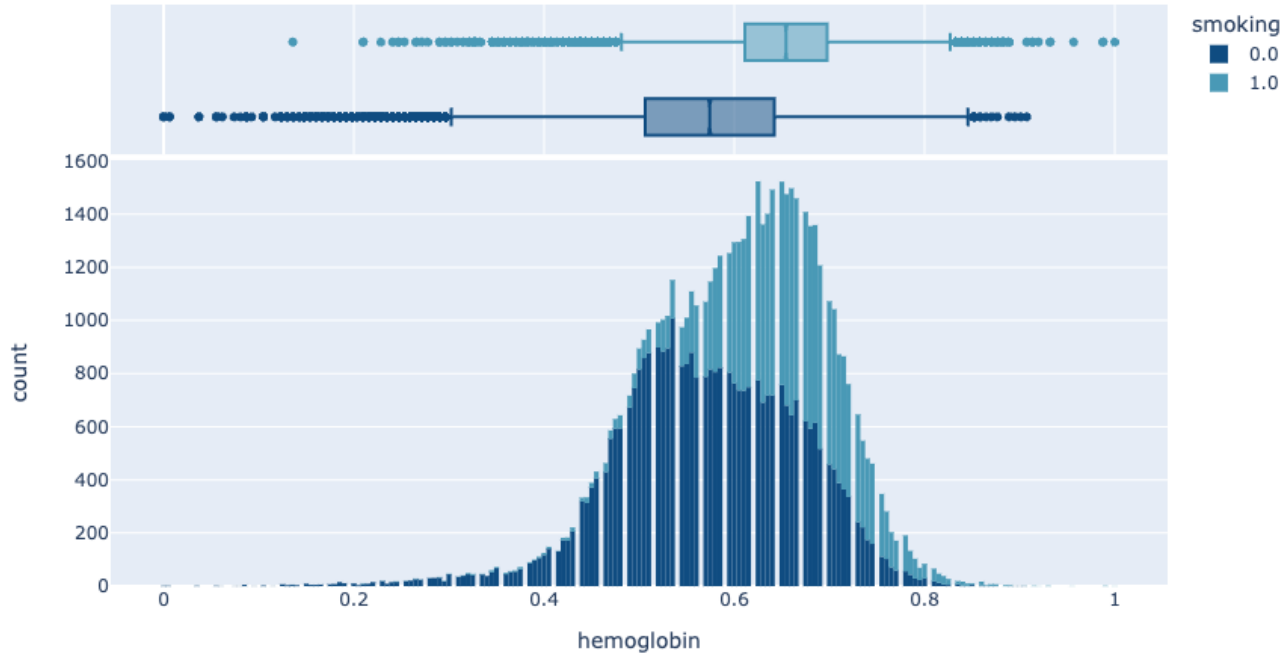
- 흡연자의 성비는 남성 96%/여성 4%
- 비흡연자의 성비는 남성 45%/여성55%

- 흡연자는 남성의 비율이 높고
- 비흡연자는 여성의 비율이 높음

2. About Data - Data Visualization

▶ 혈관 및 호흡계 생체 지표와 흡연과의 상관 분석표

	smoking
systolic	0.077256
relaxation	0.109894
fasting blood sugar	0.104440
Cholesterol	-0.026374
triglyceride	0.257390
HDL	-0.195119
LDL	-0.051816
hemoglobin	0.416544
bmi	0.110766
wwi	-0.054379
smoking	1.000000

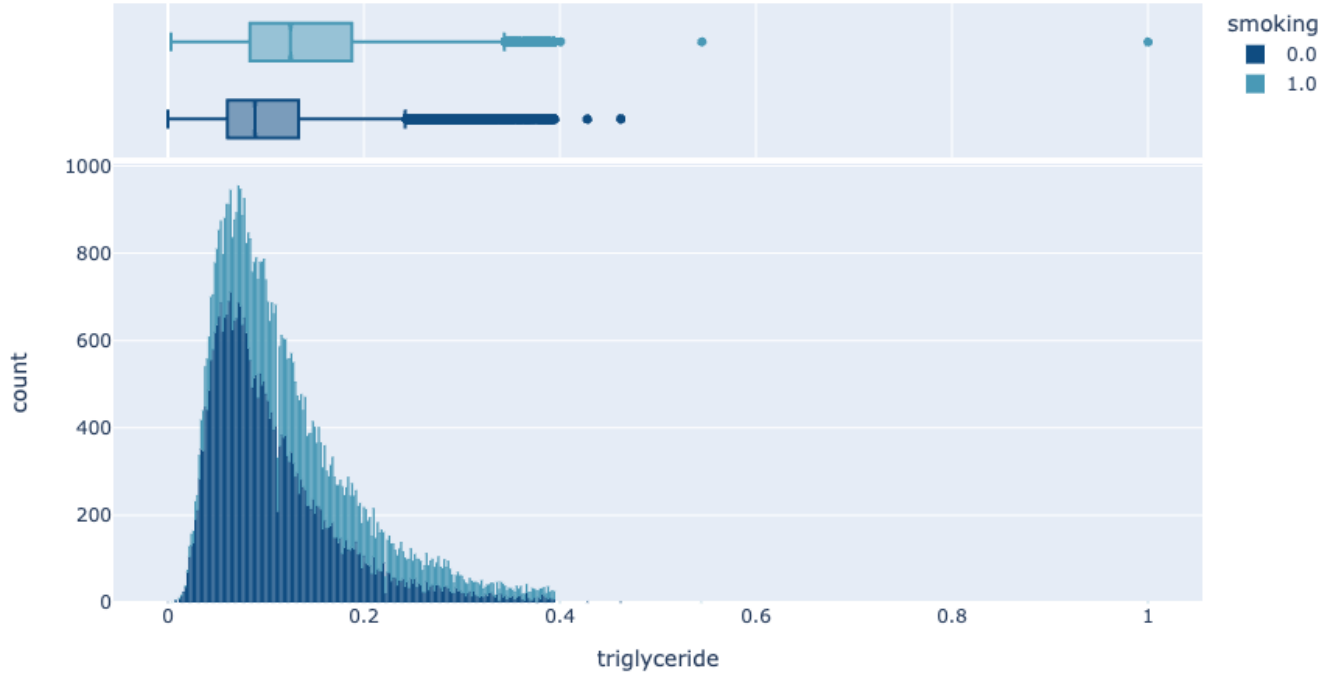


- 혈관 및 호흡의 생체 지표 중 흡연과 상관성이 높은 3개의 항목
  * hemoglobin, triglyceride(중성지방), HDL
- 가공 필드인 bmi, wwi과 흡연의 상관성은 각각 0.11, -0.06으로 상관도가 낮음

2. About Data - Data Visualization

▶ 혈관 및 호흡계 생체 지표와 흡연과의 상관 분석표

	smoking
systolic	0.077256
relaxation	0.109894
fasting blood sugar	0.104440
Cholesterol	-0.026374
triglyceride	0.257390
HDL	-0.195119
LDL	-0.051816
hemoglobin	0.416544
bmi	0.110766
wwi	-0.054379
smoking	1.000000

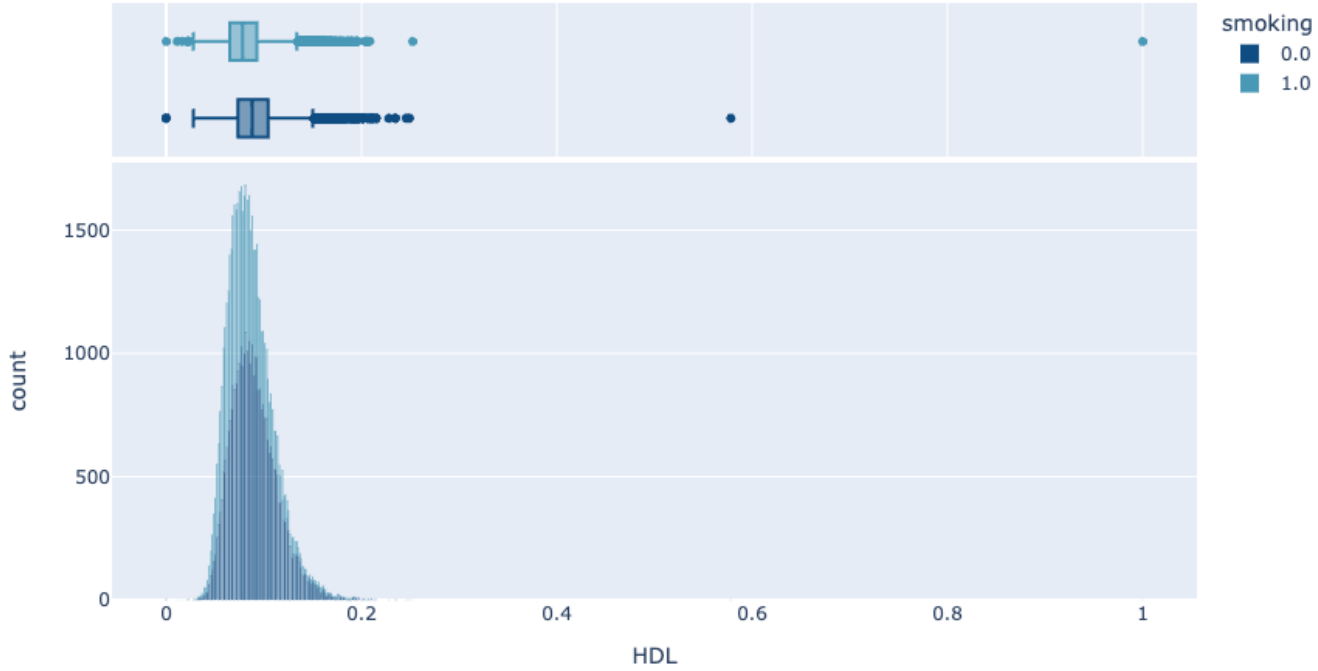


- 혈관 및 호흡의 생체 지표 중 흡연과 상관성이 높은 3개의 항목
  * hemoglobin, triglyceride(중성지방), HDL
- 가공 필드인 bmi, wwi과 흡연의 상관성은 각각 0.11, -0.06으로 상관도가 낮음

2. About Data - Data Visualization

▶ 혈관 및 호흡계 생체 지표와 흡연과의 상관 분석표

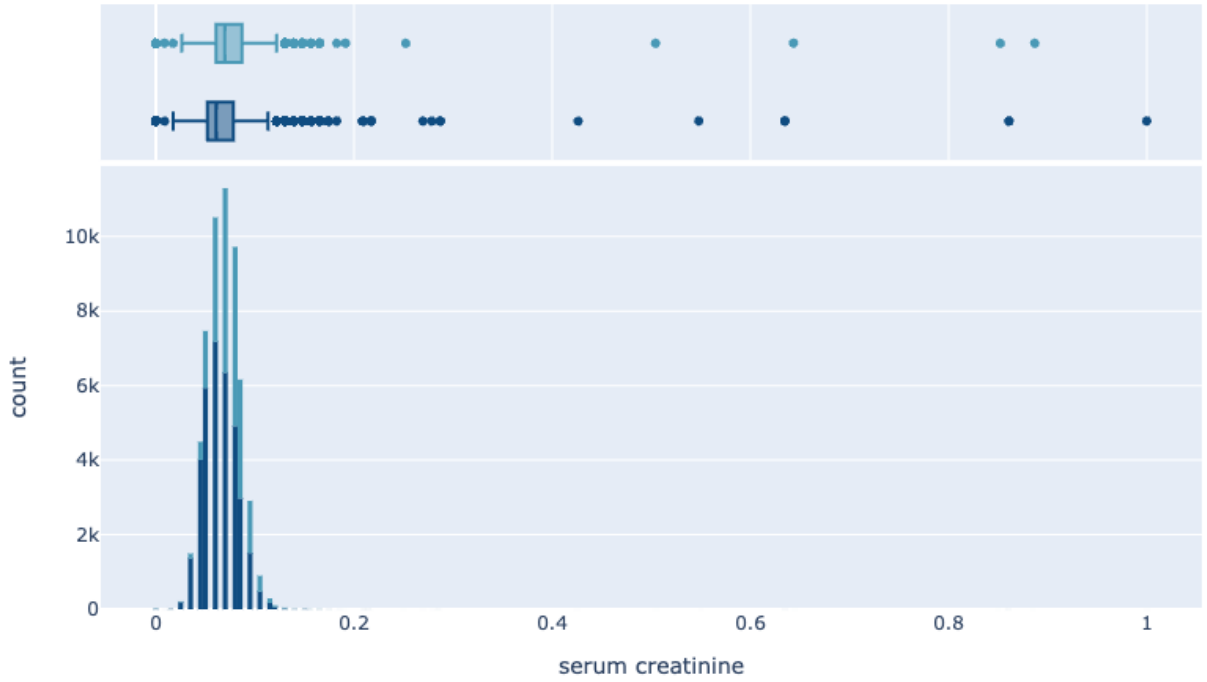
	smoking
systolic	0.077256
relaxation	0.109894
fasting blood sugar	0.104440
Cholesterol	-0.026374
triglyceride	0.257390
HDL	-0.195119
LDL	-0.051816
hemoglobin	0.416544
bmi	0.110766
wwi	-0.054379
smoking	1.000000



- 혈관 및 호흡의 생체 지표 중 흡연과 상관성이 높은 3개의 항목
- hemoglobin, triglyceride(중성지방), HDL
- 가공 필드인 bmi, wwi와 흡연의 상관성은 각각 0.11, -0.06으로 상관도가 낮음

2. About Data - Data Visualization

▶ 신장 질환 생체 지표와 흡연과의 상관 분석표

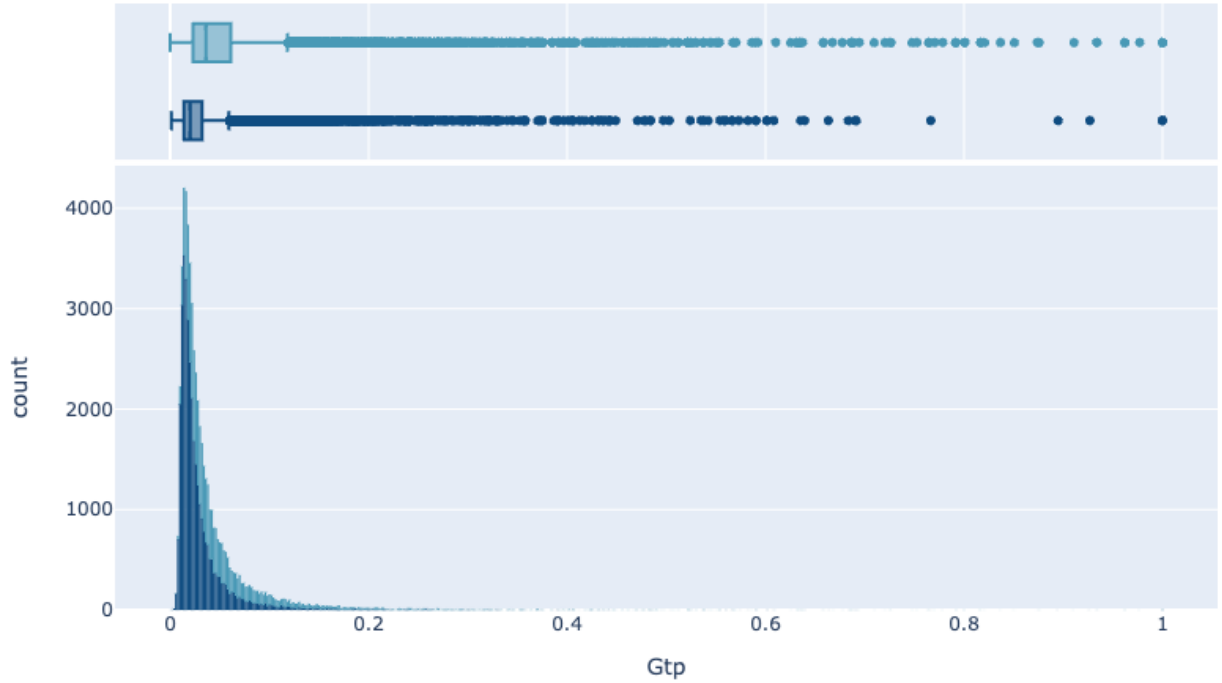


	Urine protein	serum creatinine	smoking
Urine protein	1.000000	0.087682	0.014267
serum creatinine	0.087682	1.000000	0.216812
smoking	0.014267	0.216812	1.000000

- serum creatinine(혈청 크레아티닌)의 흡연과 상관성은 0.21로 Urine protein보다 상관성이 상대적으로 높음

2. About Data - Data Visualization

▶ 간의 생체 지표와 흡연과의 상관 분석표



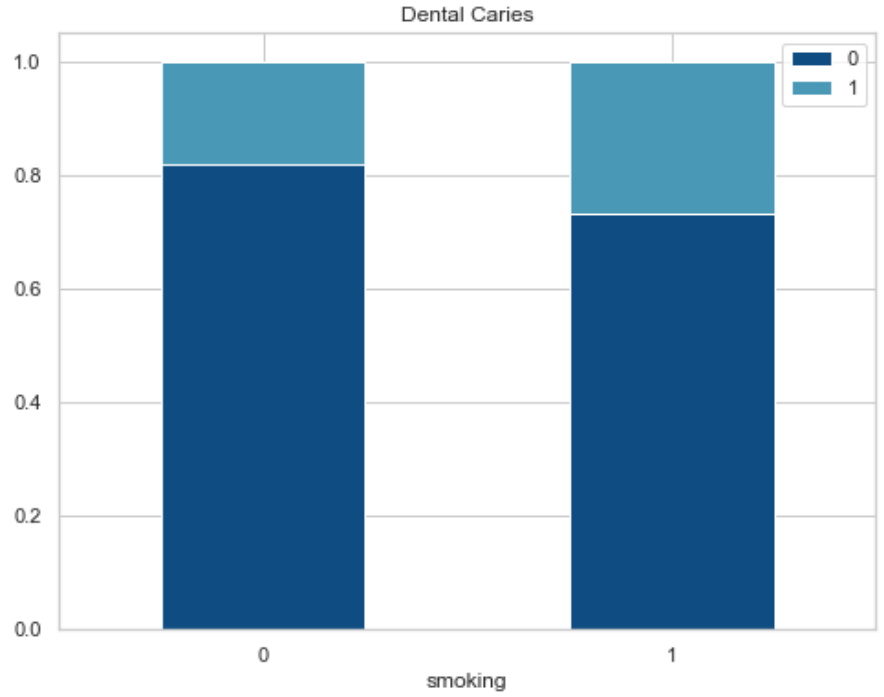
	AST	ALT	Gtp	smoking
AST	1.000000	0.740726	0.379959	0.059253
ALT	0.740726	1.000000	0.343934	0.097338
Gtp	0.379959	0.343934	1.000000	0.236619
smoking	0.059253	0.097338	0.236619	1.000000

- 간수치 지표 중 Gtp가 흡연과의 상관성이 있음

2. About Data - Data Visualization

▶ 구강 건강과 흡연/비흡연의 비율

		dental caries
smoking	dental caries	
0.0	0	0.819082
	1	0.180918
1.0	0	0.730824
	1	0.269176



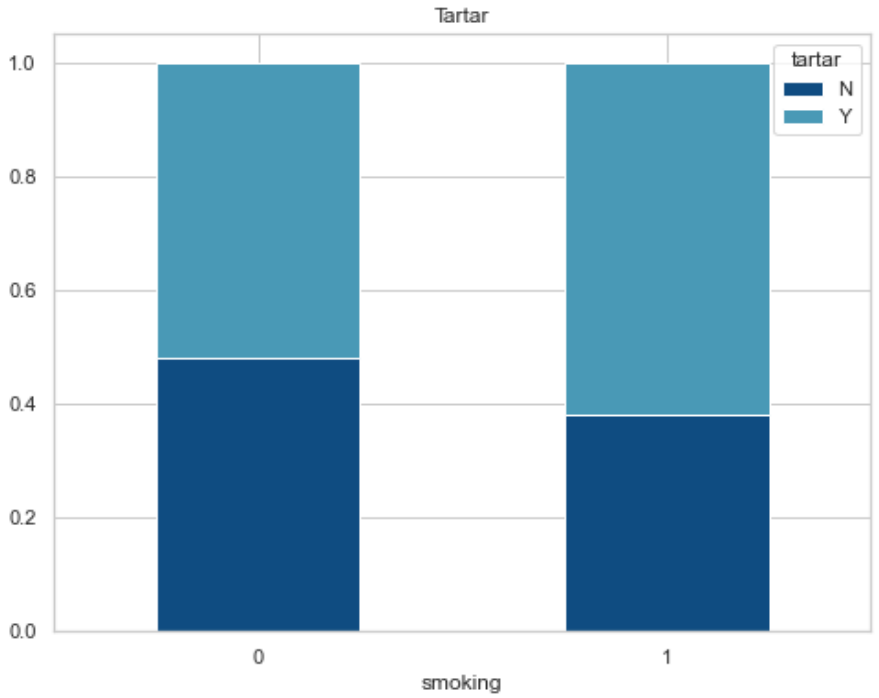
- 흡연자 중 충치가 있는 비율 27%, 흡연자 중 충치가 없는 비율 73%
- 비흡연자 중 충치가 있는 비율 18%, 비흡연자 중 충치가 없는 비율 82%
- **비흡연자 보다 흡연자의 충치비율이 9%정도 많음**

2. About Data - Data Visualization

▶ 구강 건강과 흡연/비흡연의 비율

tartar

smoking	tartar	
0.0	1	0.518205
	0	0.481795
1.0	1	0.619897
	0	0.380103



- 흡연자 중 치석이 있는 비율 52%, 흡연자 중 치석이 없는 비율 48%
- 비흡연자 중 치석이 있는 비율 62%, 비흡연자 중 치석이 없는 비율 38%
- **비흡연자 보다 흡연자의 치석비율이 10%정도 많음**

3. Modeling

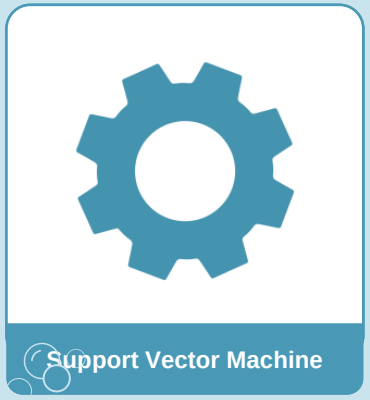


Figure out with body signal of smoking

Decision Tree

CONTENTS

01 교차 검증

02 최적 하이퍼파라미터 탐색

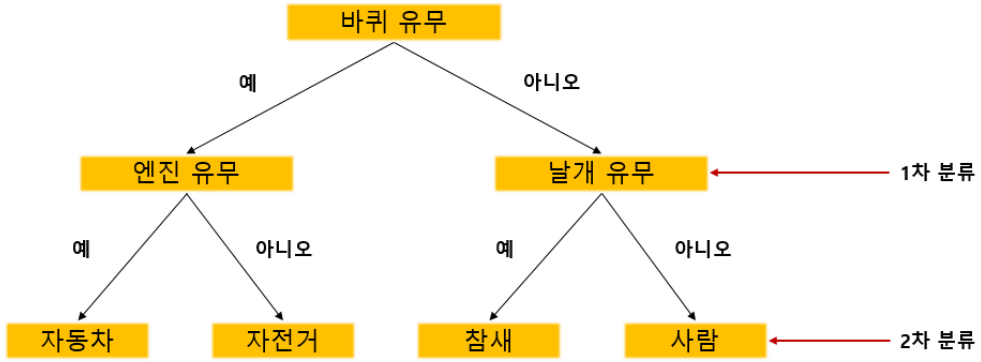
03 최종 모델 결정

Decision Tree

▶ 결정트리 소개

- 머신러닝 알고리즘 중 직관적으로 이해하기 쉬운 알고리즘
- 일반적으로 if/else기반으로 분류
- 다른 분류모델과 비교해 성능이 좋지 못함

예시



▶ 기본 모델링

- 불필요 컬럼 제거, 레이블 인코딩, 가공필드 생성
- 기본 결정트리의 성능은 좋지 못함
- train 정확도는 1이 나온 반면 test 데이터의 정확도는 0.62 과소적합이 추정됨
- 교차검증 및 하이퍼파라미터 조정으로 정확도 개선 필요

```
from sklearn.tree import DecisionTreeClassifier
import sklearn.model_selection as ms
import sklearn.metrics as mt

# 결정트리 객체 생성 및 훈련
dt_clf = DecisionTreeClassifier()
dt_clf.fit(X_train,y_train)

# 예측값 저장
y_pred = dt_clf.predict(X_test)
print('Train_Accuracy: ', dt_clf.score(X_train, y_train))

accuracy = mt.accuracy_score(y_test, y_pred)
recall = mt.recall_score(y_test, y_pred)
precision = mt.precision_score(y_test, y_pred)
f1_score = mt.f1_score(y_test, y_pred)
matrix = mt.confusion_matrix(y_test, y_pred)
auc = mt.roc_auc_score(y_test, y_pred)
```

	기본 모델
정확도	0.6228
정밀도	0.4853
재현율	0.4723
F1 score	0.4831
AUC score	0.5912

Decision Tree

▶1. 교차 검증

- 교차검증의 모든 평가지표가 기본 모델링보다 높음
- 교차검증의 평균 또한 0.68임을 확인
- 기본 모델보다 정확도가 높은 결정트리를 생성할 가능성을 확인

In [17]:
```
from sklearn.model_selection import cross_val_score, cross_validate

scores = cross_val_score(dt_clf, result1, result2, cv = 5)
scores
```

Out[17]:
```
array([0.68767394, 0.68848191, 0.6897109 , 0.68818459, 0.63485365])
```

In [18]:
```
print('교차검증 평균: ', scores.mean())
```

```
교차검증 평균:  0.6777809991767135
```

Decision Tree

▶ 2.하이퍼파라미터 조정

- GridSearchCV 사용
- max_depth, min_samplts_split, splitter 파라미터를 조정
- 가장 높은 정확도를 나타내는 모델은
  - **max_depth [5]**
  - **min_samples_split [3]**
  - **splitter['best']**
  - **정확도는 0.74를 보임**

```
In [23]: from sklearn.model_selection import GridSearchCV
         from sklearn.model_selection import RandomizedSearchCV

         dt_clf = DecisionTreeClassifier(random_state=0)

         parameters = {
                       'max_depth': [5, 7, 9],
                       'min_samples_split': [3, 5, 7],
                       'splitter': ['best']}

         grid_dt = GridSearchCV(dt_clf,
                                param_grid = parameters, cv = 5)

         grid_dt.fit(X_train, y_train)

         result = pd.DataFrame(grid_dt.cv_results_['params'])
         result['mean_test_score'] = grid_dt.cv_results_['mean_test_score']
         result.sort_values(by='mean_test_score', ascending=False)
```

Out[23]:

	max_depth	min_samples_split	splitter	mean_test_score
0	5	3	best	0.739052
1	5	5	best	0.739052
2	5	7	best	0.739052
3	7	3	best	0.737997
4	7	5	best	0.737930
5	7	7	best	0.737885
8	9	7	best	0.734294
6	9	3	best	0.734227
7	9	5	best	0.734025

Decision Tree

▶ 3. 최종모델 선정

- **max_depth [5]**
- **min_samples_split [3]**
- **splitter[‘best’]**
- 최종모델로 위의 3개의 파라미터를 조정한 모델로 결정
- train 정확도와 최종모델의 정확도 수치는 0.74와 0.73
- 기본모델에 있었던 과소적합이 해결됨
- **최종 모델의 모든 지표가 기본모델보다 향상됨**

	정확도	정밀도	재현율	f1_score	auc_score
기본 모델	0.6228	0.4853	0.4723	0.4831	0.5912
최종 모델	0.7334	0.6036	0.8012	0.6923	0.7541

In [24]:

```
from sklearn.tree import DecisionTreeClassifier
import sklearn.model_selection as ms
import sklearn.metrics as mt

# 결정트리 객체 생성 및 훈련
dt_clf = DecisionTreeClassifier(random_state=0,max_depth=5,
                                min_samples_split=3,splitter='best')
dt_clf.fit(X_train,y_train)

# 예측값 저장
y_pred = dt_clf.predict(X_test)
print('Train_Accuracy: ', dt_clf.score(X_train, y_train))

accuracy = mt.accuracy_score(y_test, y_pred)
recall = mt.recall_score(y_test, y_pred)
precision = mt.precision_score(y_test, y_pred)
f1_score = mt.f1_score(y_test, y_pred)
matrix = mt.confusion_matrix(y_test, y_pred)
auc = mt.roc_auc_score(y_test, y_pred)
```

```
Train_Accuracy:  0.7422844701815815
```

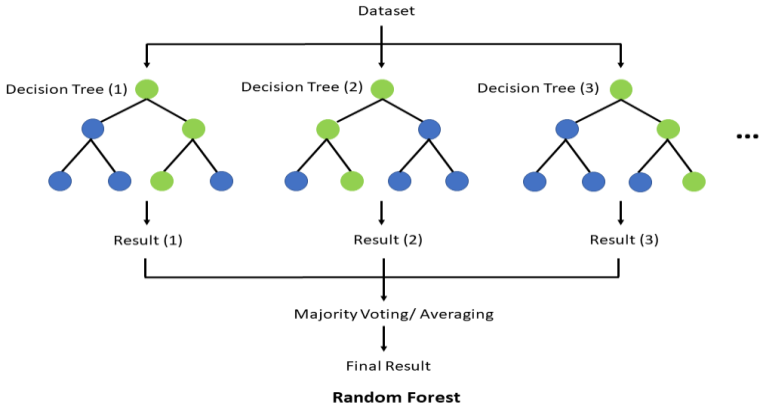
Random Forest

CONTENTS

Random Forest

랜덤 포레스트 소개

- 분류 회귀분석 등에 사용되는 앙상블 학습 방법의 일종
- 훈련과정에서 구성한 다수의 결정트리로부터 분류 또는 평균예측치를 출력해 동작함
  - 결정트리가 기반알고리즘
  - 앙상블 알고리즘 중 수행속도가 빠름
  - 다양한 영역에서 높은 예측 성능을 보임



주요 파라미터

- **n_estimators** : 트리 수 (클수록 성능은 좋아질 수 있지만 시간 오래 걸림)
- **max_depth** : 트리의 최대 깊이 (과적합 개선하기 위해 사용)
- **min_samples_leaf** : 리프 노드에 있는 최소 샘플
- **Min_samples_split** : 내부 노드를 분할하는데 필요한 최소 샘플
- **n_jobs** : -1로 설정 시 CPU 전부 사용
- **Random_state**: 난수 고정 결과가 바뀌않음

Random Forest

1. 기본 모델링

```
rf_pred_probs = rf_clf.predict_proba(x_test)[:, 1]
```

```
def get_clf_eval(y_test, pred=None, pred_proba=None):
    confusion = confusion_matrix(y_test,pred)
    accuracy = accuracy_score(y_test,pred)
    precision = precision_score(y_test,pred)
    recall = recall_score(y_test,pred)
    f1 = f1_score(y_test,pred)
    # ROC-AUC 추가
    roc_auc = roc_auc_score(y_test,pred_proba)
    print('오차 행렬')
    print(confusion)
    # ROC-AUC print 추가
    print('정확도: {0:.4f}, 정밀도: {1:.4f}, 재현율: {2:.4f},₩
    F1: {3:.4f}, AUC:{4:.4f}'.format(accuracy, precision, recall, f1, roc_auc))
```

```
get_clf_eval(y_test, rf_pred, rf_pred_probs)
```

```
오차 행렬
[[7036    0]
 [   3 4100]]
정확도: 0.9997, 정밀도: 1.0000, 재현율: 0.9993,    F1: 0.9996, AUC:1.0000
```

- 기본 모델링을 진행했으나 과적합이 일어난 것으로 예상됨
- 과적합을 해결하기 위해 파라미터를 조정함

하이퍼 파라미터	n_estimator	max_depth	min_sample_split	min_samples_leaf	max_features
	100	inf	2	1	sqrt(n_features)
	50	12	8	18	7

2 .최적 하이퍼파라미터 탐색

- 시력데이터 중 실명(4) 데이터를 보유한 행 삭제
- 실명을 이상치로 간주함
- 시력 데이터에 Robust Scaler 적용
- 시력 데이터 범주화
  - 0.1~0.9 나쁨(1)
  - 1.0~1.5 보통(2)
  - 1.6~2.0 좋음(3)
  - 9.9 실명(4)

최적 하이퍼 파라미터
max_depth : 12
min_samples_leaf: 18
min_samples_split: 8
n_estimators: 50

```
params = {
    'n_estimators':[50],
    'max_depth' : [6,8,10,12],
    'min_samples_leaf' : [8,12,18],
    'min_samples_split' : [8,16,20]
}

rf_clf = RandomForestClassifier(random_state=0, n_jobs=-1)

grid_cv = GridSearchCV(rf_clf, param_grid=params, cv=2, n_jobs=-1)
grid_cv.fit(x_train_rob, y_train)

print('최적 하이퍼 파라미터:₩n', grid_cv.best_params_)
print('최고 예측 정확도: {0:.4f}'.format(grid_cv.best_score_))
```

	정확도	정밀도	재현율
기본 모델	0.9997	1.0	0.9993
튜닝 모델	0.8285	0.7437	0.8155

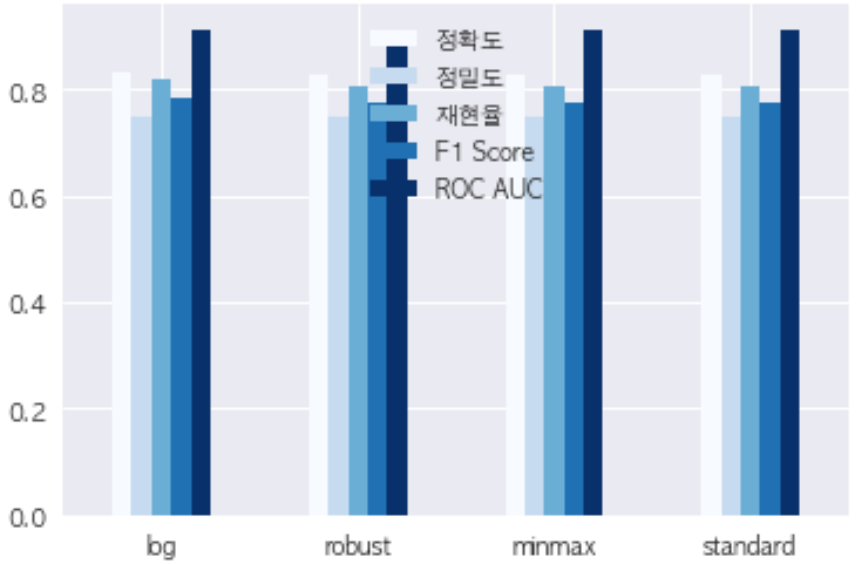
Figure out with body signal of smoking

Random Forest

3. 스케일러 변경

	정확도	정밀도	재현율	F1	AUC
log	0.8329	0.7496	0.8206	0.7835	0.9127
robust	0.8287	0.7486	0.8055	0.7760	0.9119
minmax	0.8285	0.7482	0.8055	0.7758	0.9119
standard	0.8285	0.7482	0.8055	0.7758	0.9119

- 앞서 조정한 하이퍼튜닝 값을 적용
- 스케일러를 변경하며 모델링함
- 평가지표에 유의미한 개선을 보이지는 못함



n_estimator	max_depth	min_sample_split	min_samples_leaf	max_features
50	12	8	18	7

Random Forest

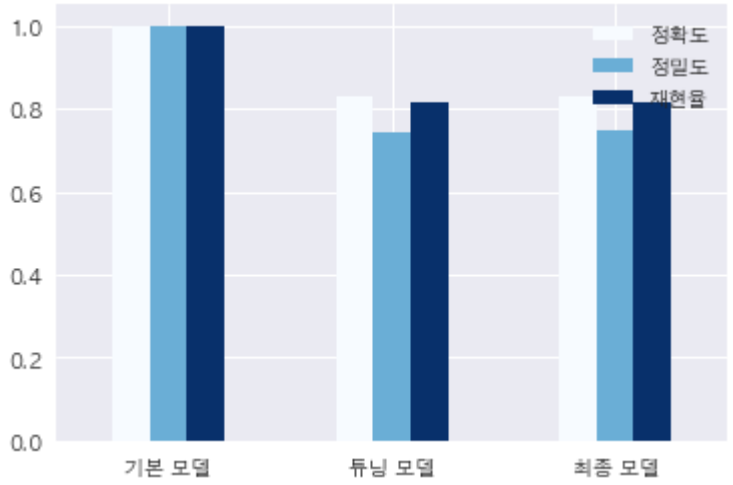
4. 최종 모델 선정

- 기본 전처리
- 하이퍼 파라미터 튜닝
- 스케일러 미적용

- 최종 모델에 사용한 하이퍼 파라미터 :
  - max_depth: 12
  - min_samples_leaf: 8
  - min_samples_split: 8
  - n_estimators: 50

- 기본모델의 과적합은 어느정도 해결한 것으로 보임
- 튜닝 모델에 비해 비약적인 성능향상은 없었으나
- 정확도가 약간 개선된 모델을 최종모델로 선정함

	정확도	정밀도	재현율
기본 모델	0.9997	1.0	0.9993
튜닝 모델	0.8285	0.7437	0.8155
최종 모델	0.8310	0.7492	0.8135



Gradient Boosting Model

CONTENTS

Figure out with body signal of smoking

Gradient Boosting Model

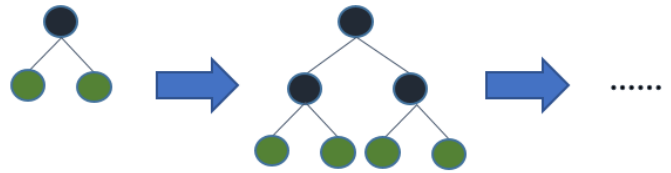
1. 모델 소개

XGB

- CART 기반으로 제작된 모델
- Gradient Boosting Model 에서의 시간이 오래 걸리는 단점을 보완하기 위해 고안된 모델
- 병렬 연산을 지원하여 속도 향상 (GPU 사용 옵션)
- 적절한 데이터 튜닝이 이루어지지 않을 경우, Overfit이 발생하기 쉬움

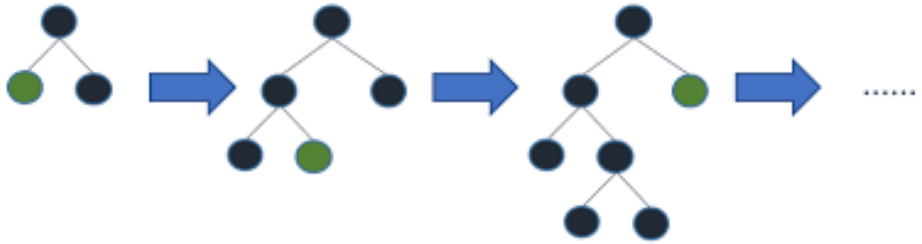
Light GBM

- Gradient-based One-Side Sampling 기술을 사용함
  - 가중치가 작을 때 상수를 적용하여 데이터를 증폭하여 데이터의 분포 정도를 줄임
- 일반적인 GBM 계열과는 다르게 leaf-wise growth방식을 채택하여 시간과 메모리를 절약
- 충분한 데이터 양이 없을 경우, overfitting 이 일어나기 쉬움



Level-wise tree growth

- **GBM : 여러 개의 모델(트리)을 사용하여 각각의 예측 결과를 생성하고 이를 결합하여 하나의 모델을 제작**
- **Boosting : Ensemble 기법 중 한 종류**



Leaf-wise tree growth

Gradient Boosting Model

2. 최적 하이퍼파라미터 탐색

- **주요 하이퍼 파라미터**
  - num_rounds = num_iterations : 수행 반복 횟수
  - eta = learning rate : 학습률 (일반적으로 0.01 - 0.2)
  - max_depth: Tree 깊이 수 (과적합 조절 용도로 사용됨)
  - min_child_samples : 최종 결정 클래스인 Leaf Node가 되기 위해 최소한으로 필요한 데이터 개체의 수
  - min_child_weight: 관측치에 대한 가중치 합의 최솟값, 이를 기준으로 추가 분기 결정 (크면 Underfitting)
  - subsample : 학습 시 데이터 샘플링 비율을 지정(과적합 제어)
  - colsample_bytree : 각 트리마다의 feature 샘플링 비율
  - gamma: split 하기 위한 최소의 loss 감소 정의

XGB

- **최적 파라미터**
  - max_depth = 6, min_child_weight = 5
  - gamma = 0
  - subsample = 0.8, colsample_bytree = 0.9

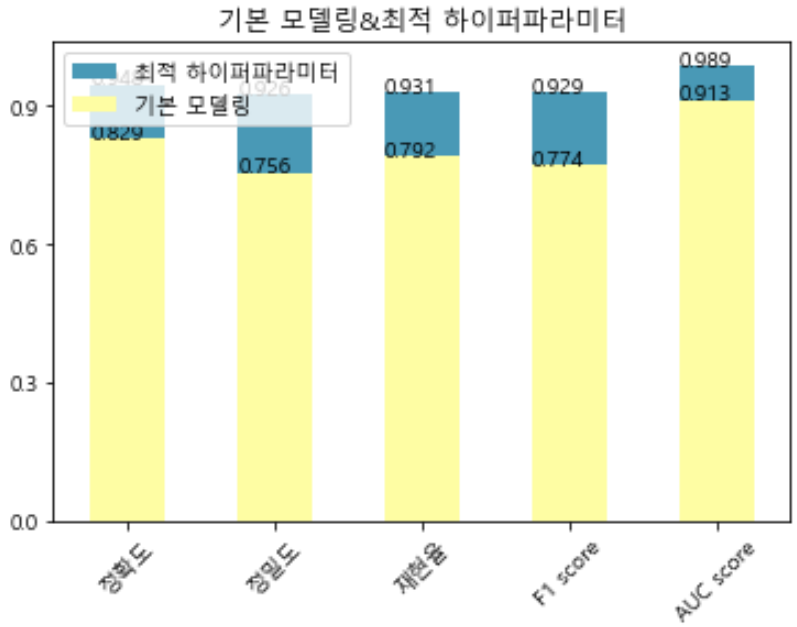
LGBM

- **최적 파라미터**
  - num_iterations = 1000, max_depth = 12
  - min_child_samples = 5, min_child_weight = 5
  - subsample = 0.8, colsample_bytree = 0.8

Gradient Boosting Model

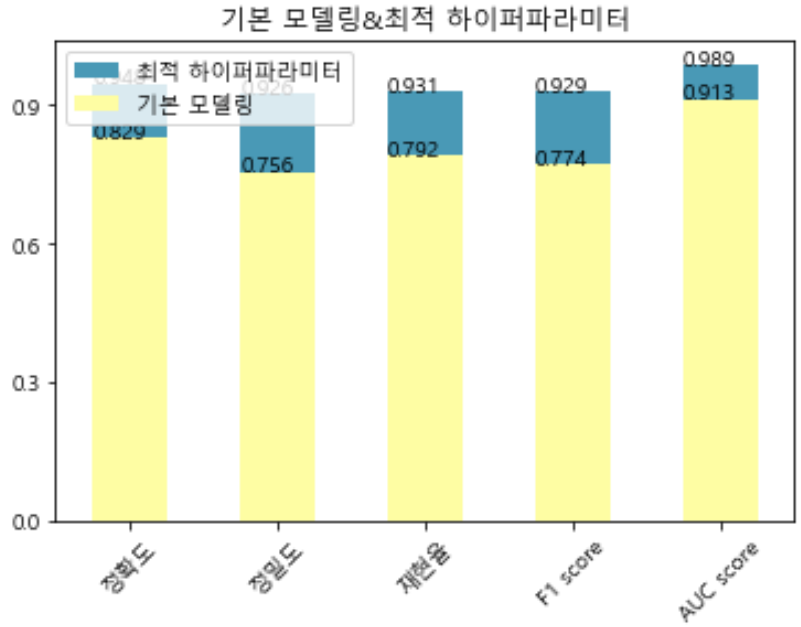
2. 최적 하이퍼파라미터 탐색

XGB



	정확도	정밀도	재현율	F1 score	AUC score
Non_scaled	0.8074	0.7306	0.7560	0.7431	0.8908
Hyperparameter tuned	0.9454	0.9272	0.9244	0.9258	0.9867

LGBM

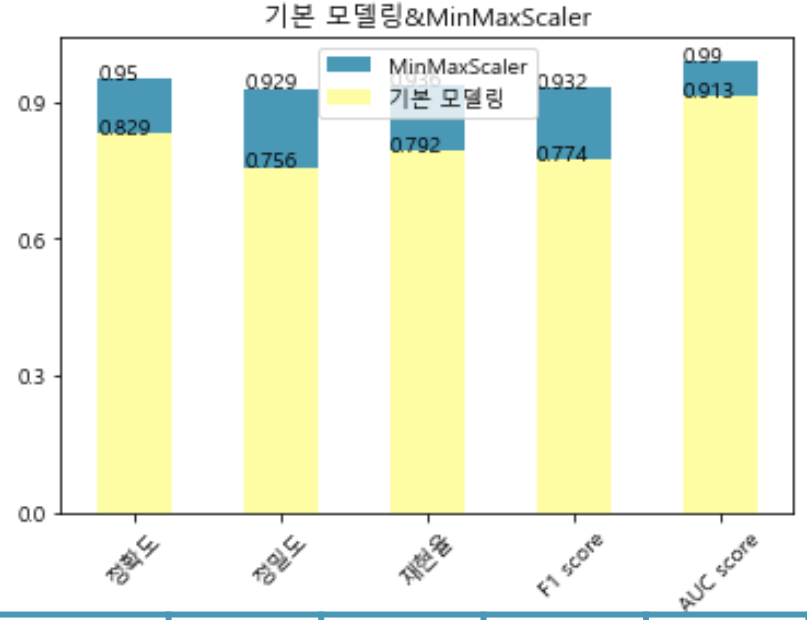


	정확도	정밀도	재현율	F1 score	AUC score
Non_scaled	0.8294	0.7565	0.7919	0.7738	0.9129
Hyperparameter tuned	0.9476	0.9265	0.9315	0.9290	0.9890

Gradient Boosting Model

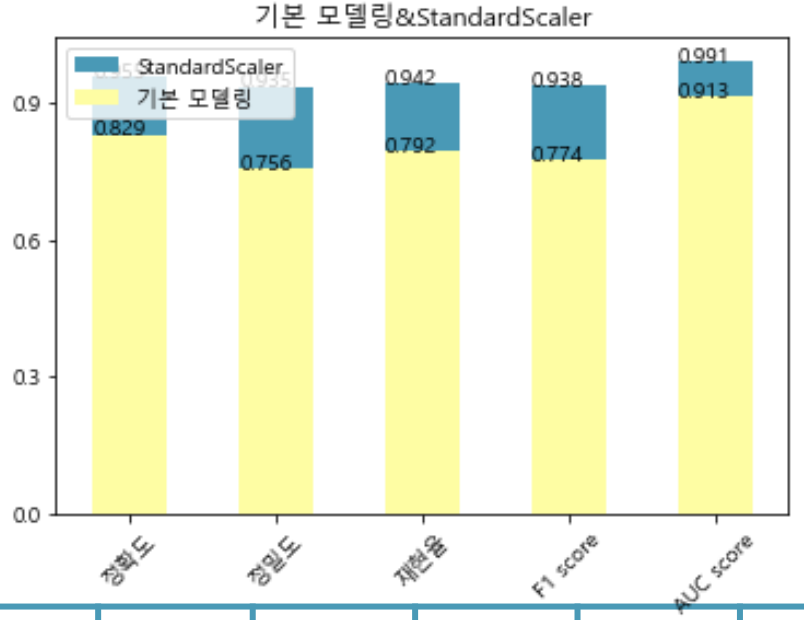
3. 스케일러 적용

XGB



스케일러	정확도	정밀도	재현율	F1 score	AUC score
Non_scaled	0.8074	0.7306	0.7560	0.7431	0.8908
Standard Scaler	0.9430	0.9244	0.9205	0.9225	0.9861
Robust Scaler	0.9444	0.9276	0.9210	0.9243	0.9870
min Max Scaler	0.9483	0.9313	0.9281	0.9297	0.9870
Log Scale	0.9444	0.9238	0.9254	0.9246	0.9867

LGBM

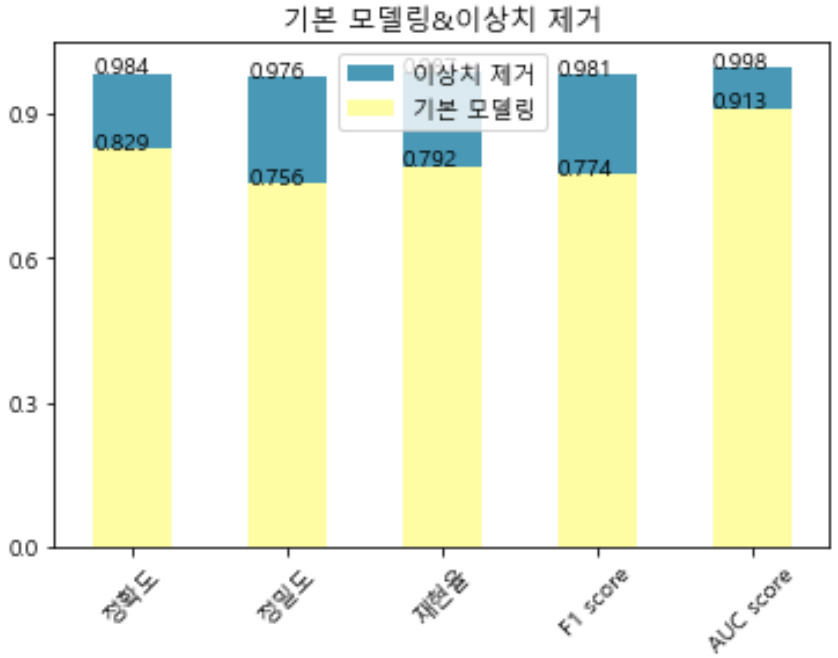


스케일러	정확도	정밀도	재현율	F1 score	AUC score
Non_scaled	0.8294	0.7565	0.7919	0.7738	0.9129
Standard Scaler	0.9545	0.9351	0.9417	0.9384	0.9909
Robust Scaler	0.9522	0.9312	0.9398	0.9355	0.9903
minMax Scaler	0.9501	0.9287	0.9364	0.9325	0.9900
Log Scale	0.9521	0.9274	0.9437	0.9355	0.9904

Gradient Boosting Model

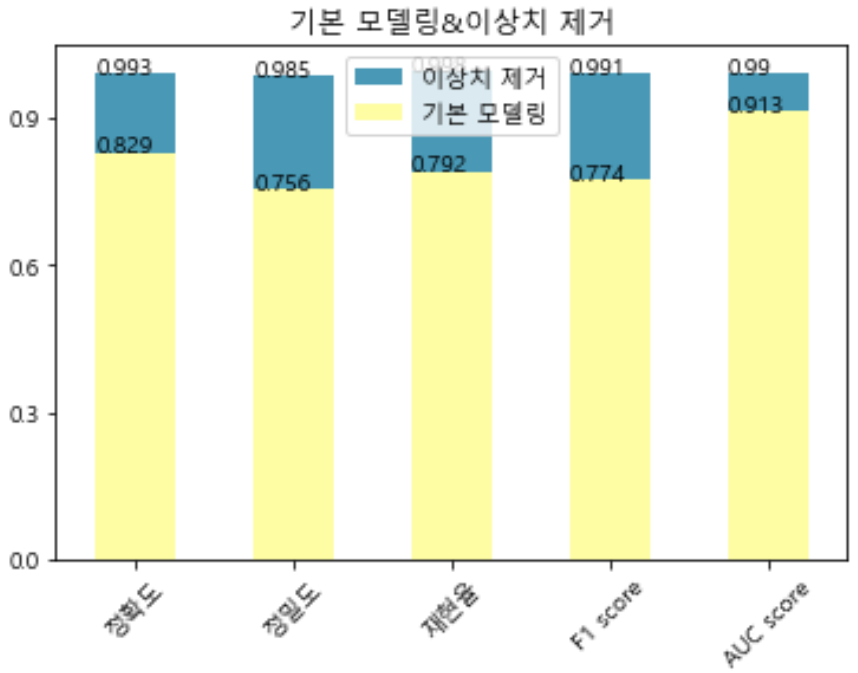
4. 이상치 제거

XGB



	정확도	정밀도	재현율	F1 score	AUC score
Non_scaled	0.8074	0.7306	0.7560	0.7431	0.8908
Remove outliers	0.9844	0.9758	0.9867	0.9812	0.9981

LGBM



	정확도	정밀도	재현율	F1 score	AUC score
Non_scaled	0.8074	0.7306	0.7560	0.7431	0.8908
Remove outliers	0.9928	0.9853	0.9975	0.9913	0.9904

XGBoost 최종 모델 비교

기본 모델링		최종 모델링
0.8074	정확도	0.9839
0.7306	정밀도	0.9744
0.7560	재현율	0,9868
0.7431	F1 SCORE	0.9805
0.8908	AUC SCORE	0.9989

Light GBM 최종 모델 비교

기본 모델링		최종 모델링
0.8294	정확도	0.9927
0.7565	정밀도	0.9861
0.7919	재현율	0.9963
0.7738	F1 SCORE	0.9912
0.9129	AUC SCORE	0.9996

Logistic Regression

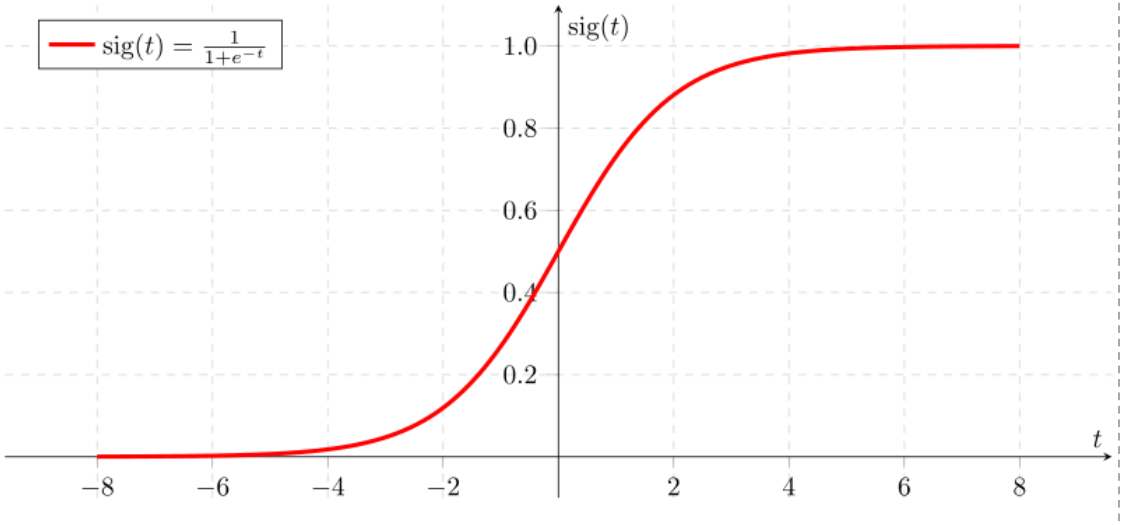
CONTENTS

Figure out with body signal of smoking

Logistic Regression

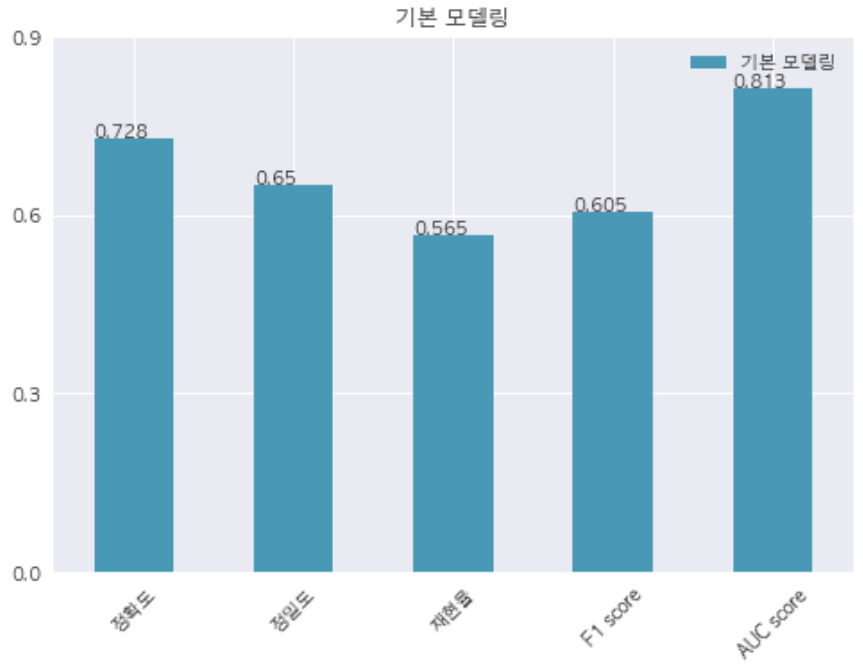
▶ 로지스틱 회귀

- **선형회귀 방식을 기반으로 시그모이드 함수를 이용**
- **로지스틱 회귀 분류에서 사용하는 주요 파라미터**
  - solver : ibfgs(기본값), liblinear, newton-cg , sag, saga
  - C (CostFunction) : 규제 강도
  - penalty : solver 값에 따라 L1, L2 규제



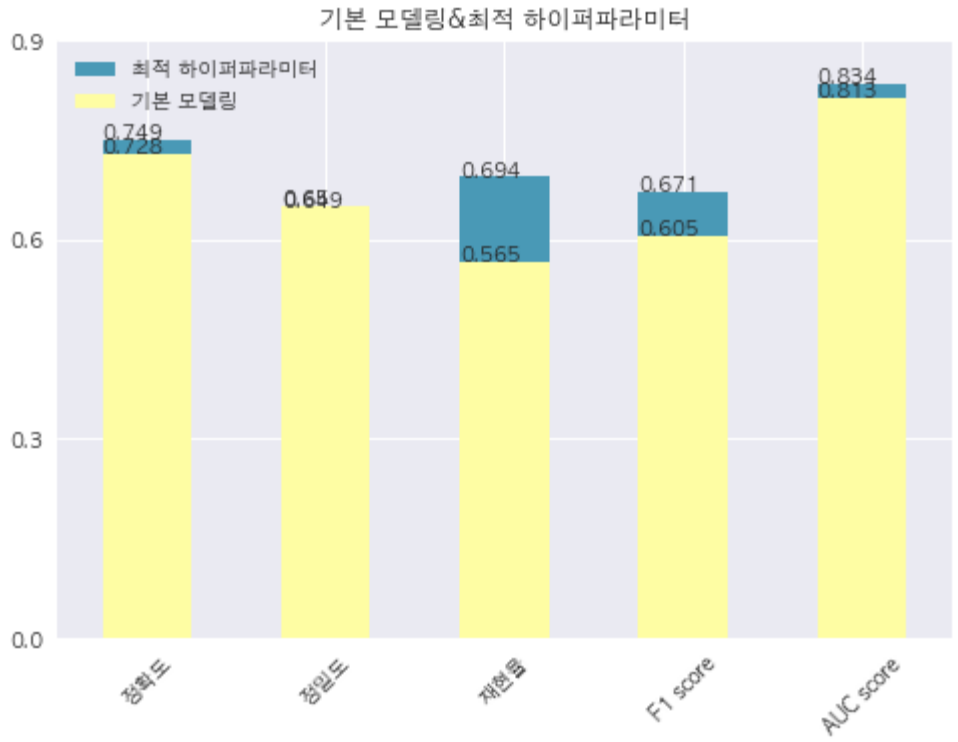
▶ 기본 모델링

```
start = time.time()
# 1.3 학습/예측
lr_clf = LogisticRegression()
lr_clf.fit(x_train, y_train)
lr_pred = lr_clf.predict(x_test)
pred_prob = lr_clf.predict_proba(x_test)[:,1]
```



Logistic Regression

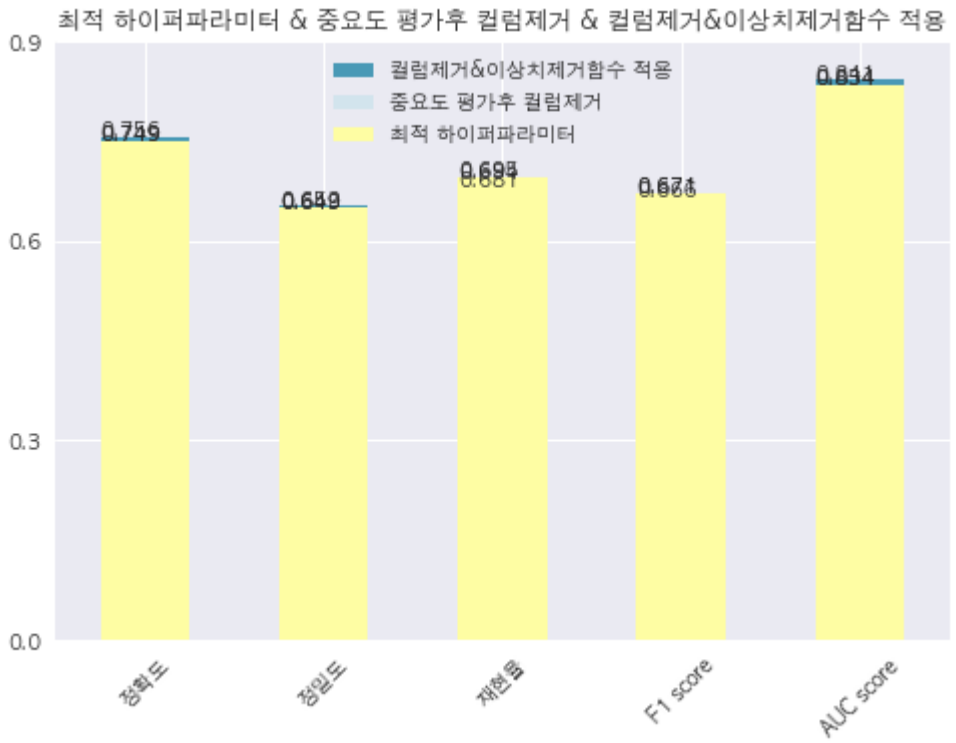
▶1. 최적 하이퍼파라미터 탐색



- **최적 파라미터**
  - GridSearchCV()를 통해 최적 파라미터 탐색
  - {'C' : 11, 'penalty':l1, 'solver' : 'liblinear'}, 최적 평균 정확도 0.762
  - 기본 모델에 비해 소폭 상승했음
  - **로지스틱 회귀모델은 하이퍼파라미터의 최적화 보다는 데이터 전처리에 영향을 많이 받음**

Logistic Regression

▶2. 피처 중요도 평가 및 이상치 제거

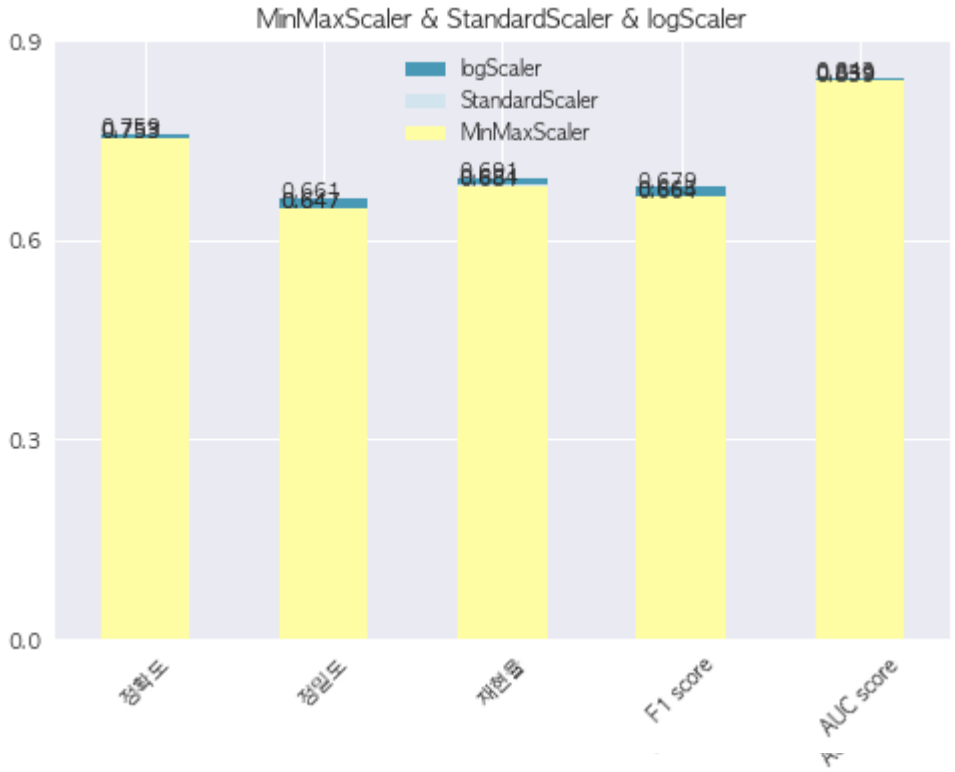


- **상관분석을 이용, 중요도 평가**
  - 낮은 순위의 피처들을 0개~10개까지 삭제 후 모델링 진행
  - 삭제 컬럼수가 늘어날수록 재현율은 향상되지만 정확도가 낮아지는 경향을 보임
  - **2개의 컬럼을 삭제했을 때 정확도, 정밀도, 재현율이 가장 균형있는 모습을 보임**
- **이상치 제거**
  - 높은 순위의 컬럼에 순차적으로 이상치 제거 적용
  - 이상치 제거를 적용한 컬럼수가 늘어날수록 정확도가 개선됨
  - 중요도 상위 1개 컬럼의 이상치를 제거한 모델을 선택

	정확도	정밀도	재현율	F1 score	AUC
2. 최적 하이퍼 파라미터	0.7488	0.6486	0.6941	0.6706	0.8338
4. 피처 중요도 평가후 컬럼제거	0.7491	0.6487	0.6953	0.6712	0.8338
6. 컬럼제거 후 이상치 제거함수 적용	0.7558	0.6519	0.6810	0.6661	0.8412

Logistic Regression

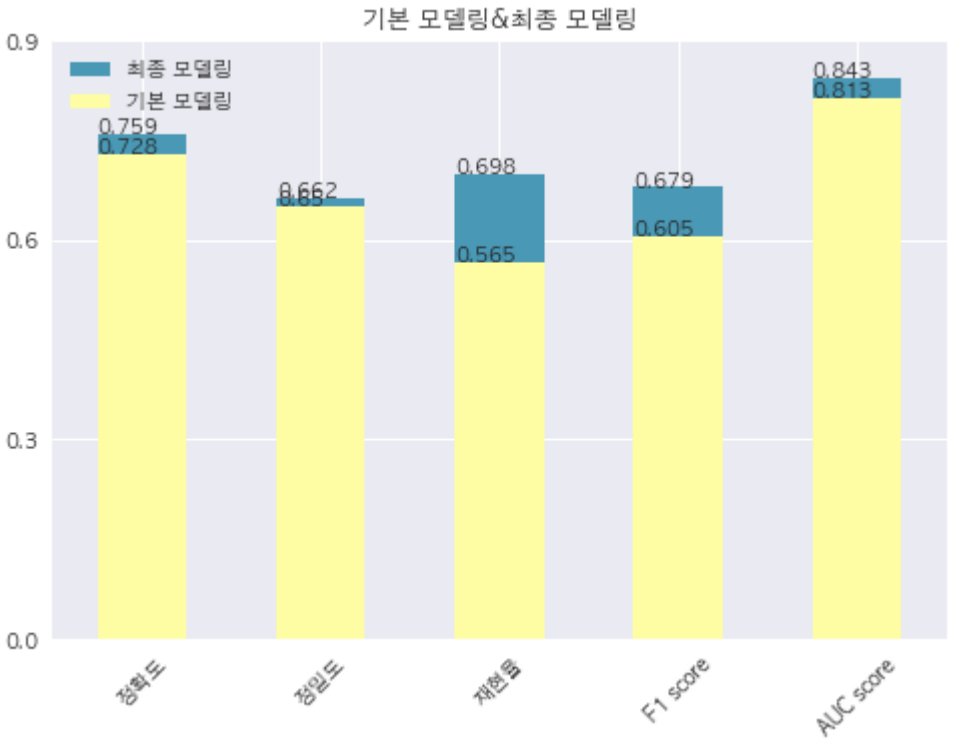
▶ 3. 스케일러 변경



- **스케일러 변경**
  - 연속형 변수인 피처에 정규화 함수를 달리 적용함
  - Standard Scaler, Min Max Scaler, log Scaler를 각각 적용함
  - 중요도 상위 1개 컬럼에 이상치 제거
  - 중요도 하위 2개 컬럼 삭제
  - log Scaler를 적용했을때 평가지표의 가장 큰 개선을 보임
  - log Scaler를 사용한 모델을 최종 모델로 선택함

	정확도	정밀도	재현율	F1 score	AUC score
7-1 MinMaxScaler()	0.7529	0.6468	0.6813	0.6636	0.8390
7-2 StandardScaler()	0.7531	0.6470	0.6835	0.6648	0.8388
7-3 log Scaler	0.7586	0.6614	0.6914	0.6789	0.8429

Logistic Regression



	정확도	정밀도	재현율	F1 score	AUC score
기본 모델링	0.727700	0.649900	0.565200	0.604600	0.812600
최적 하이퍼파라미터	0.748800	0.648600	0.694100	0.670600	0.833800
oversampling	0.728300	0.535300	0.893500	0.669500	0.836400
중요도 평가후 컬럼제거	0.749100	0.648700	0.695300	0.671200	0.833800
이상치 제거함수 순차 적용	0.755300	0.621200	0.680500	0.665500	0.841000
컬럼제거&이상치제거함수 적용	0.755800	0.651900	0.681000	0.666100	0.841200
MinMaxScaler	0.752900	0.646800	0.681300	0.663600	0.839000
StandardScaler	0.753100	0.647000	0.683500	0.664800	0.838800
logScaler	0.758600	0.661400	0.691400	0.678900	0.842900
최종 모델링	0.758800	0.661600	0.697700	0.679200	0.842900

- 중요도 상위 1개 컬럼에 이상치 제거
- 중요도 하위 2개 컬럼 삭제
- log Scale 적용
- 최적 하이퍼파라미터 적용

Support Vector Machine

CONTENTS

Figure out with body signal of smoking

Support Vector Machine

▶서포트 벡터 머신 소개

Support Vector Machine

- 결정경계 또는 초평면을 정의하는 모델
- 새로운 점이 나타나면 경계의 어느쪽에 속하는지 분류를 수행
- sklearn 라이브러리의 SVM() 함수를 사용
  - Kenal : 주어진 데이터를 공간상에 분류하는 함수를 지정
  - C : 서포트 벡터와 결정경계의 마진을 설정
  - gamma : 비선형 커널에서 결정경계의 곡률을 설정
    - C와 gamma를 높게 설정할수록 오류허용이 줄어듦
    - 과적합의 가능성이 있어 하이퍼파라미터 조정에 유의

장점

- 회귀, 분류에 모두 적용 가능
- 라벨을 직접 추정해 조건부 확률 예측모형 중 예측력이 높음
- 과적합의 가능성이 낮음
- 이상치의 영향력이 적음

단점

- 데이터 스케일링에 민감
- 고차원으로 갈수록 학습속도가 느림
- 하이퍼파라미터 조합의 경우의 수가 많음
- 모형이 복잡해 결과설명이 어려움

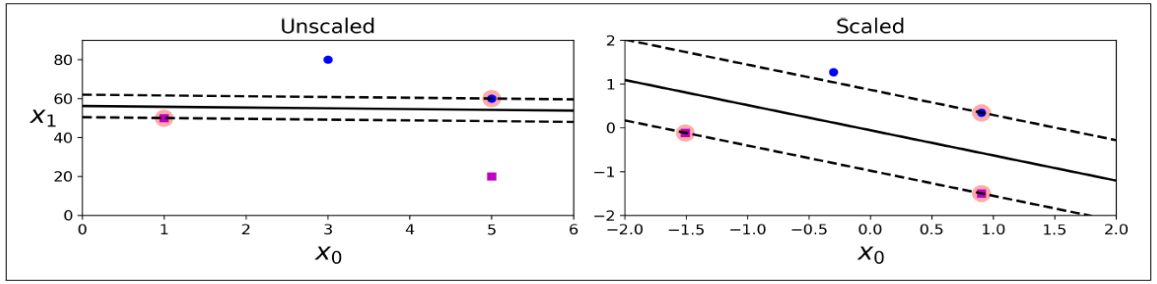


Figure 5-2. Sensitivity to feature scales

- SVM은 위 그림과 같이 스케일을 적용했을 때
- feature의 인스턴스간 마진이 커져 결정경계가 형성됨
- 스케일에 민감도가 높기 때문에 스케일러 조정에 유의를 요구함

Support Vector Machine

▶2. 기본 모델링

	정확도	정밀도	재현율	F1 Score	ROC AUC
Base model	0.736	0.675	0.545	0.603	0.696
Standard Scaling	**0.781**	**0.684**	**0.749**	**0.715**	**0.774**
Min-Max Scaling	0.742	0.622	0.758	0.683	0.745
log Scale	0.751	0.635	0.761	0.692	0.754

모델별 비교

- 하이퍼 파라미터(디폴트)
  - C : 1
  - gamma : scale
  - Kernel : RBF

- Base Model
  - 불필요한 피처만 제거
  - Scale하지 않은 데이터로 모델링
- Scaling
  - Standard
  - Min-Max Scaling
  - log Scale

Support Vector Machine

▶3. 하이퍼파라미터 탐색 및 모델링

S — **Standard Scaling**
C : 3
gamma : auto
Kernal : RBF

M — **Min-Max Scaling**
C : 50
gamma : scale
Kernal : RBF

L — **Log scaling**
C : 50
gamma : auto
Kernal : RBF

모델 별 Train data 평가지표 비교

	Standard Scaling	Min-Max Scaling	log Scale
정확도	**0.794**	0.764	0.763
정밀도	**0.703**	0.658	0.623
재현율	**0.759**	0.744	0.758
F1 Score	**0.73**	0.699	0.701
ROC AUC	**0.786**	0.701	0.762

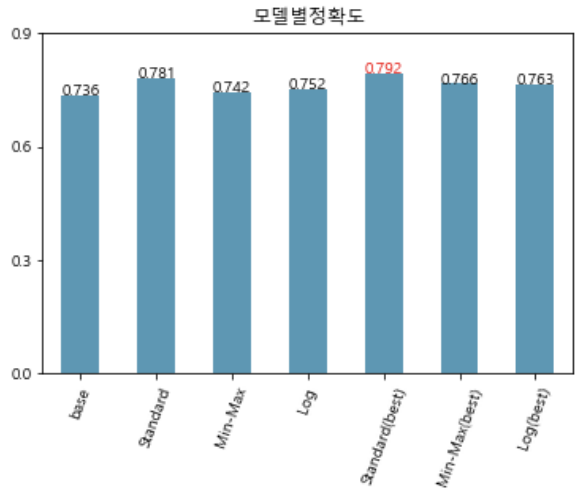
모델 별 Test data 평가지표 비교

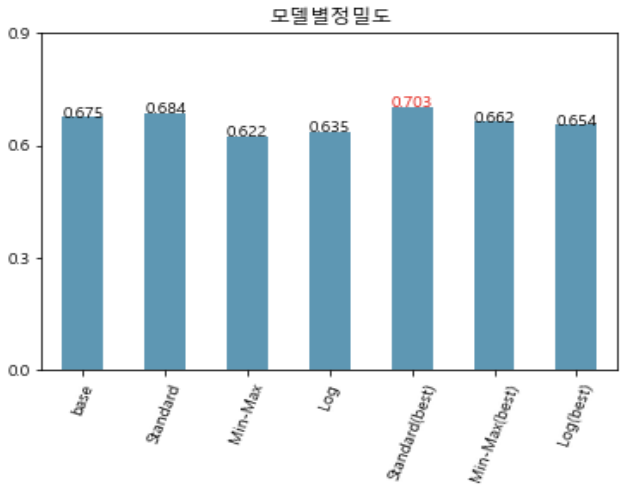
	Standard Scaling	Min-Max Scaling	log Scale
정확도	**0.792**	0.766	0.763
정밀도	**0.703**	0.662	0.654
재현율	**0.756**	0.743	0.757
F1 Score	**0.728**	0.701	0.701
ROC AUC	**0.785**	0.761	0.762

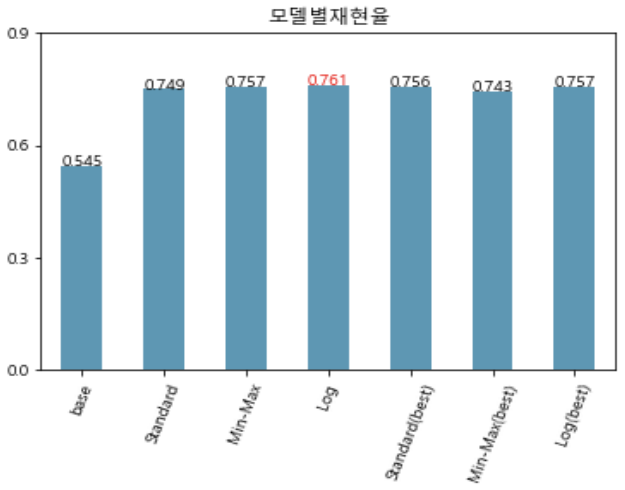
Standard Scaling은 다른 스케일링에 비해 소프트마진으로도 좋은 성능을 냈으므로 이 데이터는 **Standard scaling모델의 결정경계 분류 성능이 가장 좋은 것으로 보임**

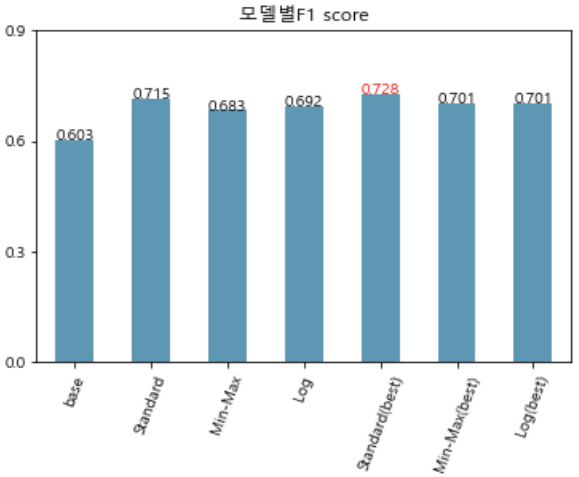
Support Vector Machine

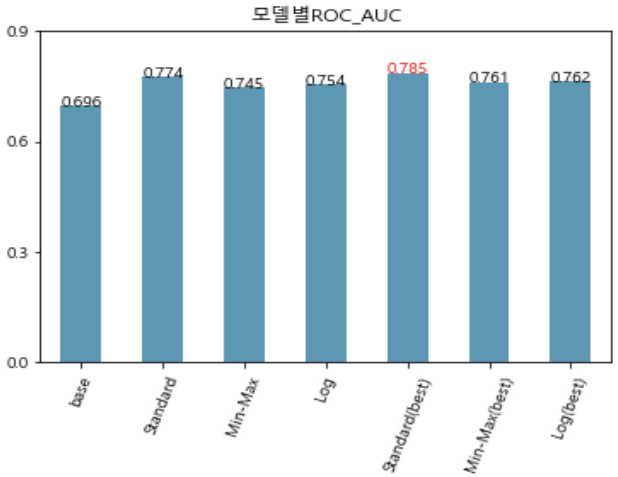
▶4. 시각화 및 최종모델 선정







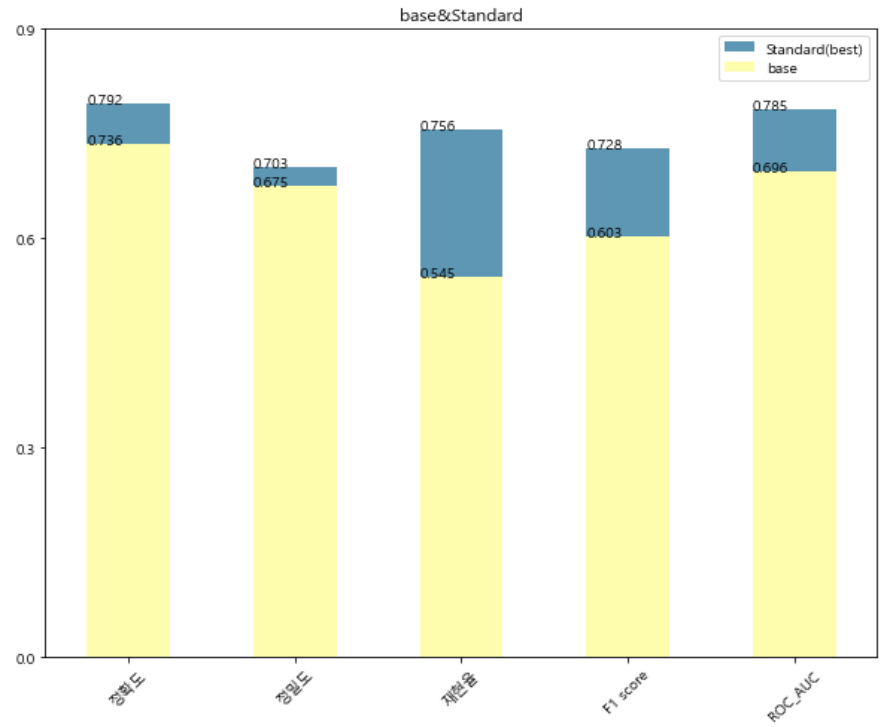




- 정확도, 정밀도, F1 Score, AUC score는 Standard 최적 하이퍼 파라미터로 예측한 모델이 가장 높음
- 재현율만 Log 디폴트 하이퍼 파라미터로 예측한 모델이 높았음을 알 수 있다

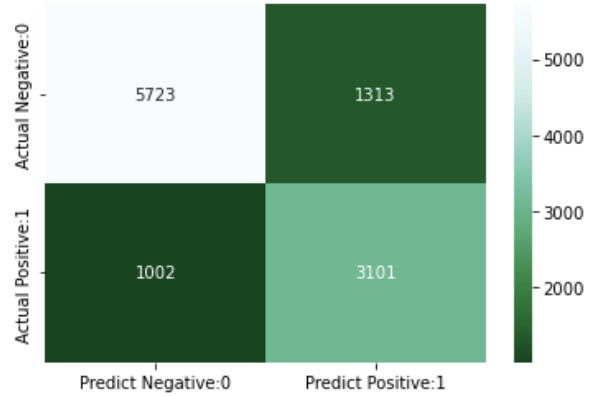
Support Vector Machine

▶4. 시각화 및 최종모델 선정



	기본 모델	최종모델
정확도	73.6%	79.2%(5.6%↑)
정밀도	67.5%	70.3%(2.8%↑)
재현율	54.5%	75.6%(21.1%↑)
F1 Score	60.3%	72.8%(12.5%↑)
ROC AUC	69.6%	78.5%(8.9%↑)

- 최종모델 : Standard Scaling- 최적 파라미터
- 모든 평가지표가 상승
- 재현율이 가장 많이 상승함



Confusion Matrix(Test data) 결과

$$정확도=\frac{5723+3101}{5723+1313+1002+3101}=79.2\%$$

$$정밀도=\frac{3101}{1313+3101}=70.3\%$$

$$재현율=\frac{3101}{1002+3101}=75.6\%$$

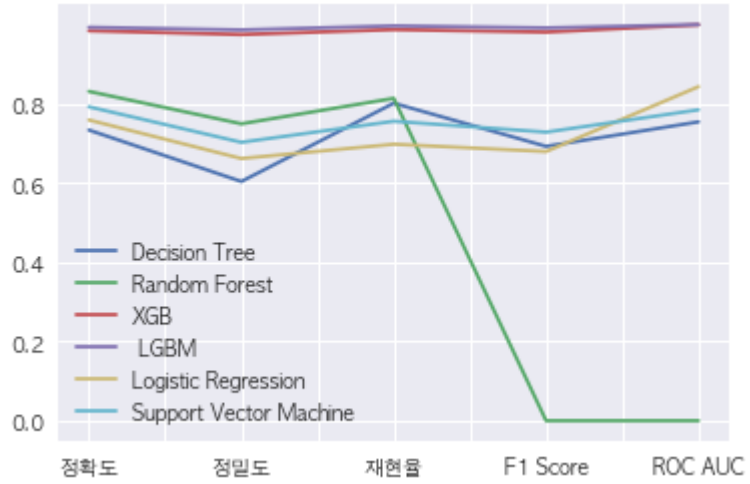
4. 최종 모델 선정

Figure out with body signal of smoking

최종 모델 선정

▶ 최종 모델 선정 - 모델별 최종 성능 지표

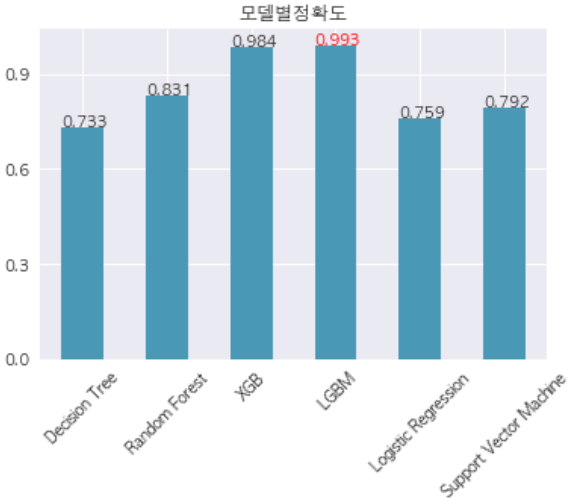
	정확도	정밀도	재현율	F1 Score	ROC AUC
Decision Tree	0.733400	0.603600	0.801200	0.692300	0.754100
Random Forest	0.831000	0.749200	0.813500	0.000000	0.000000
XGB	0.983900	0.974400	0.986800	0.980500	0.998900
LGBM	0.992700	0.986100	0.996300	0.991200	0.999600
Logistic Regression	0.758800	0.661600	0.697700	0.679200	0.842900
Support Vector Machine	0.792100	0.702500	0.755700	0.728100	0.784500

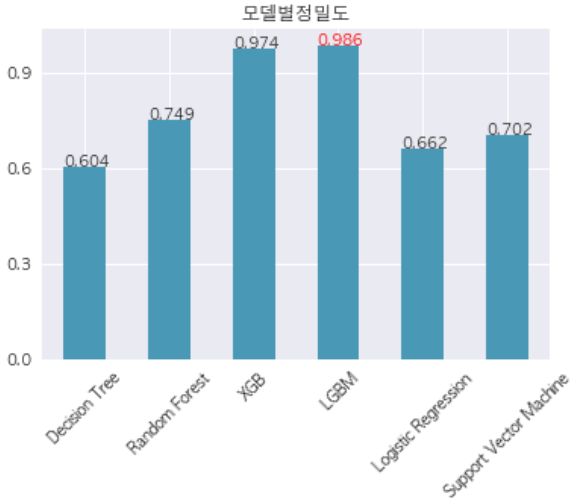


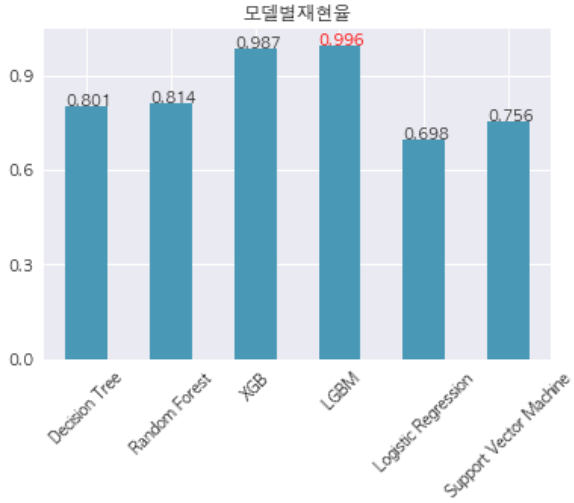
- XGB, LGBM이 지표상 가장 높은 수치를 보임
- Random Forest, SVM, Logistic, Decision Tree는 비교적 낮은 성능을 보임

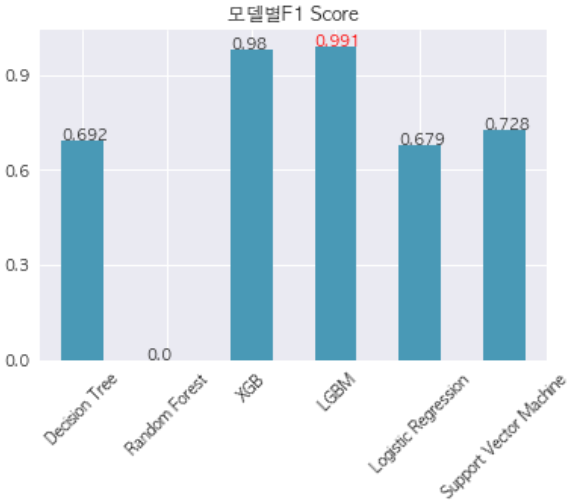
최종 모델 선정

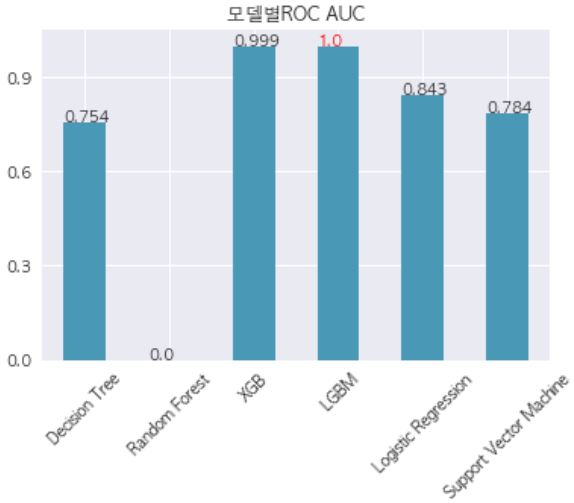
▶ 최종 모델 선정 - 모델별 최종 성능 지표









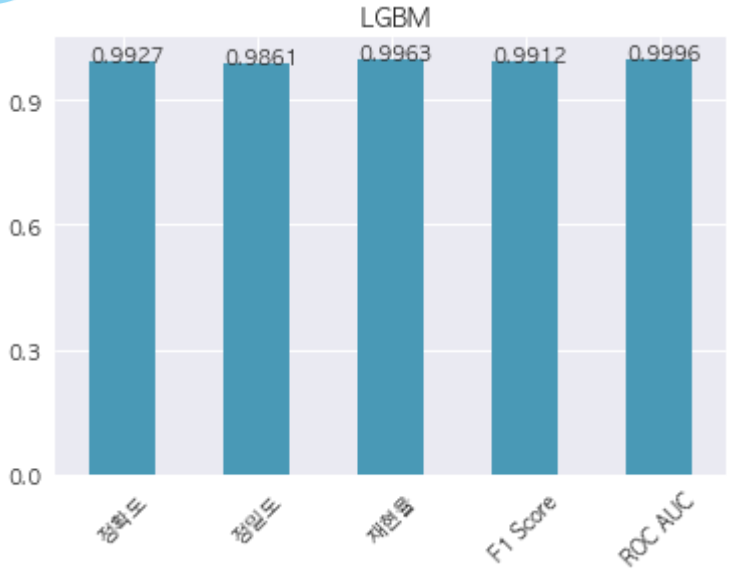


모델별 성능 지표 시각화

- 정확도 : LGBM (0.9927)
- 정밀도 : LGBM (0.9861)
- 재현율 : LGBM (0.9963)
- f1_score : LGBM (0.9912)
- auc_score : LGBM (0.9996)

최종 모델 선정

최종 모델 선정 - LGBM

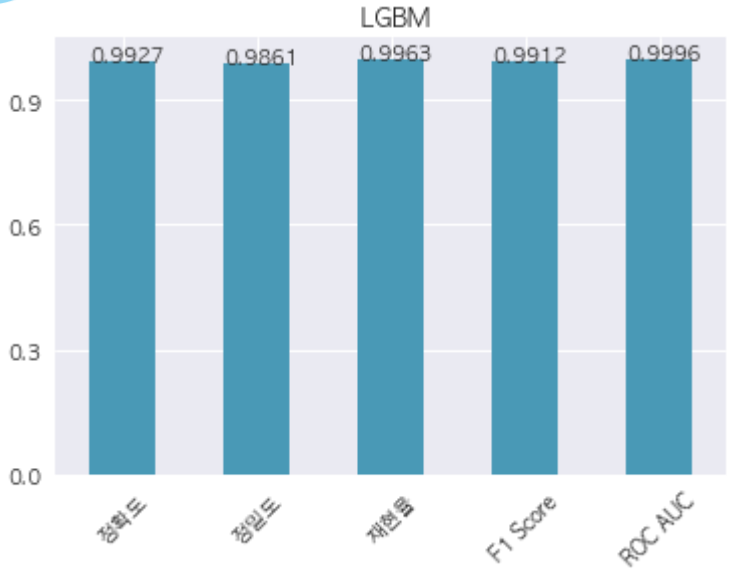


- **최적 파라미터**
  - num_iterations = 1000, max_depth = 12
  - min_child_samples = 5, min_child_weight = 5
  - subsample = 0.8, colsample_bytree = 0.8

- **Decision Tree**
  - 기본적인 모델이지만 과적합 가능성이 높음
  - 수정한 모델 역시 다른 모델에 비해 성능이 좋지 못함
- **Random Forest**
  - 최초 기본모델링시 모든 지표에서 과적합이 발생,
  - 이후 조정한 파라미터에서 과적합은 해결된 것으로 보이지만
  - LGBM보다 상대적으로 지표가 낮음
- **Logistic Regression**
  - 기본모델, 개선모델에서도 타 모델에 비해 성능이 좋지 못함
  - 사용된 데이터와 싱크로율이 좋지 못한것으로 추정됨
- **Support Vector Machine**
  - 다른 모델에 비해 성능지표가 좋지 못함
  - 모델링 시간에 상당한 시간이 소요되어 최종 모델로는 부적합

최종 모델 선정

최종 모델 선정 - LGBM



- **최적 파라미터**
  - num_iterations = 1000, max_depth = 12
  - min_child_samples = 5, min_child_weight = 5
  - subsample = 0.8, colsample_bytree = 0.8

- XGB와 LGBM 두 모델 모두 높은 성능을 보임
- LGBM이 XGB와 비교해 학습시간이 빠르며 미세한 성능의 우위를 보임

- 생체 지표를 이용한 모델링에선 재현율의 중요성이 높음
- LGBM은 높은 재현율과 더불어 균형잡힌 성능을 보여줌
- 헤모글로빈, 중성지방, HDL, Gtp등 호흡과 간에 관련된 지표들은 앞서 살펴본 상관성 분석에서 흡연과 유의미한 상관도를 보임
- LGBM의 피처 중요도 평가에서도 위의 피처들이 상위에 위치함

감사합니다
그만피시조

Figure out with body signal of smoking