

# EDA와 전처리

# 데이터타입과 결측치 비율 확인

user\_spec.csv

Column Name	Date Type	Missing Value
application_id	int64	0.0%
user_id	int64	0.0%
birth_year	float64	0.9%
gender	float64	< 0.1%
insert_time	object	0.0%
credit_score	float64	7.5%
yearly_income	float64	< 0.1%
income_type	object	< 0.1%

user\_spec.csv

Column Name	Date Type	Missing Value
company_enter_month	float64	12.3%
employment_type	object	< 0.1%
houseown_type	object	< 0.1%
desired_amount	float64	< 0.1%
purpose	object	< 0.1%
personal_rehabilitation_yn	float64	42.1%
personal_rehabilitation_complete_yn	float64	86.3%
existing_loan_cnt	float64	14.2%
existing_loan_amt	float64	22.5%

### loan\_result.csv

Column Name	Date Type	Missing Value
application_id	int64	0.0%
loanapply_insert_time	object	0.0%
bank_id	int64	0.0%
product_id	int64	0.0%
loan_limit	float64	0.1%
loan_rate	float64	0.1%
is_applied	float64	24.1%

### log\_data.csv

Column Name	Date Type	Missing Value
user_id	int64	0.0%
event	object	0.0%
timestamp	object	0.0%
mp_op	object	< 0.1%
mp_app_version	object	3.7%
data_cd	object	0.0%

# 문자형 데이터 : 한/영 표기 변환

user\_spec.csv

purpose

- (생활비, LIVING) → LIVING

employment\_type

- 정규직 → FULLTIME

houseown\_type

- 자가 → OWNED

# 수치형 데이터 : 함수식 설계

$$Age = 2022 - BirthYear$$

birth_year		age
1985.0	➡	37
1968.0		54
1997.0		25

# 가변수화

- income\_type
- employment\_type
- houseown\_type
- purpose
- personal\_rehabilitation\_yn
- personal\_rehabilitation\_complete\_yn

Pandas.get\_dummies(data, columns = [ ], drop\_first=True)

: 범주형 변수를 dummy/indicator 변수로 변환

houseown_type	houseown_type_LENT	houseown_type_OWNED	houseown_type_SPOUSE
FAMILY	0	0	0
FAMILY	0	0	0
OWNED	0	1	0

: 결측값을 범주형 변수로 인식 및 변환

personal_rehabilitation_complete_yn	personal_rehabilitation_complete_yn_0.0	personal_rehabilitation_complete_yn_1.0
NaN	0	0
NaN	0	0
NaN	0	0

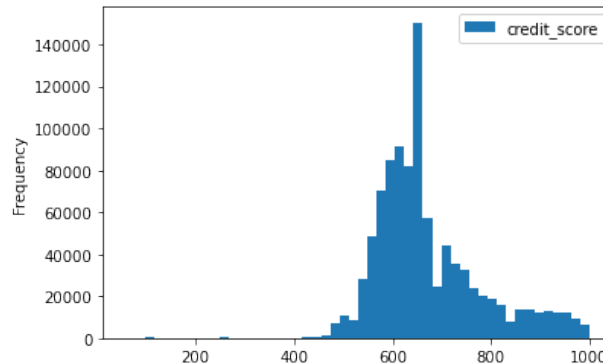
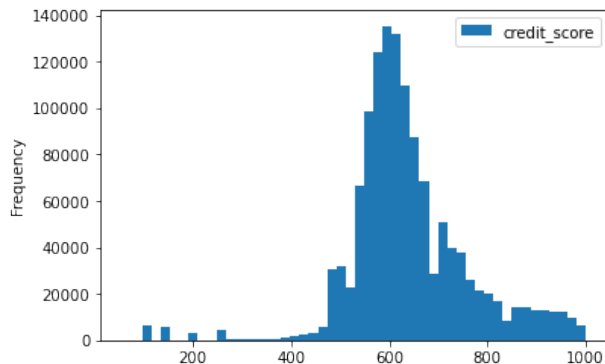
# 결측값 처리

결측값이 적은 경우, 결측값이 존재하는 행 제거

- age
- gender
- income\_type
- employment\_type
- houseown\_type
- desired\_amount
- purpose
- yearly\_income
- loan\_result
- loan\_rate

$$\frac{\text{Missing Data}}{\text{Total Data}} < 0.1\%$$

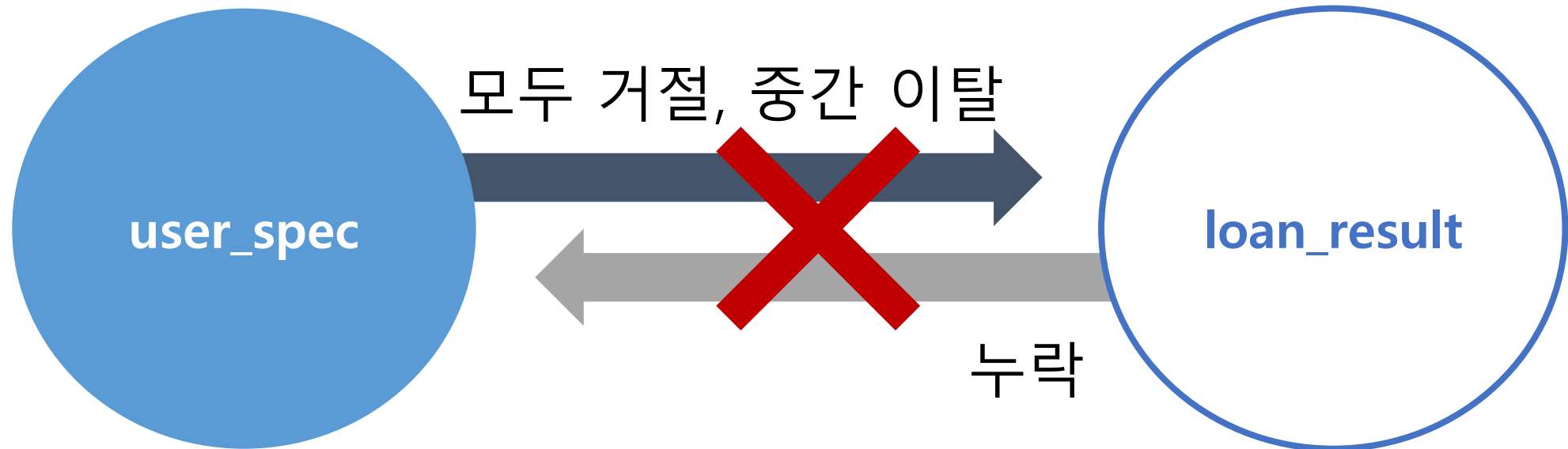
결측값의 많은 경우, 데이터 분포에 따라 보간



$$\text{median}(\text{credit\_score}) = 650.0$$

# 결측값 처리

application\_id가 결측된 행 제거





# 변수 선택

## 분석 방향성이 불일치하는 변수 삭제

- log\_data.csv
- insert\_time
- user\_id

## 다른 변수에 종속된 변수 삭제

- bank\_id  $\subset$  product\_id

## 의미가 불분명한 변수 삭제

### company\_enter\_month 변수 표기방식

- YYYY.MM.DD
- YYYY.MM
- YY.MM.DD

company\_enter\_month

201101.0

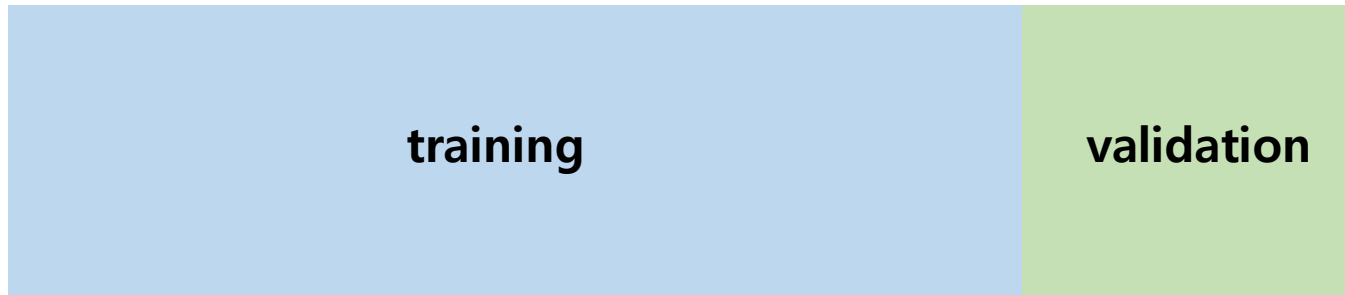
20년 11월 1일 ?  
2011년 1월 ?

# 모델링

**training data와 validation data로 split**

: 모델의 분산을 줄여 overfitting을 막기 위해 training data와 validation data로 split

(training data: 80% / validation data: 20%)



# 모델링

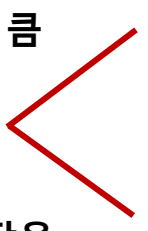
## 상품별 로지스틱 회귀 모형 만들기

예시)

$$\log\left(\frac{p}{1-p}\right) = \beta_1 + \beta_2 x_{age} + \beta_3 x_{gender} + \cdots + \beta_n x_{credit\_score}$$

$$\hat{p} = \frac{1}{1 + x^{\beta_1 + \beta_2 x_{age} + \beta_3 x_{gender} + \cdots + \beta_n x_{credit\_score}}}$$

임계치보다 큼 **1**  
**0과 1 사이의 값이 나옴**  
임계치보다 작음 **0**



# 모델링

문제점: 상품이 총 188개

→ 유사한 상품끼리 군집화

```
# unique한 product_id 개수  
len(np.unique(loan_result["product_id"]))
```

188

loan\_limit

Q3

Q2

Q1

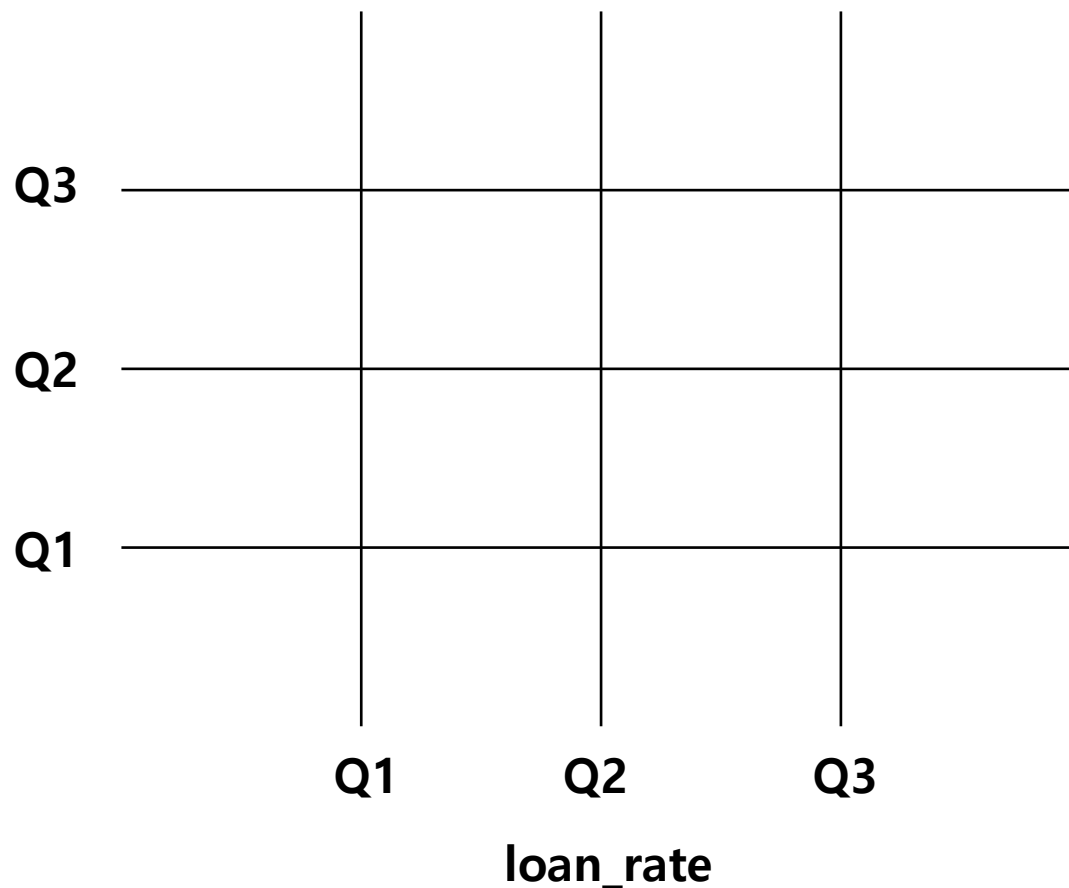
Q1

Q2

Q3

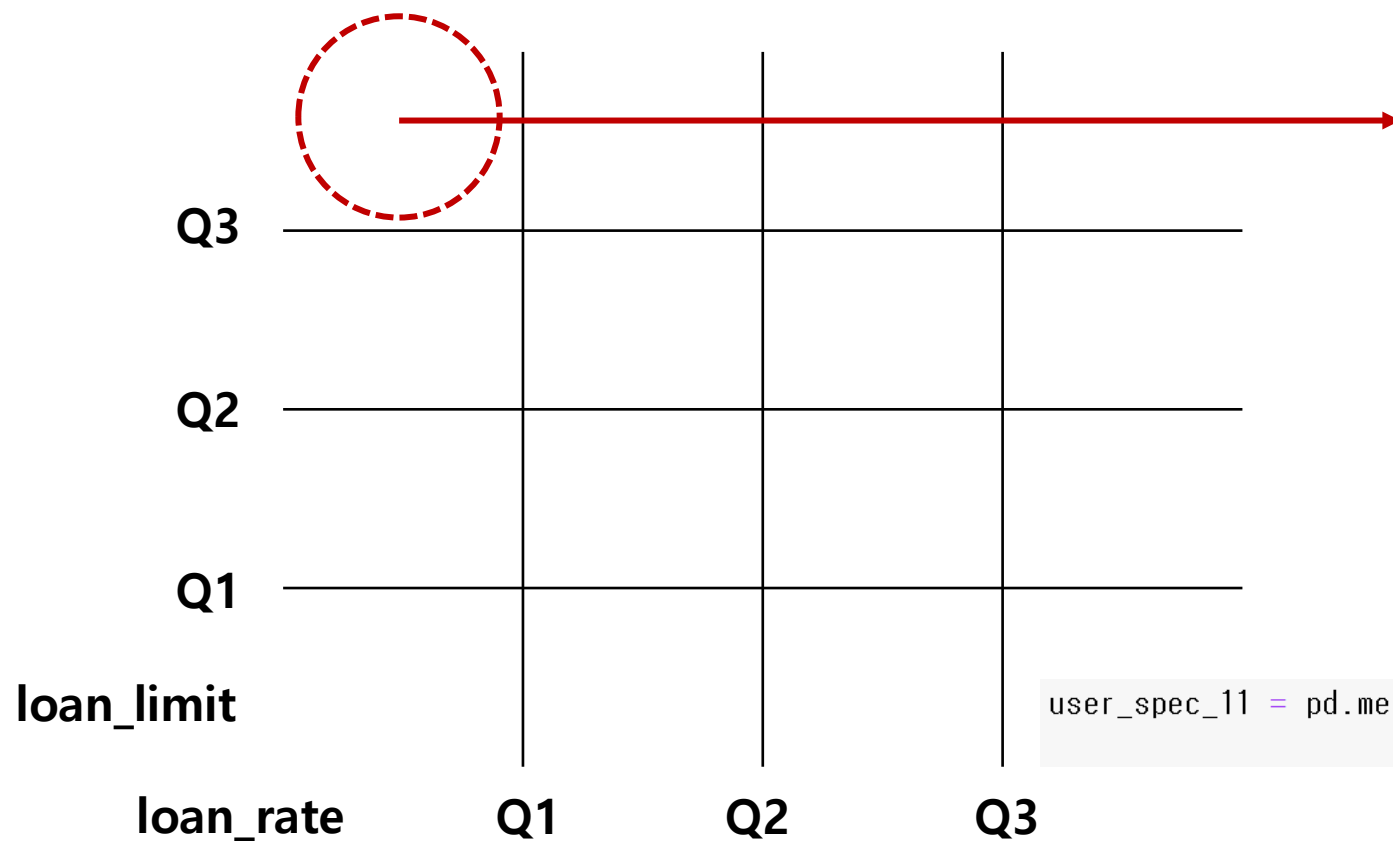
loan\_rate

총 16개의 로지스틱 회귀 모형이 나옴



# 모델링

총 16개의 로지스틱 회귀 모형이 나옴



**model\_11**

이 그룹에 속한 product\_id에 해당하는 user\_spec 데이터 프레임을 만들어서 모델링 진행

```
user_spec_11 = pd.merge(left = loan_result_train_11, right = user_spec, how = "left",  
                        left_on = "application_id", right_on = "application_id")
```

# 모델링

## 1. NA 삭제

: 전체 행의 개수에 비해 NA 포함하는 행은 적음

```
# NA 삭제
dropna_column = ['user_id', 'age', 'gender', 'credit_score', 'yearly_income', 'desired_amount',
                 'personal_rehabilitation_yn_0.0', 'personal_rehabilitation_yn_1.0', 'personal_rehabilitation_complete_yn_0.0',
                 'personal_rehabilitation_complete_yn_1.0', 'income_type_EARNEDINCOME2', 'income_type_FREELANCER', 'income_type_OTHERINCOME',
                 'income_type_PRACTITIONER', 'income_type_PRIVATEBUSINESS', 'employment_type_DAYLABOR', 'employment_type_ETC',
                 'employment_type_FULLTIME', 'houseown_type_OWNED', 'houseown_type_RENT', 'houseown_type_SPOUSE', 'purpose_BUYCAR',
                 'purpose_BUYHOUSE', 'purpose_ETC', 'purpose_HOUSEDEPOSIT', 'purpose_INVEST', 'purpose_LIVING', 'purpose_SWITCHLOAN']

user_spec_11.dropna(subset = dropna_column, inplace = True)
```

## 2. 표준화

: 해석력을 위해 모든 feature에 대해 표준화 진행

```
# 표준화
from sklearn.preprocessing import StandardScaler

normalize_column = ['loan_limit', 'loan_rate', 'age', 'credit_score', 'yearly_income', 'desired_amount', 'existing_loan_cnt', 'existing_loan_rate']
user_spec_11[normalize_column] = StandardScaler().fit_transform(user_spec_11[normalize_column])
```

# 모델링

## 3. SMOTE

: 1보다 0이 훨씬 많은 imbalanced data이기 때문에 oversampling 진행

```
smote = SMOTE()

ov_train_x_11, ov_train_y_11 = smote.fit_sample(train_x_11[features], train_y_11)
```

## 4. Feature Selection

: Tree를 통해 중요도가 높은 feature만을 이용해 모델링 진행

```
# feature selection_11
from sklearn.ensemble import ExtraTreesClassifier

etc_model = ExtraTreesClassifier()
etc_model.fit(ov_train_x_11, ov_train_y_11)

print(etc_model.feature_importances_)
feature_list = pd.concat([pd.Series(ov_train_x_11.columns), pd.Series(etc_model.feature_importances_)], axis = 1)
feature_list.columns = ['features_name', 'importance']
feature_list.sort_values("importance", ascending = False)[:8]
```

# 모델링

## 5. Logistic Regression 적용

: L2 regularization을 통해 penalty를 줌

```
from sklearn.linear_model import LogisticRegression
```

```
# model_11  
model_11 = LogisticRegression(penalty = 'l2', C = 2, fit_intercept = True, max_iter = 1000000)  
model_11.fit(ov_train_x_11[selected_feature], ov_train_y_11)
```

C:\Users\Sehee\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

LogisticRegression(C=2, max\_iter=1000000)

```
model_11.score(ov_train_x_11[selected_feature], ov_train_y_11)
```

0.6807872323547017



# 모델링

## 5. Threshold 선택하여 prediction 진행

: threshold별 accuracy를 계산하여 accuracy가 최대인 threshold 선정  
(그 threshold 이상이면 1로 예측하고 아니면 0으로 예측)



# Feature selection

---Cluster 22---

	features_name	importance
4	credit_score	0.280964
1	loan_rate	0.114569
0	loan_limit	0.093630
2	age	0.088469
6	desired_amount	0.087655
5	yearly_income	0.074994
27	purpose_LIVING	0.035657
7	personal_rehabilitation_yn_0.0	0.026660

---Cluster 24---

	features_name	importance
4	credit_score	0.247721
2	age	0.118161
1	loan_rate	0.116124
6	desired_amount	0.113876
5	yearly_income	0.096030
0	loan_limit	0.091410
27	purpose_LIVING	0.032983
28	purpose_SWITCHLOAN	0.018004

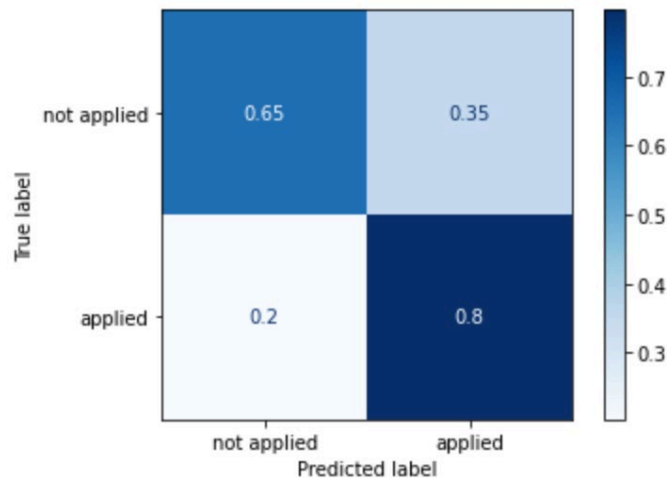
# 결과 분석

- 데이터를 훈련, 테스트 데이터로 분할해 훈련 데이터에 대해 모델 적합  
:테스트 데이터에 대해 성능 확인 후 6월 데이터 예측해 결과물 작성
- 데이터 불균형의 문제를 감안해,  $P(Y = 1|X) > 0.5$ 이면 그대로 1로 예측하는 것이 아니라 훈련 데이터에 대해 최대의 성능을 보이는 문턱 값을 사용함  
:  $P(Y = 1|X) > (\text{문턱 값})$ 이면 1로 예측.
- 성능 평가 방법은 정확도(Accuracy): 전체 예측 중 올바른 예측의 비율

# 결과 분석

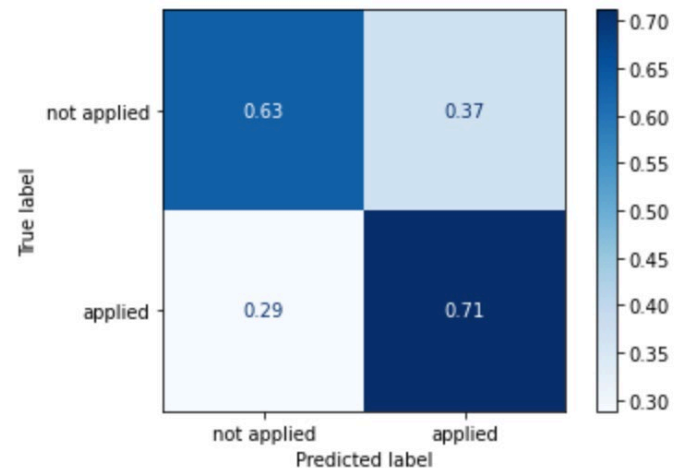
- 정확도는 완전한 지표는 아님: 모델이 Label 0, Label 1 중 하나만 제대로 예측할 수 있음
- 또다른 평가 지표: 재현율(Recall), 정밀도(Precision), F1 Score
- 재현율은 실제로 True인 데이터를 모델이 True라고 인식한 데이터의 수
- 정밀도는 모델이 True로 예측한 데이터 중 실제로 True인 데이터의 수
- F1 score는 재현율과 정밀도의 조화평균
- 훈련 데이터에 대해 최대의 성능을 보이는 문턱 값 기준으로 성능 평가

# 결과 예시



--- Cluster 22 ---

Accuracy(정확도): 0.647  
Precision(정밀도): 0.122  
Recall(재현율): 0.813  
F1 Score: 0.212

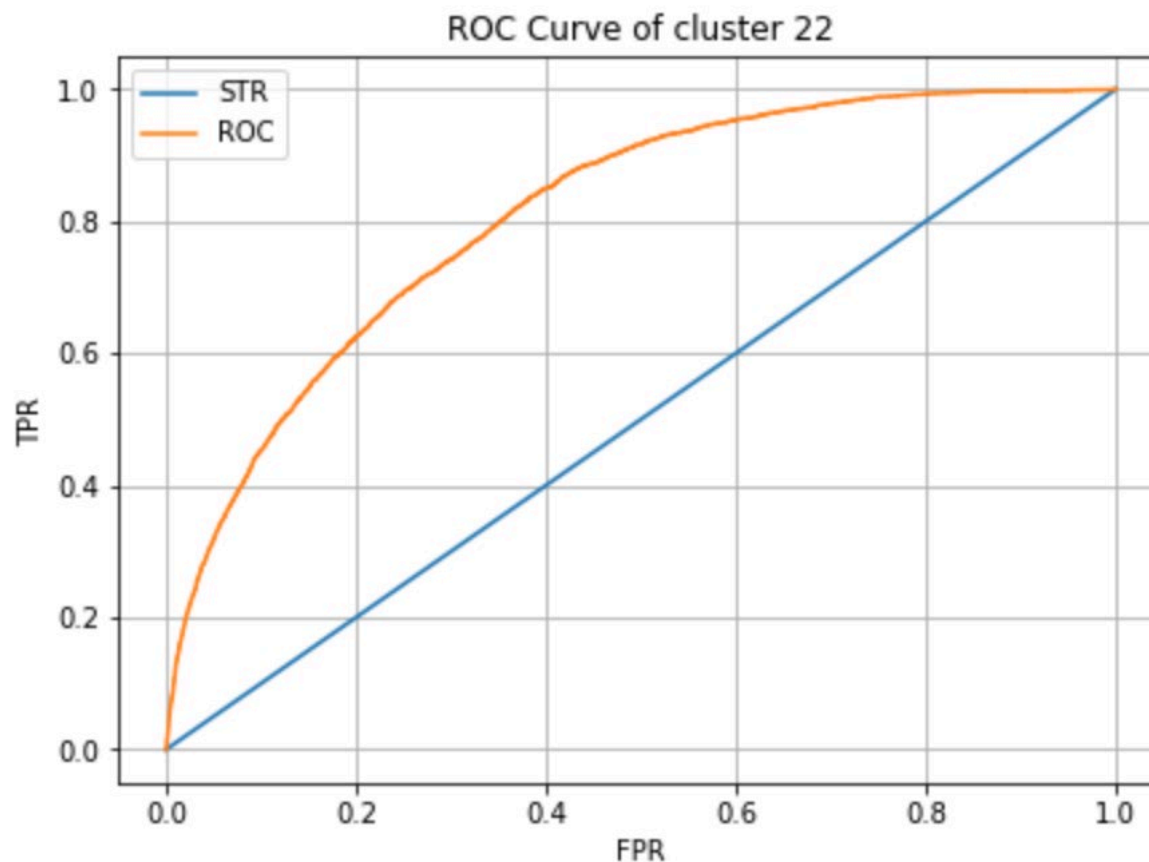


--- Cluster 24 ---

Accuracy(정확도): 0.638  
Precision(정밀도): 0.126  
Recall(재현율): 0.711  
F1 Score: 0.214

# 결과 예시

- ROC Curve: 적합된 모델의 민감도(Sensitivity)와 특이도(Specificity)의 상충관계를 시각적으로 나타낸 그림
- 민감도: 1을 1이라고 판단한 비율 / 특이도: 0을 0이라고 판단한 비율
- 민감도와 특이도는 상충관계로, 둘을 모두 1로 만들 수는 없음  
Ex) 만약 모든 데이터를 1이라고 판단한다면 민감도는 1이지만 특이도는 0
- Roc curve는 진양성 (True positive rate)과 위양성(False positive rate)을 기준으로 그려지며, 진양성 비율이 높을수록 위양성 비율도 높아질 수밖에 없음
- 곡선이 바깥쪽으로 차 있을수록 같은 진양성 비율에 대해 낮은 위양성 비율을 보이므로(특이도가 높으므로) 좋은 모델을 의미.



AUC(ROC Curve의 면적): 0.725