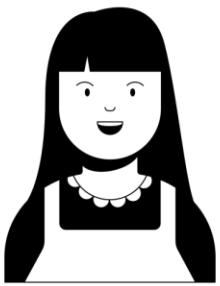


교육과정 현황 데이터를 활용한

구독 해지 예측 및 구독 유형 추천 모델 구현

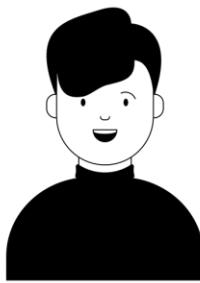
• • • ?
4조 이게돼 조

TEAM



김민주

- 팀장 [PM]
- PPT 제작
- XGBoost



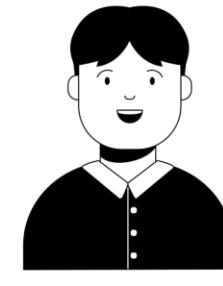
이지호

- 팀원
- 데이터 분석 종합
- LightGBM



전원영

- 팀원
- 데이터 전처리
- DT



한동현

- 팀원
- PPT 제작, 발표
- K-NN

CONTENTS

I

프로젝트개요

- 1. 주제 배경
- 2. 프로젝트 의의
- 3. 진행 일정
- 4. 활용 도구

II

데이터수집·처리

- 1. 데이터 수집
- 2. 데이터 확인
- 3. 데이터 구분
- 4. 데이터 인코딩

III

데이터분석

- 1. EDA
- 2. 데이터 전처리
- 3. 모델링
- 4. 모델링 평가

IV

결론

- 1. [프로젝트 개선점](#)
- 2. [프로젝트 평가](#)
- 3. [인용 · 출처](#)

CONTENTS

I

프로젝트개요

- 1. 주제 배경
- 2. 프로젝트 의의
- 3. 진행 일정
- 4. 활용 도구

II

데이터수집·처리

- 1. 데이터 수집
- 2. 데이터 확인
- 3. 데이터 구분
- 4. 데이터 인코딩

III

데이터분석

- 1. EDA
- 2. 데이터 전처리
- 3. 모델링
- 4. 모델링 평가

IV

결론

- 1. 프로젝트 개선점
- 2. 프로젝트 평가
- 3. 인용 · 출처

1

주제 선정 배경

팀원 선호도 고려

1. 교통데이터를 활용한 교통 취약지 예측 모델 구현
2. 시간단위별 유동인구 데이터 활용한 지역별 매출 예측
3. 해외통신사 데이터를 활용한 고객 성향 분석
4. 교육 과정 현황 데이터를 활용한 교육 추천서비스 모델 구현
5. 국가통계포털 데이터를 활용한 아파트 전세가격 예측

데이터셋 템색 후선정

1. 교통데이터를 활용한 교통 취약지 예측 모델 구현
2. 시간단위별 유동인구 데이터 활용한 지역별 매출 예측
3. 해외통신사 데이터를 활용한 고객 성향 분석
4. 교육 과정 현황 데이터를 활용한 교육 추천서비스 모델 구현
5. 국가통계포털 데이터를 활용한 아파트 전세가격 예측



최종주제 선정

'교육 과정 현황 데이터를 활용한 구독 해지 예측 및 구독 유형 추천 모델 구현'

2 프로젝트 의의

디지털 플랫폼 규모, 2025

60 trillion
(7경 2000조 원)

\$

30 percent
(글로벌 전체 기업 매출)



애드테크 규모, 2025

9조 9833억 원

7조 3250억 원

36.29% ▲

2021년 2025년

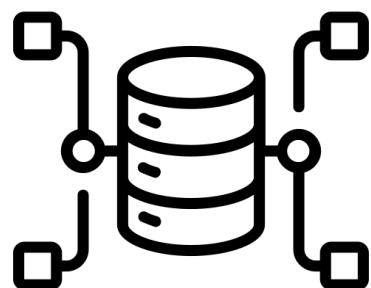
2 프로젝트 의의

에듀테크 시장에서의 프로젝트 의의

Platform



Data

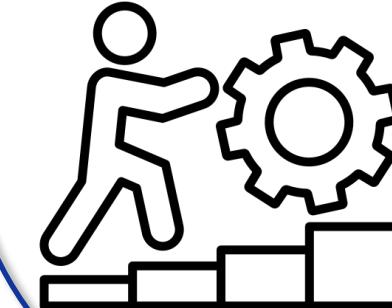


User



⋮

Improve



3 진행 일정

수정예정
내일 빠르게 고치겠습니다

23.12.12 ~ 13

프로젝트 주제 선정

- 4번주제선정

워크스페이스 구축

- 협업관련채널구축

23.12.14

데이터 수집 · 처리

- 수집한데이터확인
- 데이터 전처리

23.12.15 ~ 26

EDA

- 컬럼간상관성,이상치확인

데이터 분석 모델링

- 알고리즘별 모델링
- DT,KNN,XGBoost,LightGBM 모델링후,평가

23.12.27 ~ 28

데이터 분석 종합

- 분석내용취합

23.12.29 ~ 24.01.02

발표 자료 작성

- 분석내용PPT작성
- 스크립트작성

24.01.03

프로젝트 발표

3 진행 일정

수정예정
내일 빠르게 고치겠습니다

23.12.12 ~ 13

23.12.14

23.12.15 ~ 26

프로젝트 주제 선정

- 4번주제선정

워크스페이스 구축

- 협업관련채널구축

데이터 수집 · 처리

- 수집한데이터확인
- 데이터 전처리

EDA

- 컬럼간상관성,이상치확인

데이터 분석 모델링

- 알고리즘별 모델링
- DT,KNN,XGBoost,LightGBM 모델링후,평가

23.12.27 ~ 28

23.12.29 ~ 24.01.02

24.01.03

데이터 분석 종합

- 분석내용취합

발표 자료 작성

- 분석내용PPT작성
- 스크립트작성

프로젝트 발표

3 진행 일정

수정예정
내일 빠르게 고치겠습니다

23.12.12 ~ 13

23.12.14

23.12.15 ~ 26

프로젝트 주제 선정

- 4번주제선정

워크스페이스 구축

- 협업관련채널구축

데이터 수집 · 처리

- 수집한데이터확인
- 데이터 전처리

EDA

- 컬럼간상관성,이상치확인

데이터 분석 모델링

- 알고리즘별 모델링
- DT,KNN,XGBoost,LightGBM 모델링후,평가

23.12.27 ~ 28

23.12.29 ~ 24.01.02

24.01.03

데이터 분석 종합

- 분석내용취합

발표 자료 작성

- 분석내용PPT작성
- 스크립트작성

프로젝트 발표

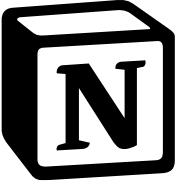
4 활용 도구

Discord



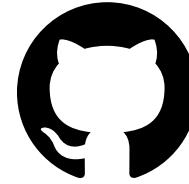
- 프로젝트 관련 회의
- 공지사항 전달

Notion



- 프로젝트 일정 관리
- 레퍼런스 아카이빙

GitHub



- 파일 버전 관리
- 작성 코드 공유

Office 365



- PPT 동시 작성
- 작성 PPT 동기화

CONTENTS

I

프로젝트개요

- 1. 주제 배경
- 2. 프로젝트 의의
- 3. 진행 일정
- 4. 활용 도구

II

데이터수집·처리

- 1. 데이터 수집
- 2. 데이터 확인
- 3. 데이터 구분
- 4. 데이터 인코딩

III

데이터분석

- 1. EDA
- 2. 데이터 전처리
- 3. 모델링
- 4. 모델링 평가

IV

결론

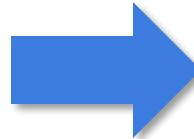
- 1. 프로젝트 개선점
- 2. 프로젝트 평가
- 3. 인용 · 출처

1

데이터 수집

DACON

해커톤 37회
학습 플랫폼 이용자 구독 강신 예측



학습 플랫폼 이용자 데이터



train.csv



test.csv

...

2 데이터 확인

필요한 모듈 import

```
import pandas as pd  
import numpy as np
```

Pandas로 데이터 로드

```
train_origin = pd.read_csv('train.csv')  
test_origin = pd.read_csv('test.csv')
```

로드 데이터 복사

```
train = train_origin.copy()  
test = test_origin.copy()
```

로드 데이터 info로 확인

```
train.info()  
test.info()
```

2

데이터 확인 :info

train.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   user_id          10000 non-null   object  
 1   subscription_duration  10000 non-null   int64  
 2   recent_login_time    10000 non-null   int64  
 3   average_login_time   10000 non-null   float64 
 4   average_time_per_learning_session 10000 non-null   float64 
 5   monthly_active_learning_days   10000 non-null   int64  
 6   total_completed_courses    10000 non-null   int64  
 7   recent_learning_achievement 10000 non-null   float64 
 8   abandoned_learning_sessions 10000 non-null   int64  
 9   community_engagement_level 10000 non-null   int64  
 10  preferred_difficulty_level 10000 non-null   object  
 11  subscription_type       10000 non-null   object  
 12  customer_inquiry_history 10000 non-null   int64  
 13  payment_pattern        10000 non-null   int64  
 14  target               10000 non-null   int64  
dtypes: float64(3), int64(9), object(3)
memory usage: 1.1+ MB
```

test.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   user_id          10000 non-null   object  
 1   subscription_duration  10000 non-null   int64  
 2   recent_login_time    10000 non-null   int64  
 3   average_login_time   10000 non-null   float64 
 4   average_time_per_learning_session 10000 non-null   float64 
 5   monthly_active_learning_days   10000 non-null   int64  
 6   total_completed_courses    10000 non-null   int64  
 7   recent_learning_achievement 10000 non-null   float64 
 8   abandoned_learning_sessions 10000 non-null   int64  
 9   community_engagement_level 10000 non-null   int64  
 10  preferred_difficulty_level 10000 non-null   object  
 11  subscription_type       10000 non-null   object  
 12  customer_inquiry_history 10000 non-null   int64  
 13  payment_pattern        10000 non-null   int64  
dtypes: float64(3), int64(8), object(3)
memory usage: 1.1+ MB
```

2

데이터 확인 : nunique

고유값 확인

```
train[train.columns].nunique()  
test[test.columns].nunique()
```

train[train.columns].unique()

| | |
|-----------------------------------|-------|
| user_id | 10000 |
| subscription_duration | 23 |
| recent_login_time | 29 |
| average_login_time | 10000 |
| average_time_per_learning_session | 10000 |
| monthly_active_learning_days | 24 |
| total_completed_courses | 27 |
| recent_learning_achievement | 10000 |
| abandoned_learning_sessions | 13 |
| community_engagement_level | 5 |
| preferred_difficulty_level | 3 |
| subscription_type | 2 |
| customer_inquiry_history | 10 |
| payment_pattern | 8 |
| target | 2 |
| dtype: int64 | |

test[test.columns].unique()

| | |
|-----------------------------------|-------|
| user_id | 10000 |
| subscription_duration | 23 |
| recent_login_time | 29 |
| average_login_time | 10000 |
| average_time_per_learning_session | 10000 |
| monthly_active_learning_days | 24 |
| total_completed_courses | 27 |
| recent_learning_achievement | 10000 |
| abandoned_learning_sessions | 12 |
| community_engagement_level | 5 |
| preferred_difficulty_level | 3 |
| subscription_type | 2 |
| customer_inquiry_history | 11 |
| payment_pattern | 8 |
| dtype: int64 | |

3 데이터 구분:index

user_id 컬럼을 Index로 설정

```
train = train.set_index(['user_id'])  
test = test.set_index(['user_id'])
```



출력

| | subscription_duration | recent_login_time | average_login_time |
|----------|-----------------------|-------------------|--------------------|
| user_id | | | |
| b919c29d | 13 | 14 | 14.946163 |
| a0a60abb | 16 | 18 | 18.453224 |
| b9f171ae | 22 | 1 | 16.195228 |
| 5dc0ba8b | 1 | 19 | 17.628656 |
| 65c83654 | 4 | 5 | 21.390656 |
| ... | ... | ... | ... |



3 데이터 구분: dtype

데이터 타입 구분

```
# 수치형, 범주형 데이터 각각 numerical cols, categorical cols로 구분  
categorical_cols = [col for col in train.columns if train[col].dtype == 'object']  
  
numerical_cols = [col for col in train.columns[:-2]  
                  if train[col].dtype in ['int64', 'float64'] and col != 'payment_pattern']
```

범주형 컬럼

```
['preferred_difficulty_level', 'subscription_type']
```

수치형 컬럼

```
['subscription_duration',  
 'recent_login_time',  
 'average_login_time',  
 'average_time_per_learning_session',  
 'monthly_active_learning_days',  
 'total_completed_courses',  
 'recent_learning_achievement',  
 'abandoned_learning_sessions',  
 'community_engagement_level',  
 'customer_inquiry_history']
```

구분 결과

| 인덱스 | 컬럼명 | 타입 |
|-------|-----------------------------------|--------|
| 23 | user_id | 범주형 컬럼 |
| 23 | subscription_duration | 수치형 컬럼 |
| 29 | recent_login_time | 수치형 컬럼 |
| 10000 | average_login_time | 수치형 컬럼 |
| 10000 | average_time_per_learning_session | 수치형 컬럼 |
| 27 | monthly_active_learning_days | 수치형 컬럼 |
| 27 | total_completed_courses | 수치형 컬럼 |
| 10000 | recent_learning_achievement | 수치형 컬럼 |
| 13 | abandoned_learning_sessions | 수치형 컬럼 |
| 5 | community_engagement_level | 수치형 컬럼 |
| 2 | preferred_difficulty_level | 범주형 컬럼 |
| 2 | subscription_type | 범주형 컬럼 |
| 8 | customer_inquiry_history | 수치형 컬럼 |
| 8 | payment_pattern | 기타 컬럼 |
| 2 | target | 기타 컬럼 |
| 2 | dtype: int64 | 기타 컬럼 |

4 데이터 인코딩: one-hot encoding

원-핫 인코딩

```
from sklearn.preprocessing import OneHotEncoder  
  
# OneHotEncoder 인더스트화  
encoder = OneHotEncoder(sparse_output=False)  
  
# train의 'payment_pattern' 인코딩  
encoded = encoder.fit_transform(train[['payment_pattern']])  
encoded_df = pd.DataFrame(encoded, index=train.index,  
                           columns=encoder.get_feature_names_out(['payment_pattern']))  
  
# 기존의 'payment_pattern' column 삭제  
train = train.drop(columns='payment_pattern')  
train = pd.concat([train, encoded_df], axis=1)  
  
# test의 'payment_pattern' 인코딩  
encoded = encoder.transform(test[['payment_pattern']])  
encoded_df = pd.DataFrame(encoded, index=test.index,  
                           columns=encoder.get_feature_names_out(['payment_pattern']))  
  
# 기존의 'payment_pattern' column 삭제  
test = test.drop(columns='payment_pattern')  
test = pd.concat([test, encoded_df], axis=1)
```

출력

| user_id | payment_pattern_0 | payment_pattern_1 |
|----------|-------------------|-------------------|
| b919c29d | 0.0 | 0.0 |
| a0a60abb | 0.0 | 0.0 |
| b9f171ae | 0.0 | 0.0 |
| 5dc0ba8b | 1.0 | 0.0 |
| 65c83654 | 1.0 | 0.0 |
| ... | ... | ... |

4 데이터 인코딩: label encoding

범주형 데이터 확인

```
# 범주형 데이터 확인  
for col in categorical_cols:  
    display(train[col].value_counts())
```

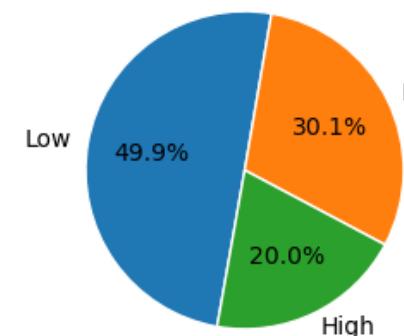
범주형 데이터 시각화

```
# 범주형 데이터 값 비율 확인 시각화  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
for col in categorical_cols:  
    ratio = train[col].value_counts()  
    labels = train[col].unique()  
    wedgeprops={'width': 1, 'edgecolor': 'w', 'linewidth': 1}  
  
    plt.figure(figsize = (3,3))  
    plt.title(col)  
    plt.pie(ratio, labels=labels, autopct='%.1f%%',  
    startangle=260, counterclock=False, wedgeprops=wedgeprops)  
    plt.show()
```

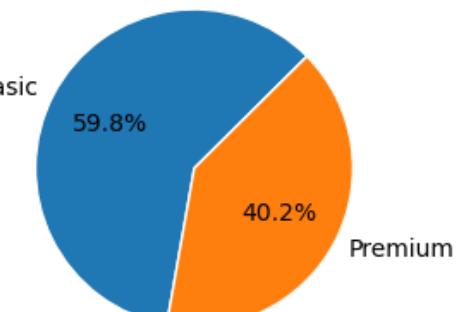
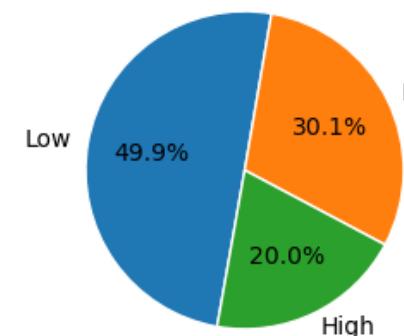
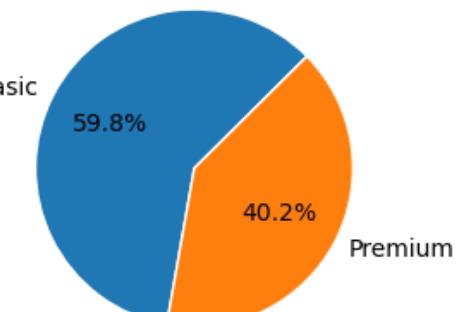


```
preferred_difficulty_level  
Low      4992  
Medium   3008  
High     2000  
Name: count, dtype: int64  
  
subscription_type  
Basic    5979  
Premium  4021  
Name: count, dtype: int64
```

preferred_difficulty_level



subscription_type



4 데이터 인코딩: label encoding

라벨 인코딩

```
# 'preferred_difficulty_level' 인코딩  
train['preferred_difficulty_level'] =  
train['preferred_difficulty_level'].map({'Low':1,'Medium':2,'High':3})  
test['preferred_difficulty_level'] =  
test['preferred_difficulty_level'].map({'Low':1,'Medium':2,'High':3})  
  
# 'subscription_type' 인코딩  
train['subscription_type'] = train['subscription_type'].map({'Basic':0, 'Premium':1})  
test['subscription_type'] = test['subscription_type'].map({'Basic':0, 'Premium':1})  
  
# 확인  
train[categorical_cols].head()
```

출력

preferred_difficulty_level subscription_type

user_id

| | | |
|----------|---|---|
| b919c29d | 1 | 0 |
| a0a60abb | 2 | 0 |
| b9f171ae | 2 | 1 |
| 5dc0ba8b | 1 | 0 |
| 65c83654 | 2 | 0 |

CONTENTS

I

프로젝트개요

- 1. 주제 배경
- 2. 프로젝트 의의
- 3. 진행 일정
- 4. 활용 도구

II

데이터수집·처리

- 1. 데이터 수집
- 2. 데이터 확인
- 3. 데이터 구분
- 4. 데이터 인코딩

III

데이터분석

- 1. EDA
- 2. 데이터 전처리
- 3. 모델링
- 4. 모델링 평가

IV

결론

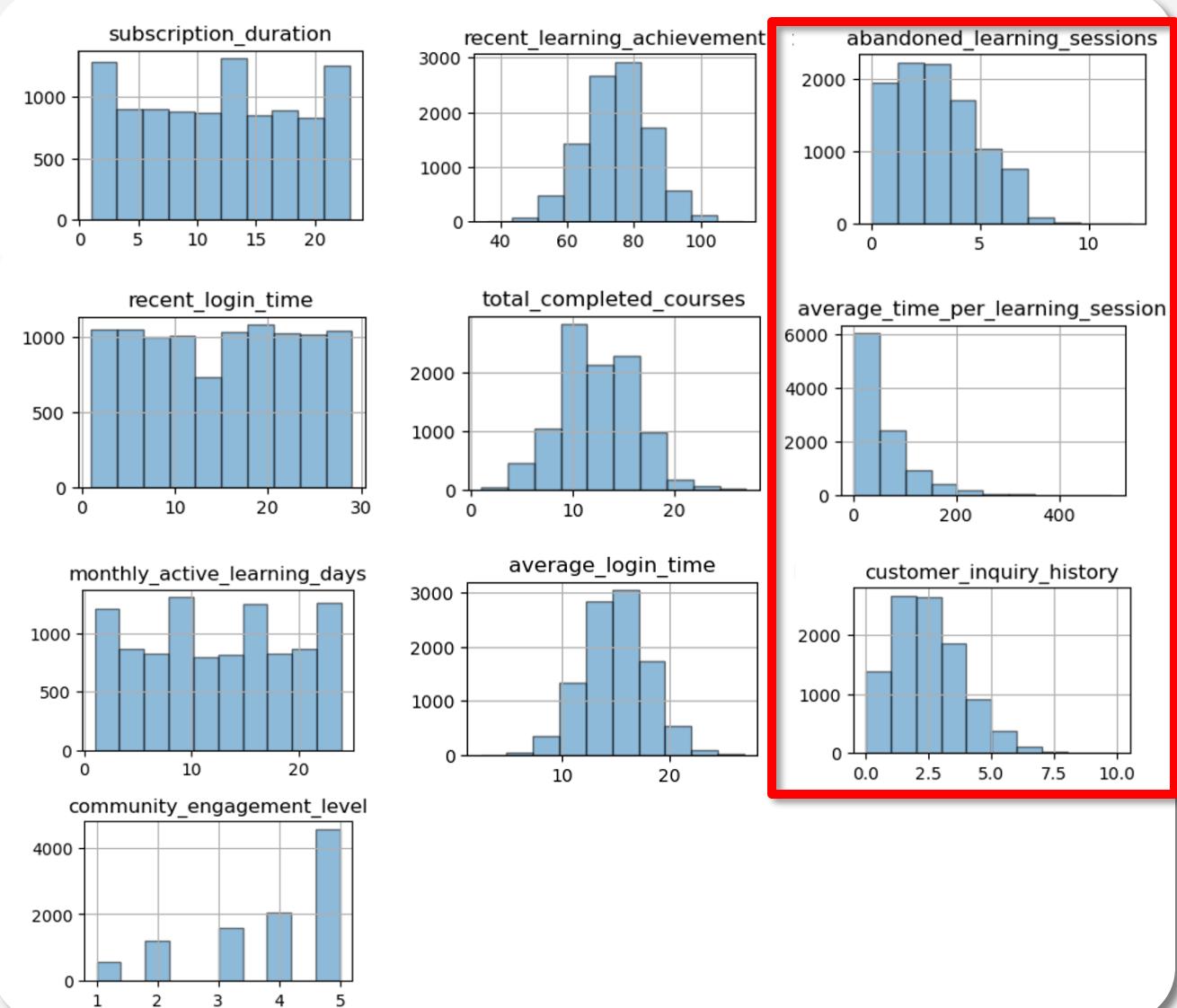
- 1. 프로젝트 개선점
- 2. 프로젝트 평가
- 3. 인용 · 출처

1

EDA: outlier

수치형 시각화

```
train[numerical_cols].hist(alpha=0.5, edgecolor='k', layout=(6,2),  
figsize=(6, 12))  
  
plt.suptitle('Value', fontsize=16)  
plt.xlabel('Value')  
plt.ylabel('frequency')  
  
plt.tight_layout(rect=[0, 0, 1, 1])  
plt.show()
```



1

EDA: correlation

상관성 확인

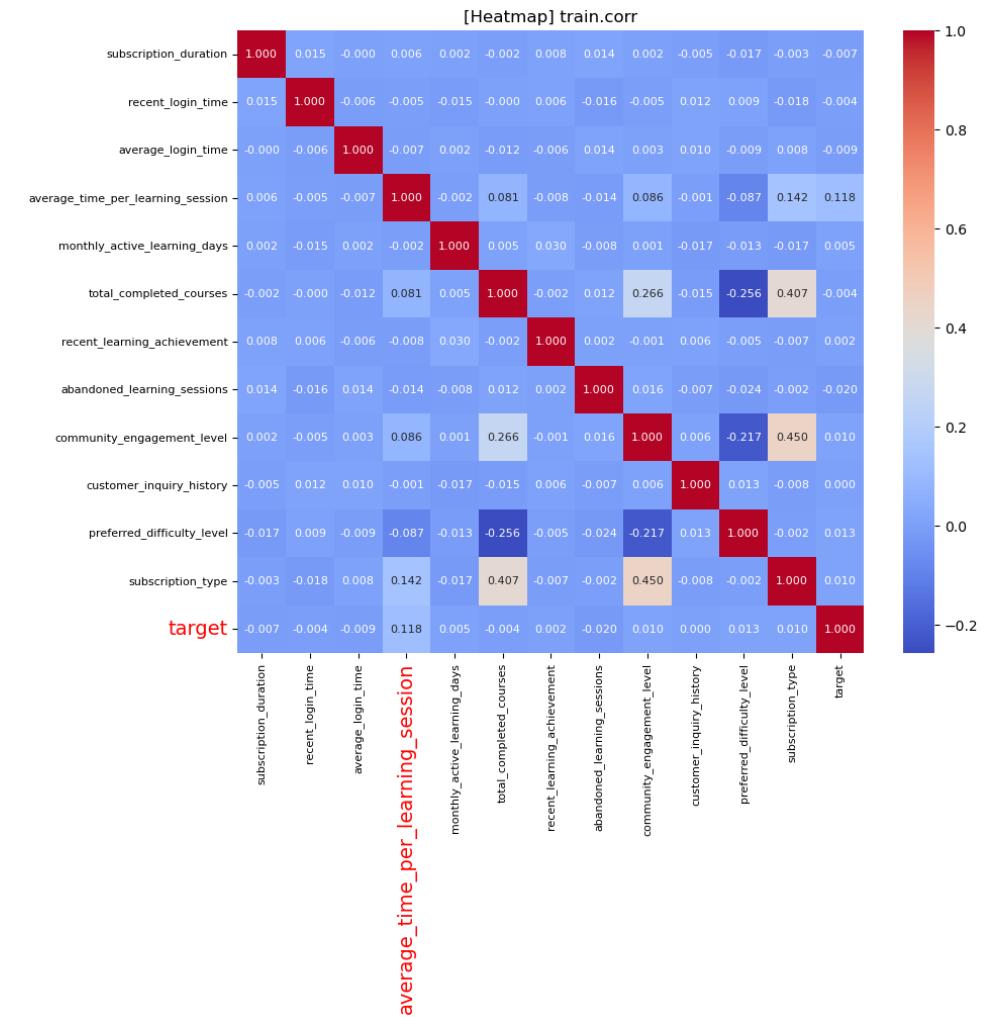
```
# 수치형 컬럼간 상관성 확인을 위해 corr_train에 할당
corr_train = train[numerical_cols + ['target']].corr()

# 상관관계 heatmap
plt.figure(figsize=(10,8))
sns.heatmap(corr_train, annot=True, fmt='.3f', cmap='coolwarm',
            annot_kws={'size': 8})
plt.title('[Heatmap] train_numerical_cols')

# 유의되는 column 표시
plt.tick_params(axis='both', which='both', labelsize=8)
plt.gca().get_xticklabels()[3].set_fontsize(14)
plt.gca().get_xticklabels()[3].set_color('red')
plt.gca().get_yticklabels()[10].set_fontsize(14)
plt.gca().get_yticklabels()[10].set_color('red')

plt.show()
```

출력



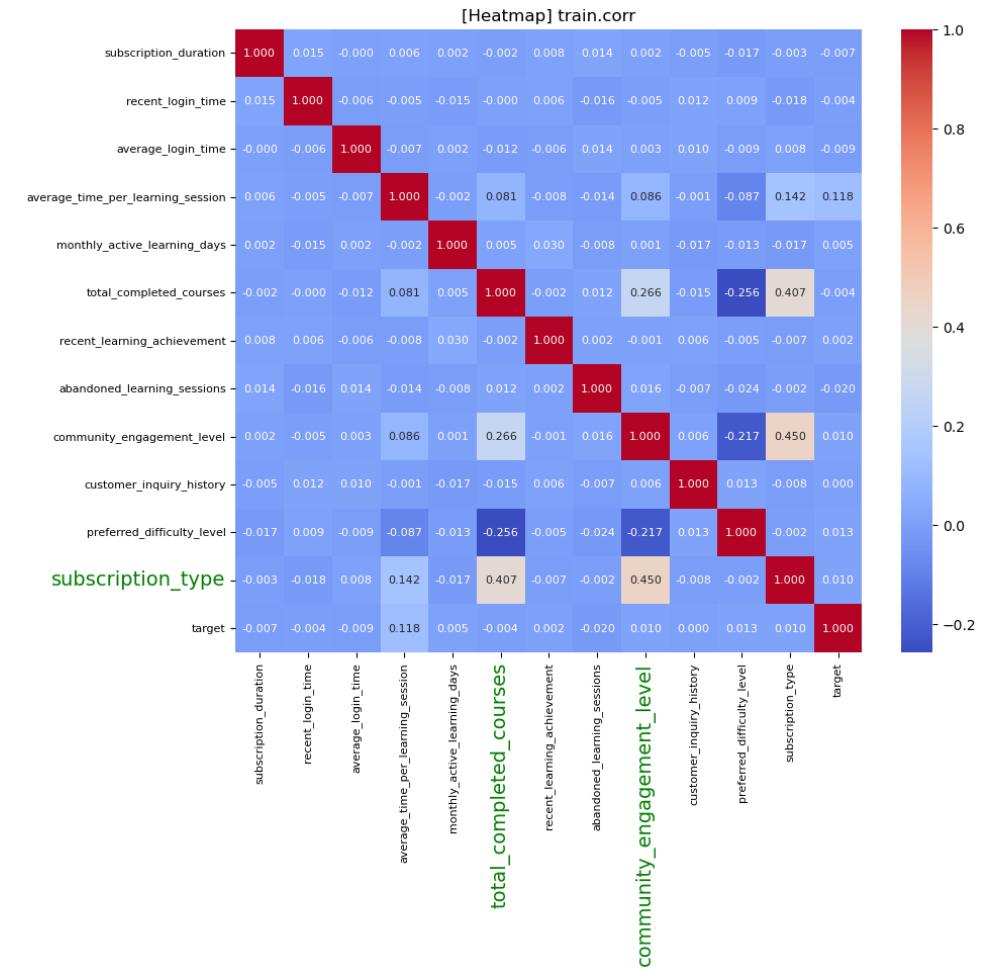
1

EDA: correlation

상관성 확인

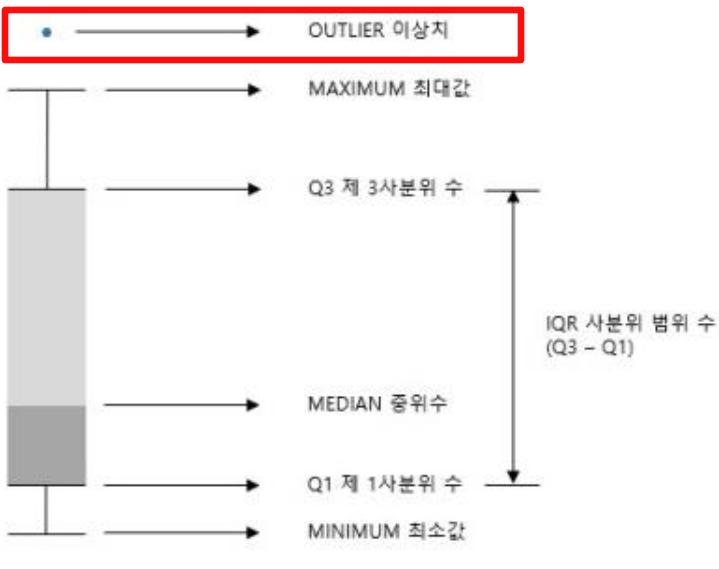
```
# 수치형 컬럼간 상관성 확인을 위해 corr_train에 할당  
corr_train = train[numerical_cols + ['target']].corr()  
  
# 상관관계 heatmap  
plt.figure(figsize=(10,8))  
sns.heatmap(corr_train, annot=True, fmt='.3f', cmap='coolwarm',  
            annot_kws={'size': 8})  
plt.title('[Heatmap] train_numerical_cols')  
  
# 유의되는 column 표시  
plt.tick_params(axis='both', which='both', labelsize=8)  
plt.gca().get_xticklabels()[3].set_fontsize(14)  
plt.gca().get_xticklabels()[3].set_color('red')  
plt.gca().get_yticklabels()[10].set_fontsize(14)  
plt.gca().get_yticklabels()[10].set_color('red')  
  
plt.show()
```

출력



2 데이터 전처리 : outlier

이상치 개념



이상치 제거

```
# 이상치 제거 함수
def replace_outliers(df, col):
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)

    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    df[col] = df[col].apply(lambda x: lower_bound if x < lower_bound else (upper_bound if x > upper_bound else x))
    return df

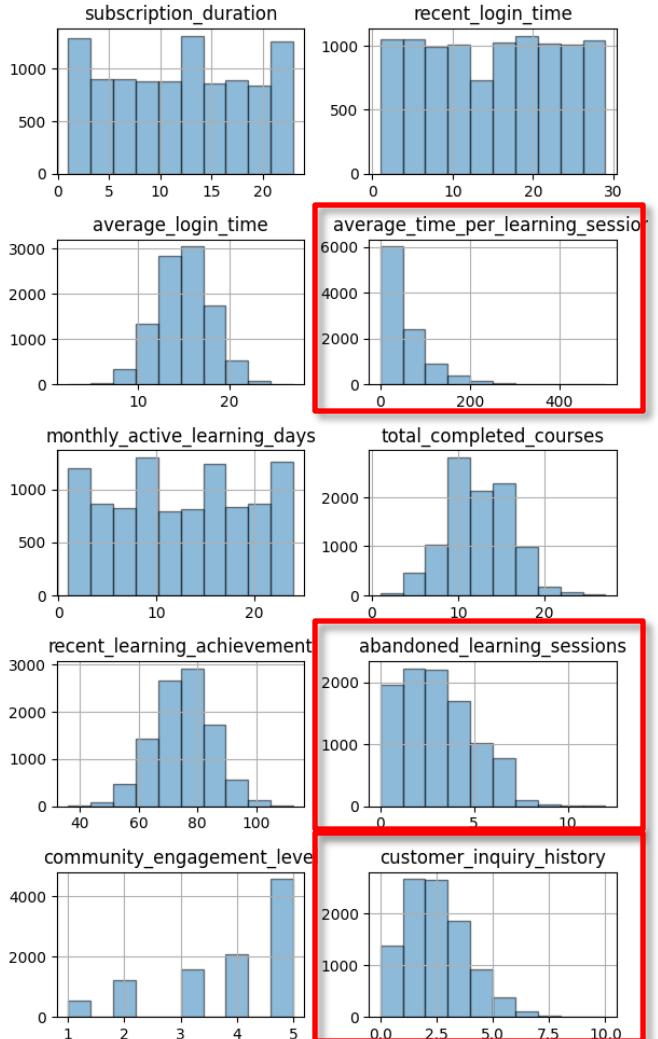
# 이상치를 제거해야하는 columns
outlier_columns = ['average_time_per_learning_session', 'abandoned_learning_sessions',
'customer_inquiry_history']

for col in outlier_columns:
    train = replace_outliers(train, col)
```

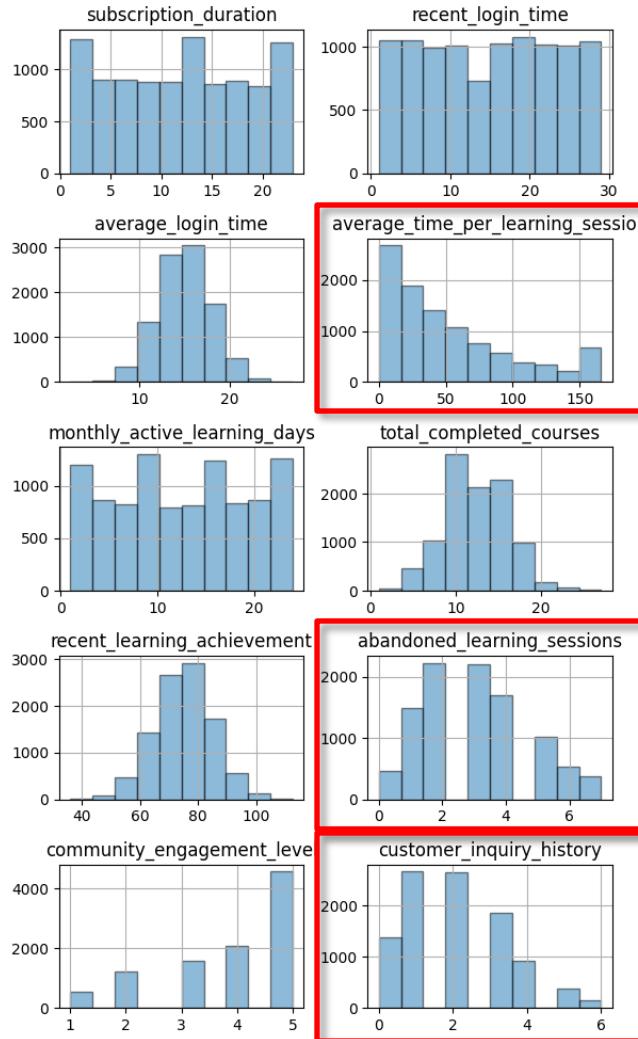
2

데이터 전처리 : outlier

이상치 제거 전



이상치 제거 후



2 데이터 전처리 : scaling

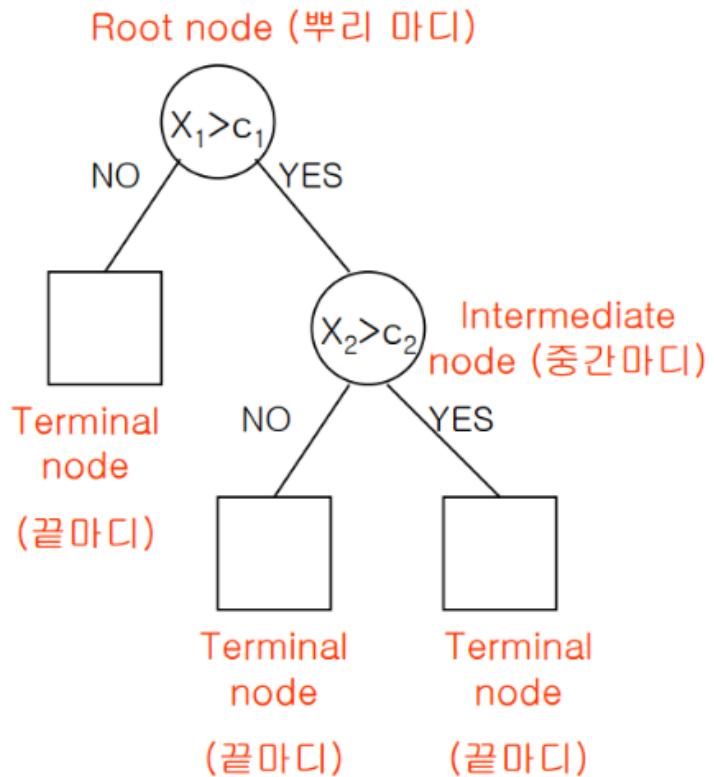
MinMax스케일링

```
from sklearn.preprocessing import MinMaxScaler  
  
scaler = MinMaxScaler()  
for col in numerical_cols:  
    train[col] = scaler.fit_transform(train[col].values.reshape(-1,1))  
    test[col] = scaler.transform(test[col].values.reshape(-1,1))
```



| | subscription_duration | recent_login_time | average_login_time |
|----------|-----------------------|-------------------|--------------------|
| user_id | | | |
| b919c29d | 0.545455 | 0.464286 | 0.510710 |
| a0a60abb | 0.681818 | 0.607143 | 0.653087 |
| b9f171ae | 0.954545 | 0.000000 | 0.561419 |
| 5dc0ba8b | 0.000000 | 0.642857 | 0.619612 |
| 65c83654 | 0.136364 | 0.142857 | 0.772338 |
| ... | ... | ... | ... |

3 모델링: DT



POINT 01

노드와 브랜치를 활용해서 이진 분류 반복하는 알고리즘

POINT 02

Terminal 노드에서 새로운 데이터 확인 후, 가장 높은 빈도 범주로 데이터 분류

POINT 03

직관적이고 시각화 우수, 예측과 성능은 낮은 편

3 모델링: DT

```
from sklearn.tree import DecisionTreeClassifier

def model_dt(x_train, x_val, y_train, y_val):
    model = DecisionTreeClassifier(random_state=12)
    model.fit(x_train, y_train)
    pred = model.predict(x_val)
    accuracy = accuracy_score(y_val, pred)

    return accuracy

# 최적의 파라미터 저장
# DT의 경우 직접 지정한 파라미터들 중 최적값을 저장
best_dt = {'random_state': 12, 'max_depth': 10, 'min_samples_leaf': 4}
Best_dt = {'random_state': 12, 'max_depth': 10, 'min_samples_leaf': 4}
```

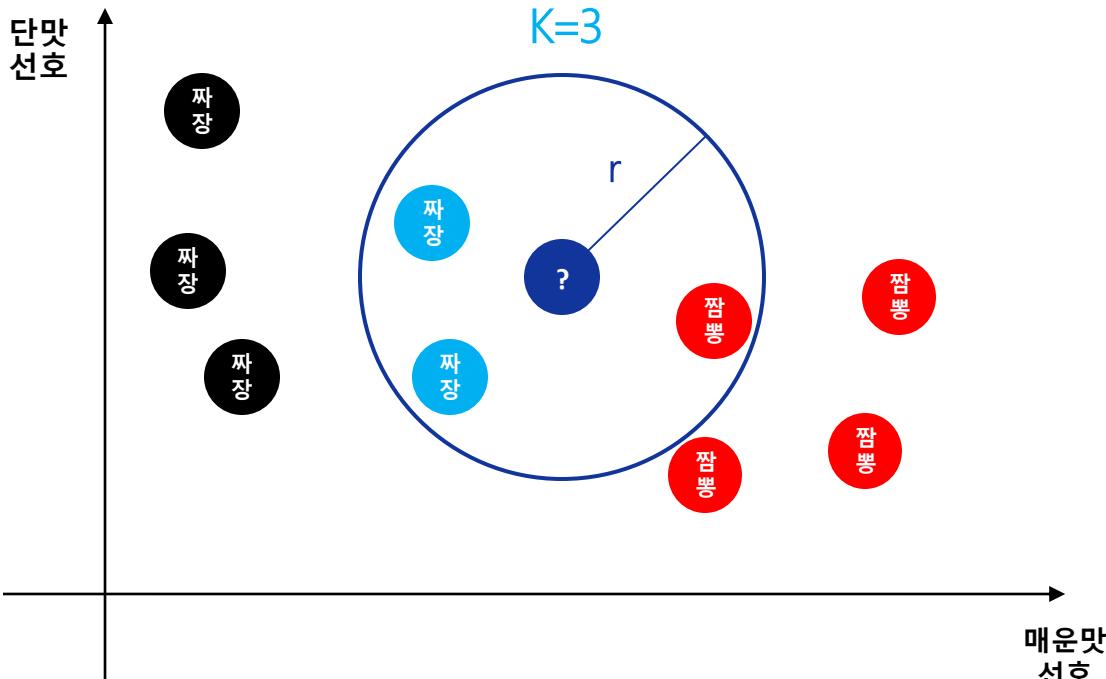
- DT 모델 생성 후, 학습 진행
- 구현한 모든 모델 random_state=12로 지정

- 시각화하여 최적 max_depth와 leaf 조절

검증 코드 실행 결과

```
DT - 구독 해지 예측 최적 파라미터:{'random_state': 12, 'max_depth': 10, 'min_samples_leaf': 4}
DT - 구독 해지 예측 최적 파라미터의 정확도:0.67
DT - 구독 유형 추천 최적 파라미터:{'random_state': 12, 'max_depth': 10, 'min_samples_leaf': 4}
DT - 구독 유형 추천 최적 파라미터의 정확도:0.73
```

3 모델링: KNN



POINT 01

가장 인접한 이웃 K 개의 레이블 참조하여 분류하는 알고리즘

POINT 02

r 거리에 있는 이웃 K 개 레이블을 참조하여 예측

POINT 03

수치형 데이터 다룰 때 정확도 속도 우수 하지만,
거리 기반의 분류로 이상치 처리, 표준화 필요

3 모델링: KNN

```
from sklearn.neighbors import KNeighborsClassifier

score_list = []
def model_knn(trial, x_train, x_val, y_train, y_val):
    param_grid = {
        'n_neighbors': trial.suggest_int('n_neighbors', 1, 50),
        'weights': trial.suggest_categorical('weights', ['uniform', 'distance']),
        'metric': trial.suggest_categorical('metric', ['euclidean', 'manhattan']),
        'p': trial.suggest_int('p', 1, 5)
    }
    model = KNeighborsClassifier(**param_grid)

    fold = StratifiedKFold(n_splits=5, shuffle=True, random_state=12)

    for t, v in fold.split(x, y):
        x_train, x_val = x.iloc[t], x.iloc[v]
        y_train, y_val = y.iloc[t], y.iloc[v]

        model.fit(x_train, y_train)

        pred = model.predict(x_val)
        accuracy = accuracy_score(y_val, pred)

    return accuracy
```

```
# 구독 해지 예측 최적 파라미터
study = optuna.create_study(direction='maximize')
study.optimize(lambda trial: model_knn(trial, x_train, x_val, y_train, y_val), n_trials=100, n_jobs=-1)
best_knn = study.best_trial.params
```

```
# 구독 유형 추천 최적 파라미터
Study = optuna.create_study(direction='maximize')
Study.optimize(lambda trial: model_knn(trial, X_train, X_val, Y_train, Y_val), n_trials=100, n_jobs=-1)
Best_knn = Study.best_trial.params
```

```
print(f'KNN - 구독 해지 예측 최적 파라미터:{study.best_trial.params}')
print(f'KNN - 구독 해지 예측 최적 파라미터의 정확도:{study.best_trial.values[0]}')

print(f'KNN - 구독 유형 추천 최적 파라미터:{Study.best_trial.params}')
print(f'KNN - 구독 유형 추천 최적 파라미터의 정확도:{Study.best_trial.values[0]}')
```

- 파라미터 그리드 생성 후, 모델링 진행
- 성능 개선을 위해 K-fold, n=5 교차 검증
- 검증 후 평가 지표 리턴

- optuna 활용하여 하이퍼파라미터 최적화

- 최적 파라미터 출력

3 모델링: KNN

```
[I 2023-12-28 22:29:10,323] A new study created in memory with name: no-name-9ab6affd-94a5-4c98-841b-187c8b114ee0
[I 2023-12-28 22:29:13,504] Trial 1 finished with value: 0.594 and parameters: {'n_neighbors': 29, 'weights': 'distance', 'metric': 'euclidean', 'p': 1}. Best is trial 1 with value: 0.594.
[I 2023-12-28 22:29:13,600] Trial 3 finished with value: 0.5925 and parameters: {'n_neighbors': 26, 'weights': 'distance', 'metric': 'euclidean', 'p': 1}. Best is trial 1 with value: 0.594.
[I 2023-12-28 22:29:14,196] Trial 0 finished with value: 0.602 and parameters: {'n_neighbors': 46, 'weights': 'distance', 'metric': 'manhattan', 'p': 2}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:16,875] Trial 2 finished with value: 0.5615 and parameters: {'n_neighbors': 22, 'weights': 'uniform', 'metric': 'manhattan', 'p': 1}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:19,742] Trial 7 finished with value: 0.5855 and parameters: {'n_neighbors': 14, 'weights': 'uniform', 'metric': 'euclidean', 'p': 4}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:20,198] Trial 4 finished with value: 0.5775 and parameters: {'n_neighbors': 16, 'weights': 'uniform', 'metric': 'euclidean', 'p': 1}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:20,296] Trial 5 finished with value: 0.5895 and parameters: {'n_neighbors': 33, 'weights': 'uniform', 'metric': 'euclidean', 'p': 3}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:21,621] Trial 6 finished with value: 0.578 and parameters: {'n_neighbors': 5, 'weights': 'uniform', 'metric': 'manhattan', 'p': 3}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:24,264] Trial 10 finished with value: 0.601 and parameters: {'n_neighbors': 50, 'weights': 'distance', 'metric': 'manhattan', 'p': 2}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:25,564] Trial 8 finished with value: 0.588 and parameters: {'n_neighbors': 44, 'weights': 'uniform', 'metric': 'euclidean', 'p': 2}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:27,901] Trial 11 finished with value: 0.5955 and parameters: {'n_neighbors': 47, 'weights': 'uniform', 'metric': 'euclidean', 'p': 2}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:28,019] Trial 9 finished with value: 0.6005 and parameters: {'n_neighbors': 41, 'weights': 'uniform', 'metric': 'manhattan', 'p': 1}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:28,008] Trial 13 finished with value: 0.591 and parameters: {'n_neighbors': 39, 'weights': 'distance', 'metric': 'manhattan', 'p': 3}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:30,520] Trial 12 finished with value: 0.5805 and parameters: {'n_neighbors': 32, 'weights': 'uniform', 'metric': 'euclidean', 'p': 4}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:33,918] Trial 14 finished with value: 0.601 and parameters: {'n_neighbors': 50, 'weights': 'distance', 'metric': 'manhattan', 'p': 3}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:33,929] Trial 15 finished with value: 0.601 and parameters: {'n_neighbors': 50, 'weights': 'distance', 'metric': 'manhattan', 'p': 3}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:36,487] Trial 17 finished with value: 0.598 and parameters: {'n_neighbors': 49, 'weights': 'distance', 'metric': 'manhattan', 'p': 2}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:36,523] Trial 16 finished with value: 0.602 and parameters: {'n_neighbors': 48, 'weights': 'distance', 'metric': 'manhattan', 'p': 2}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:40,515] Trial 18 finished with value: 0.5915 and parameters: {'n_neighbors': 38, 'weights': 'distance', 'metric': 'manhattan', 'p': 2}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:40,591] Trial 19 finished with value: 0.591 and parameters: {'n_neighbors': 39, 'weights': 'distance', 'metric': 'manhattan', 'p': 2}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:43,626] Trial 21 finished with value: 0.5925 and parameters: {'n_neighbors': 37, 'weights': 'distance', 'metric': 'manhattan', 'p': 2}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:43,636] Trial 20 finished with value: 0.5915 and parameters: {'n_neighbors': 38, 'weights': 'distance', 'metric': 'manhattan', 'p': 2}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:47,887] Trial 22 finished with value: 0.556 and parameters: {'n_neighbors': 3, 'weights': 'distance', 'metric': 'manhattan', 'p': 4}. Best is trial 0 with value: 0.602.
[I 2023-12-28 22:29:47,920] Trial 23 finished with value: 0.598 and parameters: {'n_neighbors': 44, 'weights': 'distance', 'metric': 'manhattan', 'p': 4}. Best is trial 0 with value: 0.602.
...
...
```

| | |
|---------------|-------------------------------|
| • n_neighbors | K 값 |
| • uniform | 근접 이웃 가중치 무시, 모든 이웃 가중치 동일 부여 |
| • distance | 근접 이웃 가중치 적용, 근접 이웃 가중치 추가 부여 |
| • manhattan | 이웃간 거리를 좌표 거리로 계산 |
| • euclidean | 이웃간 거리를 직선 거리로 계산 |
| • p | 거리 측정 시, 사용되는 변수 값 |

검증 코드 실행 결과

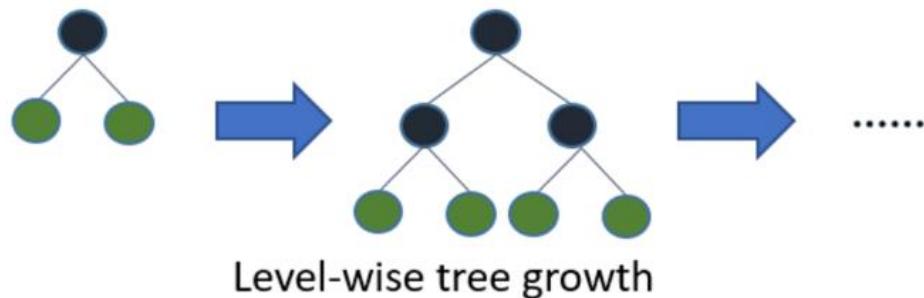
```
KNN - 구독 해지 예측 최적 파라미터:{'n_neighbors': 47, 'weights': 'uniform', 'metric': 'manhattan', 'p': 5}
KNN - 구독 해지 예측 최적 파라미터의 정확도:0.605
KNN - 구독 유형 추천 최적 파라미터:{'n_neighbors': 49, 'weights': 'uniform', 'metric': 'manhattan', 'p': 5}
KNN - 구독 유형 추천 최적 파라미터의 정확도:0.6025
```

- optuna를 활용한 파라미터 테스트 총 100회 시행

• 정확도 결과

- 구독 해지 예측 정확도 : 0.605
- 구독 유형 예측 정확도 : 0.602

3 모델링: XGBoost



POINT 01

GBM기반으로 병렬학습을 구현한 알고리즘

POINT 02

직전 트리 틀린 결과에 가중치 적용,
다음 트리 생성하여 분류 반복

POINT 03

병렬 학습으로 분류 속도가 빠르며, 과적합 규제 기능
해석과 파라미터 튜닝 난이도가 높음

3 모델링: XGBoost

```
import xgboost as xgb
from sklearn.model_selection import GridSearchCV

def model_xgb(x,y):
    model = xgb.XGBClassifier()
    param_grid= {
        'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9],
        'min_child_weight': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
        'subsample': [0.3, 0.5, 0.55, 0.6, 0.7],
        'colsample_bytree': [0.3, 0.4, 0.5, 0.6, 0.7, 0.8],
        'n_estimators': [104, 109, 210, 300, 350, 500]
    }
    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=3, n_jobs=-1,
verbose=2)
    grid_search.fit(x, y)

    return [grid_search.best_params_, grid_search.best_score_]
```

- 조정하기 전의 정확도가 낮아 파라미터 조정을 결정
- 최적의 Hyper Parameter 탐색 목적 → **GridSearch** 사용
- 정확도가 높이기 위해 파라미터를 정밀하게 세분화
- 실행 시간이 오래 걸려 미리 돌린 값들을 사용

3 모델링: XGBoost

```
# XGboost fitting은 시간이 오래 걸렸으므로 결과만 따로 불러와 사용하도록 함.
```

```
best_xgb = {  
    'colsample_bytree':0.5,  
    'n_estimators':104,  
    'max_depth':1,  
    'min_child_weight':10,  
    'random_state':12,  
    'subsample':0.7,  
    'n_jobs':-1,  
}
```

```
Best_xgb = {  
    'colsample_bytree':0.5,  
    'n_estimators':109,  
    'max_depth':1,  
    'min_child_weight':3,  
    'random_state':12,  
    'subsample':0.6,  
    'n_jobs':-1,  
}
```

- 앞선 GridSearch를 통해 얻은 값으로 구성
- 2개 모델 최적의 하이퍼파라미터가 다름
→ 2개의 함수로 나눠서 작성

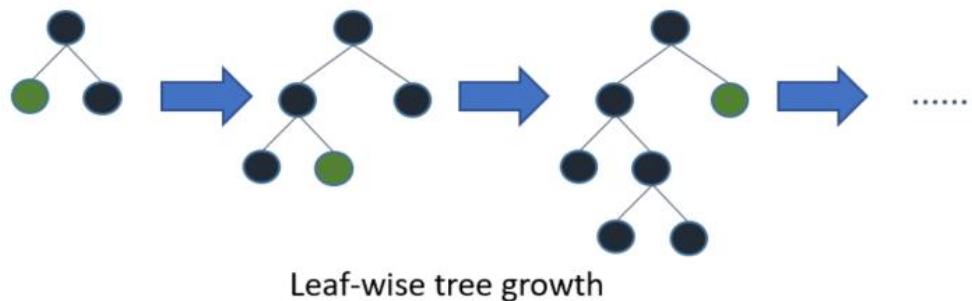
• 정확도 결과

- 구독 해지 예측 정확도 : 0.6156
- 구독 유형 예측 정확도 : 0.7695

검증 코드 실행 결과

```
XGboost - 구독 해지 예측 최적 파라미터:{{'colsample_bytree': 0.5, 'max_depth': 1, 'min_child_weight': 10,  
'n_estimators': 104, 'subsample': 0.7}} XGboost - 구독 해지 예측 최적 파라미터의 정확도:0.6156  
XGboost - 구독 유형 추천 최적 파라미터:{{'colsample_bytree': 0.5, 'max_depth': 1, 'min_child_weight': 3,  
'n_estimators': 109, 'subsample': 0.6}} XGboost - 구독 유형 추천 최적 파라미터의 정확도:0.7695
```

3 모델링: LightGBM



POINT 01

기존 Boosting 모델과 달리 **리프 노드 기준으로 트리 분할** 알고리즘

POINT 02

트리 균형을 고려하지 않고 최대 손실값 가지는 리프 노드를 지속 분류

POINT 03

메모리 사용량 적어, 큰 데이터셋에서 처리속도 우수
XGBoost 대비, 더 높은 과적합 소지

3 모델링: LightGBM

```
from lightgbm import LGBMClassifier

def model_lgbm(trial, x_train, x_val, y_train, y_val):

    # optuna를 이용한 하이퍼 파라미터 조정
    num_leaves = trial.suggest_int('num_leaves', 20, 3000, log=True)
    max_depth = trial.suggest_int('max_depth', 3, 10)
    learning_rate = trial.suggest_float('learning_rate', 0.01, 0.5)
    subsample = trial.suggest_float('subsample', 0.5, 1.0)
    colsample_bytree = trial.suggest_float('colsample_bytree', 0.5, 1.0)

    # 모델 생성 및 훈련
    model = LGBMClassifier(
        learning_rate=learning_rate,
        max_depth=max_depth,
        subsample=subsample,
        colsample_bytree=colsample_bytree,
        num_leaves=num_leaves,
        random_state=12
    )
    model.fit(x_train, y_train)

    # 검증 세트에서의 성능 평가
    pred = model.predict(x_val)
    accuracy = accuracy_score(y_val, pred)
    return accuracy
```

- 최적의 Hyper Parameter를 찾기 위해 Optuna 라이브러리를 사용

설정할 Parameter List

- num_leaves = 트리가 가질 수 있는 최대 리프의 수를 지정
- max_depth = 각 결정 트리의 최대 깊이를 결정
- learning_rate = 각 부스팅 단계에서 적용되는 학습률 조절
- subsample = 훈련 데이터 샘플의 비율을 설정
- colsample_bytree = 각 트리를 구축할 때 사용할 특성의 비율을 결정

3 모델링: LightGBM

```
# 구독 해지 예측 최적 파라미터
study = optuna.create_study(direction='maximize')
study.optimize(lambda trial: model_lgbm(trial, x_train, x_val, y_train, y_val), n_trials=10, n_jobs=-1)
best_lgbm = study.best_trial.params

# 구독 유형 추천 최적 파라미터
Study = optuna.create_study(direction='maximize')
Study.optimize(lambda trial: model_lgbm(trial, X_train, X_val, Y_train, Y_val), n_trials=10, n_jobs=-1)
Best_lgbm = Study.best_trial.params

print(f'LightGBM - 구독 해지 예측 최적 파라미터:{study.best_trial.params}')
print(f'LightGBM - 구독 해지 예측 최적 파라미터의 정확도:{study.best_trial.values[0]}')

print(f'LightGBM - 구독 유형 추천 최적 파라미터:{Study.best_trial.params}')
print(f'LightGBM - 구독 유형 추천 최적 파라미터의 정확도:{Study.best_trial.values[0]}')
```

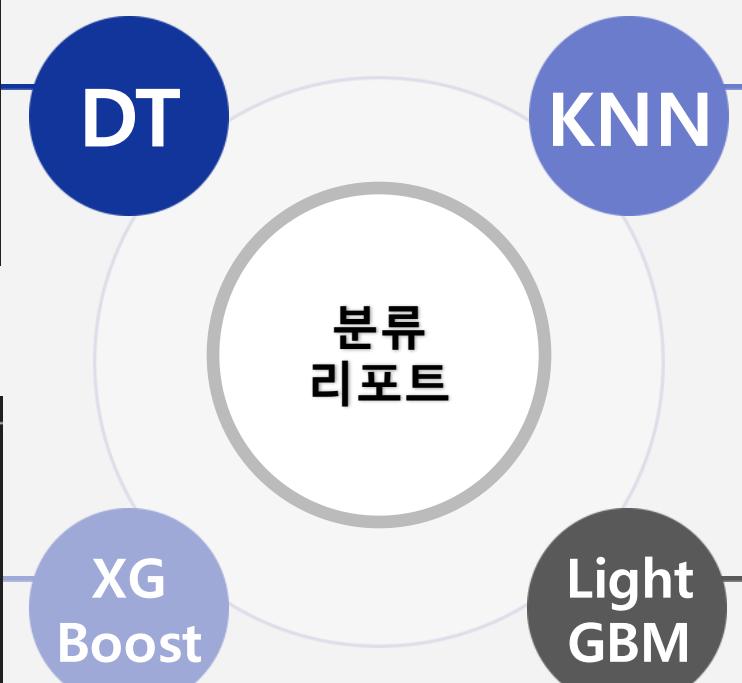
- 모델에 맞는 Hyper Parameter 범위 정의

검증 코드 실행 결과

```
LightGBM - 구독 해지 예측 최적 파라미터:{'num_leaves': 264, 'max_depth': 3, 'learning_rate': 0.07378972832681366, 'subsample': 0.746044039369156, 'colsample_bytree': 0.7310619069063958} LightGBM -
구독 해지 예측 최적 파라미터의 정확도:0.628
LightGBM - 구독 유형 추천 최적 파라미터:{'num_leaves': 58, 'max_depth': 3, 'learning_rate': 0.09067421701828435, 'subsample': 0.6577685444377724, 'colsample_bytree': 0.5533182529399239} LightGBM -
구독 유형 추천 최적 파라미터의 정확도:0.7758
```

3 모델링: 종합(구독해지 예측)

| DT | | | | |
|--------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.71 | 0.23 | 0.34 | 3801 |
| 1 | 0.67 | 0.94 | 0.78 | 6199 |
| accuracy | | | 0.67 | 10000 |
| macro avg | 0.69 | 0.58 | 0.56 | 10000 |
| weighted avg | 0.68 | 0.67 | 0.61 | 10000 |

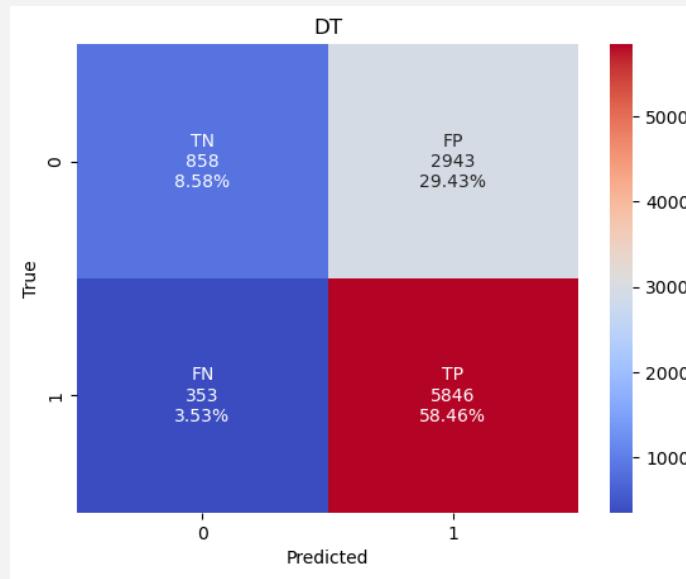


| KNN | | | | |
|--------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.52 | 0.09 | 0.15 | 3801 |
| 1 | 0.63 | 0.95 | 0.76 | 6199 |
| accuracy | | | | 0.62 |
| macro avg | | 0.58 | 0.52 | 0.45 |
| weighted avg | 0.59 | 0.62 | 0.53 | 10000 |

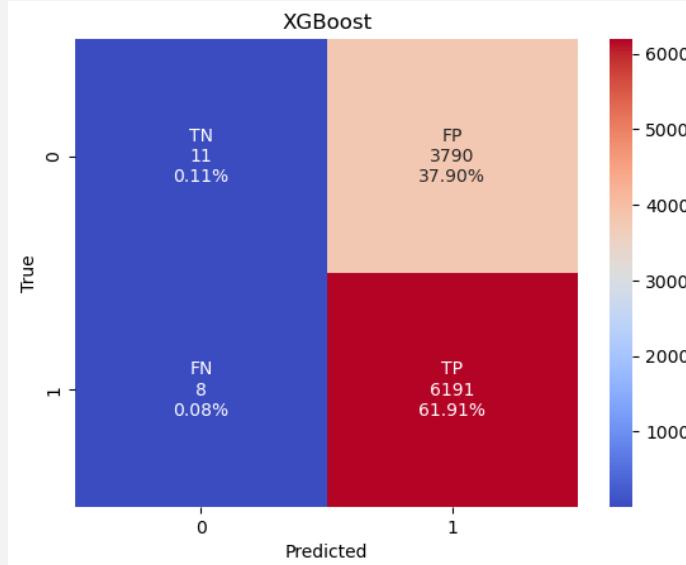
| XGBoost | | | | |
|--------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.58 | 0.00 | 0.01 | 3801 |
| 1 | 0.62 | 1.00 | 0.77 | 6199 |
| accuracy | | | 0.62 | 10000 |
| macro avg | 0.60 | 0.50 | 0.39 | 10000 |
| weighted avg | 0.60 | 0.62 | 0.48 | 10000 |

| LightGBM | | | | |
|--------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.97 | 0.02 | 0.03 | 3801 |
| 1 | 0.62 | 1.00 | 0.77 | 6199 |
| accuracy | | | | 0.63 |
| macro avg | 0.80 | 0.51 | 0.40 | 10000 |
| weighted avg | 0.76 | 0.63 | 0.49 | 10000 |

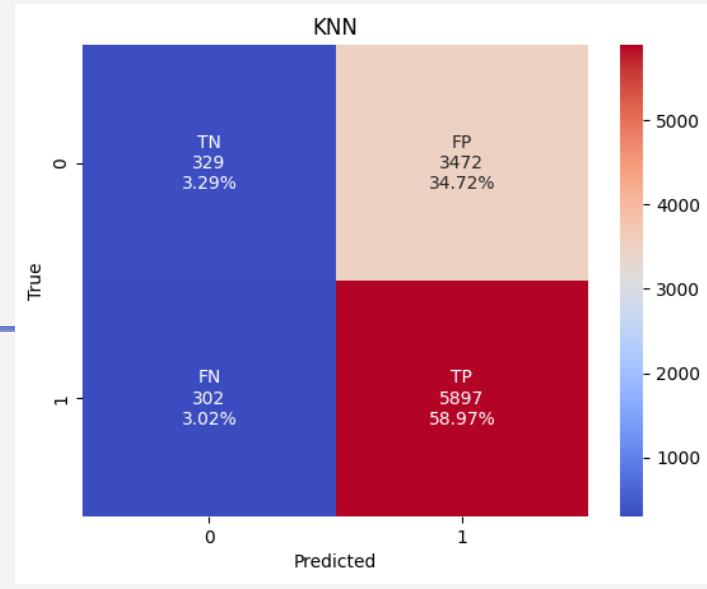
3 모델링: 종합(구독해지 예측)



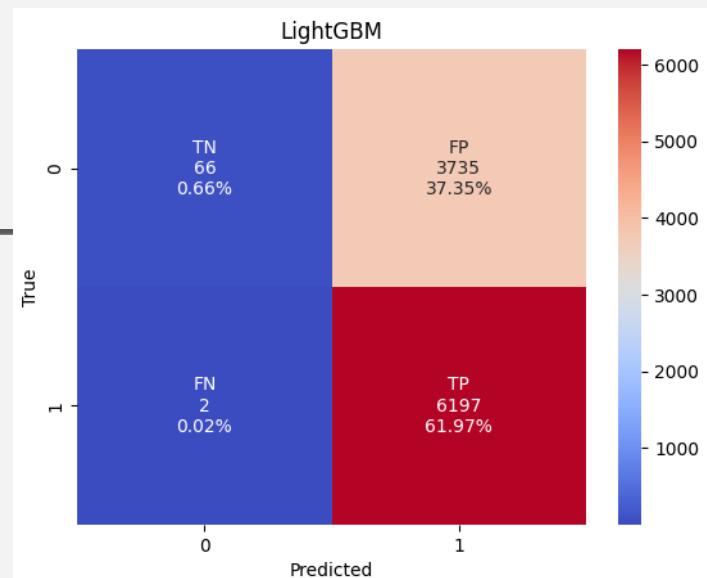
DT



XG
Boost



KNN



Light
GBM

혼동행렬

3 모델링: 종합(구독유형 추천)

| DT | | | | |
|--------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.71 | 0.79 | 0.75 | 5028 |
| 1 | 0.76 | 0.68 | 0.72 | 4972 |
| accuracy | | | 0.73 | 10000 |
| macro avg | 0.73 | 0.73 | 0.73 | 10000 |
| weighted avg | 0.73 | 0.73 | 0.73 | 10000 |

DT

| KNN | | | | |
|--------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.68 | 0.80 | 0.73 | 5028 |
| 1 | 0.75 | 0.62 | 0.68 | 4972 |
| accuracy | | | 0.71 | 10000 |
| macro avg | 0.72 | 0.71 | 0.71 | 10000 |
| weighted avg | 0.72 | 0.71 | 0.71 | 10000 |

KNN

| XGBoost | | | | |
|--------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.76 | 0.80 | 0.78 | 5028 |
| 1 | 0.78 | 0.74 | 0.76 | 4972 |
| accuracy | | | 0.77 | 10000 |
| macro avg | 0.77 | 0.77 | 0.77 | 10000 |
| weighted avg | 0.77 | 0.77 | 0.77 | 10000 |

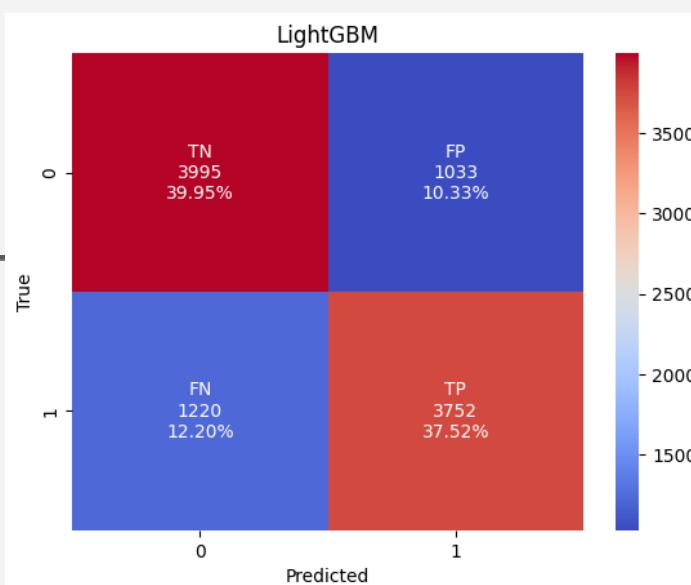
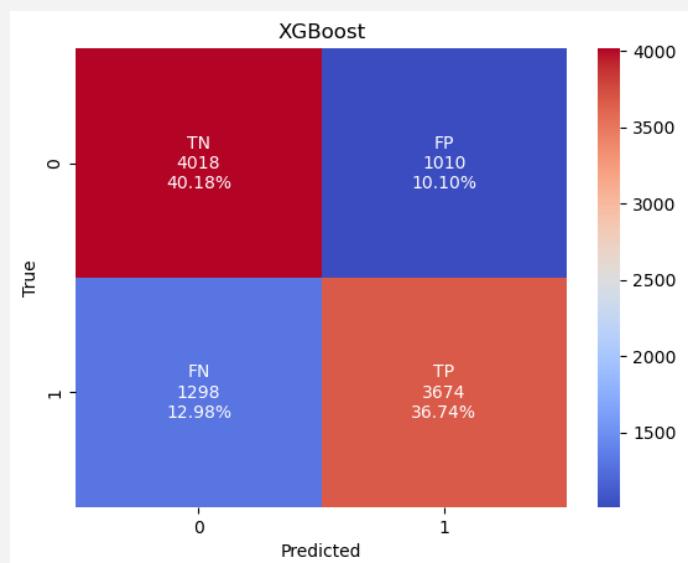
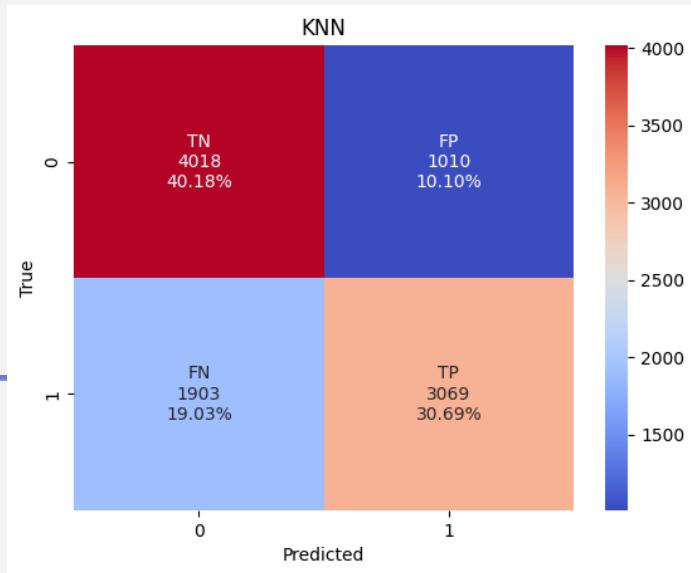
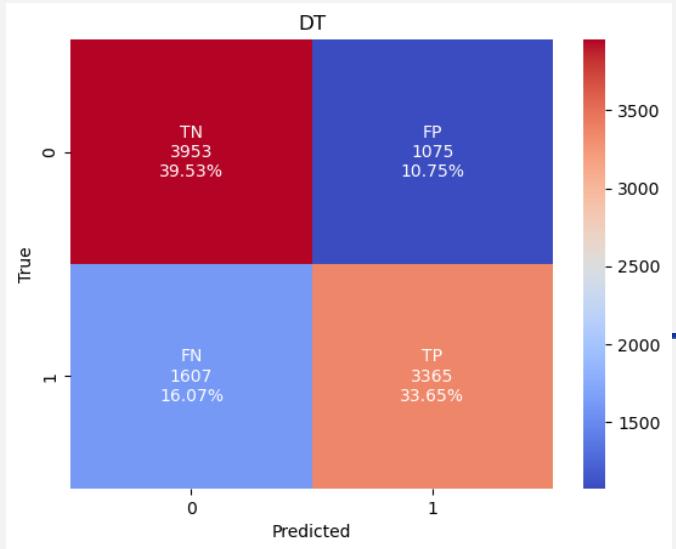
XG
Boost

| LightGBM | | | | |
|--------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.77 | 0.79 | 0.78 | 5028 |
| 1 | 0.78 | 0.75 | 0.77 | 4972 |
| accuracy | | | 0.77 | 10000 |
| macro avg | 0.78 | 0.77 | 0.77 | 10000 |
| weighted avg | 0.78 | 0.77 | 0.77 | 10000 |

Light
GBM

분류
리포트

3 모델링: 종합(구독유형 추천)



CONTENTS

I

프로젝트개요

- 1. 주제 배경
- 2. 프로젝트 의의
- 3. 진행 일정
- 4. 활용 도구

II

데이터수집·처리

- 1. 데이터 수집
- 2. 데이터 확인
- 3. 데이터 구분
- 4. 데이터 인코딩

III

데이터분석

- 1. EDA
- 2. 데이터 전처리
- 3. 모델링
- 4. 모델링 평가

IV

결론

- 1. 프로젝트 개선점
- 2. 프로젝트 평가
- 3. 인용 · 출처

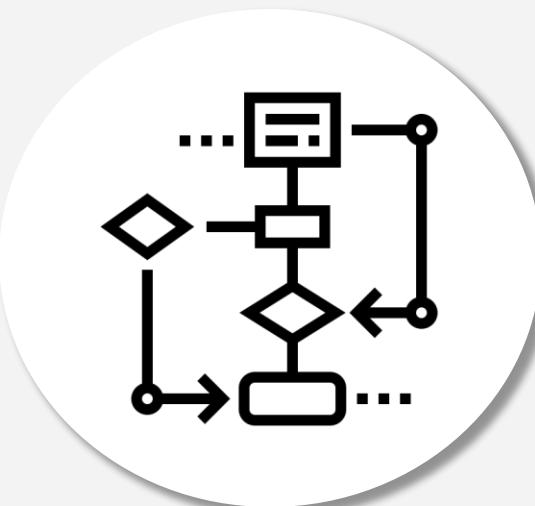
1

프로젝트 개선점

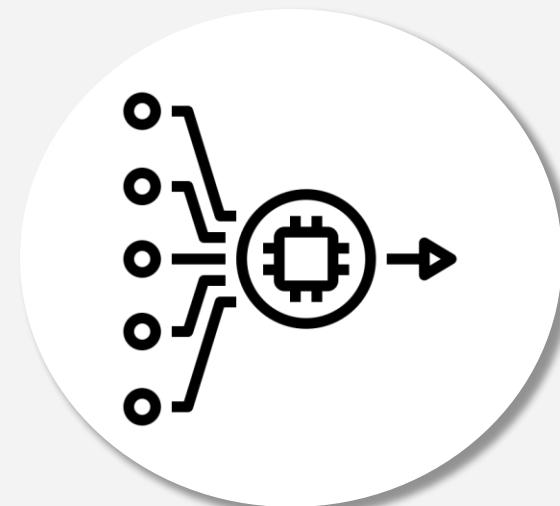
Plan



Process



Product



- 충분한 분석 기간, 부족한 취합 기간
- 시간 부족으로 앙상블 학습 적용한 디벨롭 제한

- EDA 시각화 그래프 코드 작성, 수정 리소스 과다
- EDA 이후 일정 시작 딜레이 발생

- 데이터셋 관련 문제
- 모델 정확도가 대체적으로 낮음

2 프로젝트 평가



머신러닝 모델 구현 중 시행착오를 통한 지식과 경험 획득



플랫폼 서비스 개선에 기여할 수 있는 **프로토타입** **프로덕트** 개발



실제 활용 가능한 **프로덕트** 개발의 시작점

3 인용 · 출처

| Page | Div | Contents | Source |
|------|-----------|---|--------------------|
| | 프로젝트 의의 | 맥킨지, 2025 플랫폼 시장 전망 | 링크 |
| | 프로젝트 의의 | KT 엔터프라이즈, '맥킨지, 2025 플랫폼 시장 전망' 관련 포스팅 | 링크 |
| | 프로젝트 의의 | Etnews, 국내 에듀테크 시장, 2025년 10조원 규모 성장 전망 | 링크 |
| | 데이터 수집 | 학습 플랫폼 구독자 예측 해커톤 csv파일 | 링크 |
| | DT | DecisionTreeClassifier + 라벨인코딩 + 로그변환 | 링크 |
| | KNN | K-NN 알고리즘 코드 인용, KNN모델, optuna | 링크 |
| | KNN | K-NN 알고리즘 코드 인용, KNN모델, 라벨인코딩 | 링크 |
| | KNN | K-NN 알고리즘 코드 인용, Knn + optuna | 링크 |
| | optuna | 공식 홈페이지 참고 | 링크 |
| | optuna | Optuna를 활용한 하이퍼파라미터 최적화 | 링크 |
| | optuna | Hyperparameter 조정 : optuna | 링크 |
| | Xgboost | 하이퍼파라미터 조정을 통한 XG부스트 모델 | 링크 |
| | Xgboost | 공식 홈페이지 참고 | 링크 |
| | Xgboost | 개념 정리 및 하이퍼파라미터 참고 | 링크 |
| | Light GBM | 공식 홈페이지 참고 | 링크 |
| | DT | DT 알고리즘 개념 | 링크 |
| | | | 링크 |
| | | | 링크 |
| | | | 링크 |

E.O.D

Q&A

부제목으로 사용한 맑은고딕 세미라이트 볼드체

제목으로 사용한 맑은고딕 기본 볼드체

본문으로 사용한 맑은고딕 세미라이트 볼드체입니다.
가독성이 좋은 문서를 위해서 직접 타이핑을 해보고
자간/행간을 조절해보시기 바랍니다.

본문으로 사용한 맑은고딕 기본체입니다.
가독성이 좋은 문서를 위해서 직접 타이핑을 해보고
자간/행간을 조절해보시기 바랍니다.

또 다른 디자인으로 더 나은 세상을 꿈꾸는
We are another branding group.

파워포인트에서도 폰트를 세밀하게 다듬으면
전문 디자이너가 작업한 것과 같이 예쁜 타이포그래피를 만들 수 있습니다.
참고로 폰트에 따라 자간/행간조절이 달라질 수 있기 때문에 무작정 따라하기보다는
'가독성을 좋게 만드는 것'을 목표로해야 합니다.

같은 폰트라고 해도 폰트 조절을 어떻게 하는지에 따라 퀄리티가 달라지기 마련인데
세밀한 폰트 조절을 통해 보다 가독성 좋은 문서로 완성하시길 바라겠습니다!