# 정기 예금 가입 여부 예측

AI_02_전혜정

# INTRO

이 데이터를 통해 고객이 정기 예금을 가입할 가능성을 예측하고,

이를 통해 마케팅 캠페인의 효율성을 높이는 것 입니다.

마케팅 담당자로서 정기 예금과 관련이 있는 요소들을 파악해보고, 고객의 행동을 이해해보자.

이번 과제의 목표는 **가장 정확한 분류 모델**을 개발하여 **고객이 정기 예금을 가입할지 여부를 예측**하고,

그 모델을 통해 도출한 인사이트를 바탕으로 **비즈니스 전략을 제시**하는 것이다.

# 데이터 소개

이 데이터는 2008년부터 2010년까지의 은행 마케팅 캠페인 데이터를 포함하고 있습니다.

```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object
dtypes: float64(5), int64(5), object(11)
```

데이터 형태 : (41188, 21)

# 데이터 소개

| 컬럼명 | 설명 |
|---|---|
| age | 나이 (숫자) |
| job | 직업 (범주형) |
| marital | 결혼 여부 (범주형) |
| education | 교육 수준 (범주형) |
| default | 신용 불량 여부 (범주형) |
| housing | 주택 대출 여부 (범주형) |
| loan | 개인 대출 여부 (범주형) |
| contact | 연락 유형 (범주형) |
| month | 마지막 연락 월 (범주형) |
| day_of_week | 마지막 연락 요일 (범주형) |

| 컬럼명 | 설명 |
|---|---|
| duration | 마지막 연락 지속 시간, 초 단위 (숫자) |
| campaign | 캠페인 동안 연락 횟수 (숫자) |
| pdays | 이전 캠페인 후 지난 일수 (숫자) |
| previous | 이전 캠페인 동안 연락 횟수 (숫자) |
| poutcome | 이전 캠페인의 결과 (범주형) |
| emp.var.rate | 고용 변동률 (숫자) |
| cons.price.idx | 소비자 물가지수 (숫자) |
| cons.conf.idx | 소비자 신뢰지수 (숫자) |
| euribor3m | 3개월 유리보 금리 (숫자) |
| nr.employed | 고용자 수 (숫자) |
| y | 정기 예금 가입 여부 (이진: yes=1, no=0) |

# EDA

## 결측치

```
data.isna().sum()
```

```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

: 결측치 없음

## 중복값

```
data.duplicated().sum()
```

```
12
```

```
data.drop_duplicates(inplace=True)
data.duplicated().sum()
```

```
0
```

: 중복값 12개 존재 → 제거

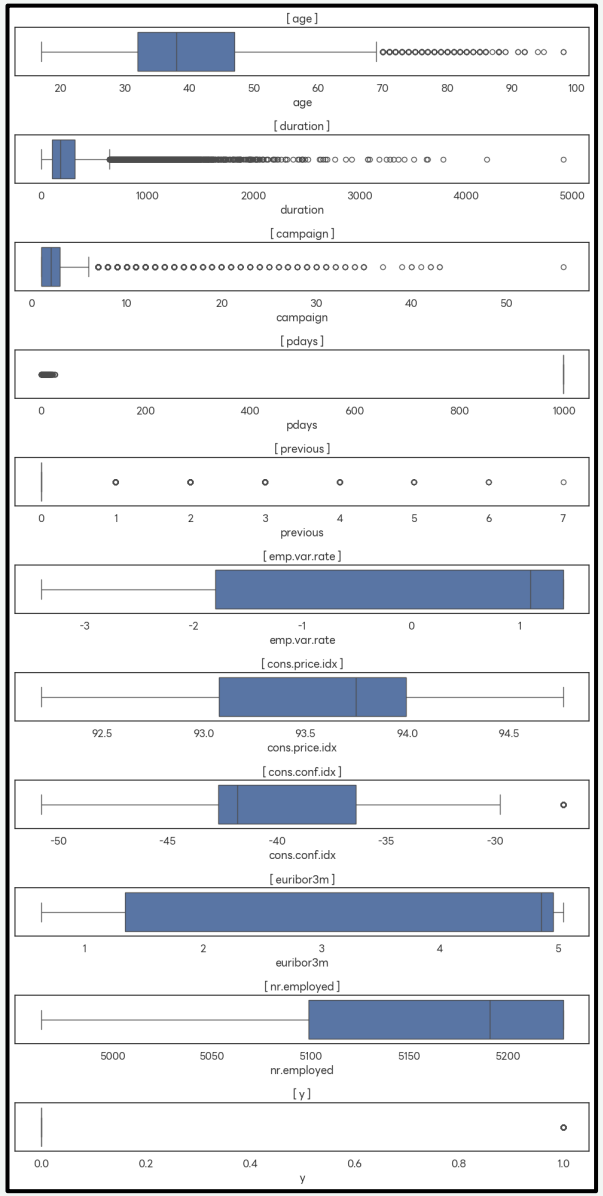
## 이상치

: 이상치 존재

: 하지만 전체 트리 모델 구조에는 큰 영향이 없음.

: 이상치에 강인함.

: 이상치 제거 전/후, 2가지 모두 시도해보았으나, 모델의 성능이 비슷하므로 최종 모델에서는 이상치 제거 하지 않고 모델링 함.



EDA

# 'y' 컬럼 데이터타입 변환 (object -> int64)

```
<class 'pandas.core.frame.DataFrame'>
Index: 41176 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41176 non-null  int64
 1   job             41176 non-null  object
 2   marital         41176 non-null  object
 3   education       41176 non-null  object
 4   default         41176 non-null  object
 5   housing         41176 non-null  object
 6   loan            41176 non-null  object
 7   contact         41176 non-null  object
 8   month           41176 non-null  object
 9   day_of_week     41176 non-null  object
 10  duration        41176 non-null  int64
 11  campaign        41176 non-null  int64
 12  pdays           41176 non-null  int64
 13  previous        41176 non-null  int64
 14  poutcome        41176 non-null  object
 15  emp.var.rate    41176 non-null  float64
 16  cons.price.idx  41176 non-null  float64
 17  cons.conf.idx   41176 non-null  float64
 18  euribor3m       41176 non-null  float64
 19  nr.employed     41176 non-null  float64
 20  y               41176 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.9+ MB
```

```
data["y"].value_counts()
```

```
y
no     36537
yes     4639
Name: count, dtype: int64
```

```
data["y"] = np.where(data["y"] == "yes", 1, 0).astype(int)
```

```
data["y"].value_counts()
```

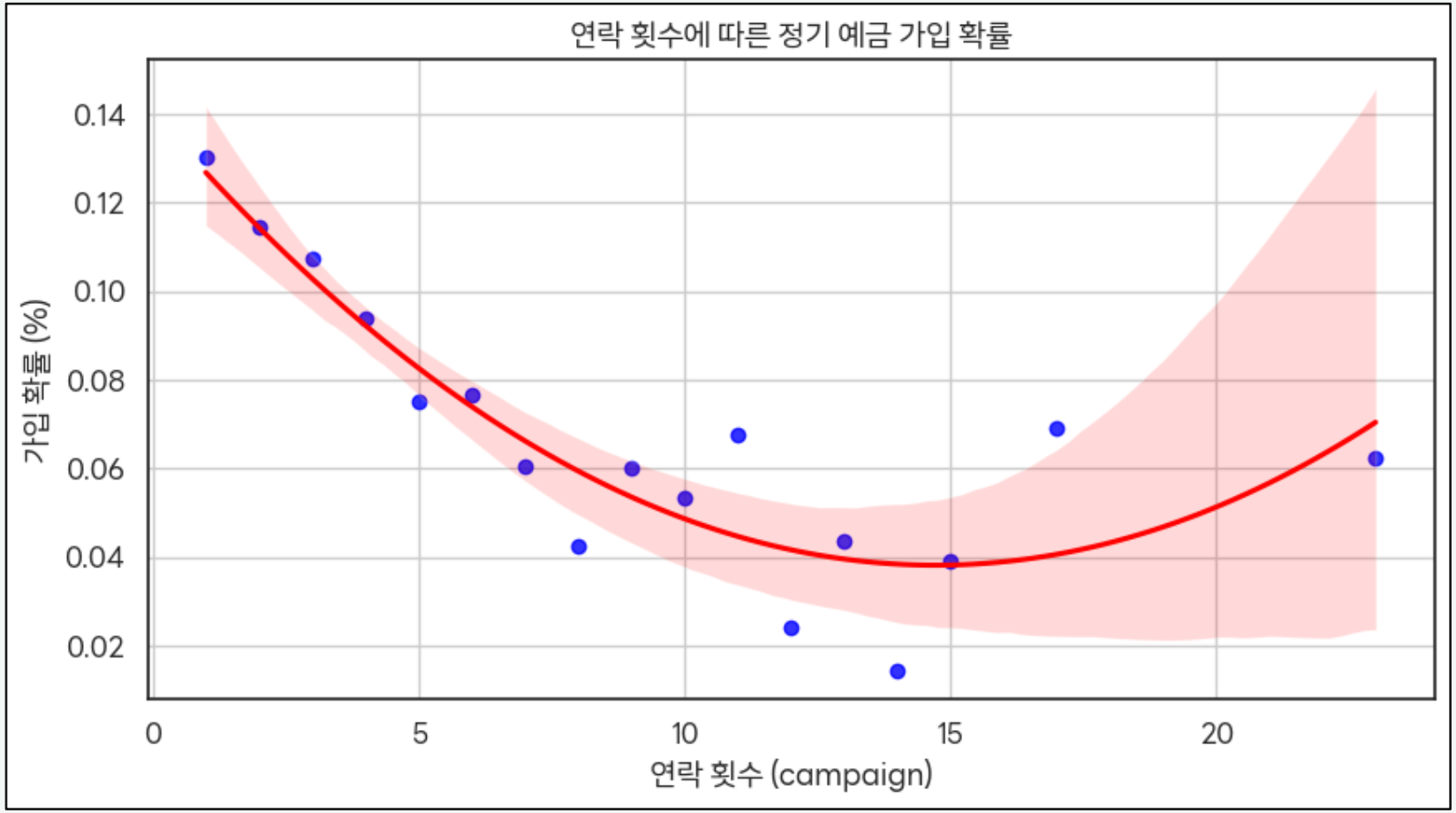
```
y
0    36537
1     4639
Name: count, dtype: int64
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 41176 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41176 non-null  int64
 1   job             41176 non-null  object
 2   marital         41176 non-null  object
 3   education       41176 non-null  object
 4   default         41176 non-null  object
 5   housing         41176 non-null  object
 6   loan            41176 non-null  object
 7   contact         41176 non-null  object
 8   month           41176 non-null  object
 9   day_of_week     41176 non-null  object
 10  duration        41176 non-null  int64
 11  campaign        41176 non-null  int64
 12  pdays           41176 non-null  int64
 13  previous        41176 non-null  int64
 14  poutcome        41176 non-null  object
 15  emp.var.rate    41176 non-null  float64
 16  cons.price.idx  41176 non-null  float64
 17  cons.conf.idx   41176 non-null  float64
 18  euribor3m       41176 non-null  float64
 19  nr.employed     41176 non-null  float64
 20  y               41176 non-null  int64
dtypes: float64(5), int64(6), object(10)
memory usage: 6.9+ MB
```

처음 데이터를 확인해보니 타깃 변수인 'y' 컬럼이 문자열로 되어 있었다.
하지만 'y'는 예금에 가입했는지 여부를 나타내는 이진 값(0 또는 1)만을 가지므로, 문자열 정보가 전혀 필요하지 않다.
따라서 의미상 더 적합한 int64 타입으로 변환하여,
불필요한 문자열 처리도 피할 수 있고, 모델링 시 타입으로 인한 경고나 오류도 줄일 수 있었다

EDA

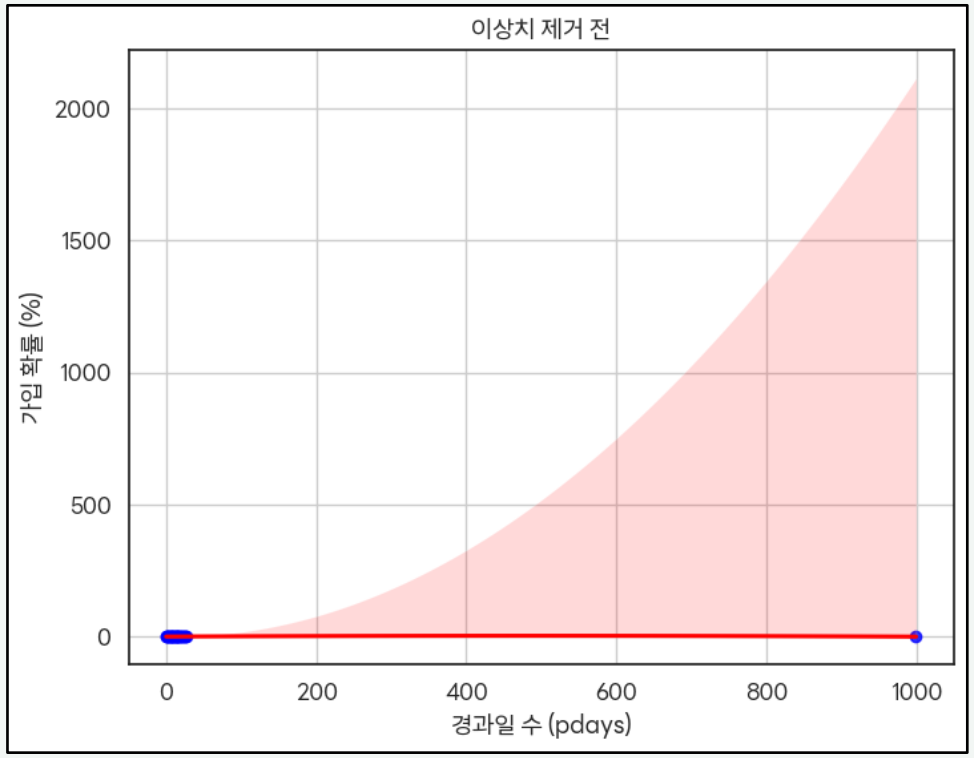
## 연락 횟수('campaign')에 따른 정기 예금 가입('y') 확률

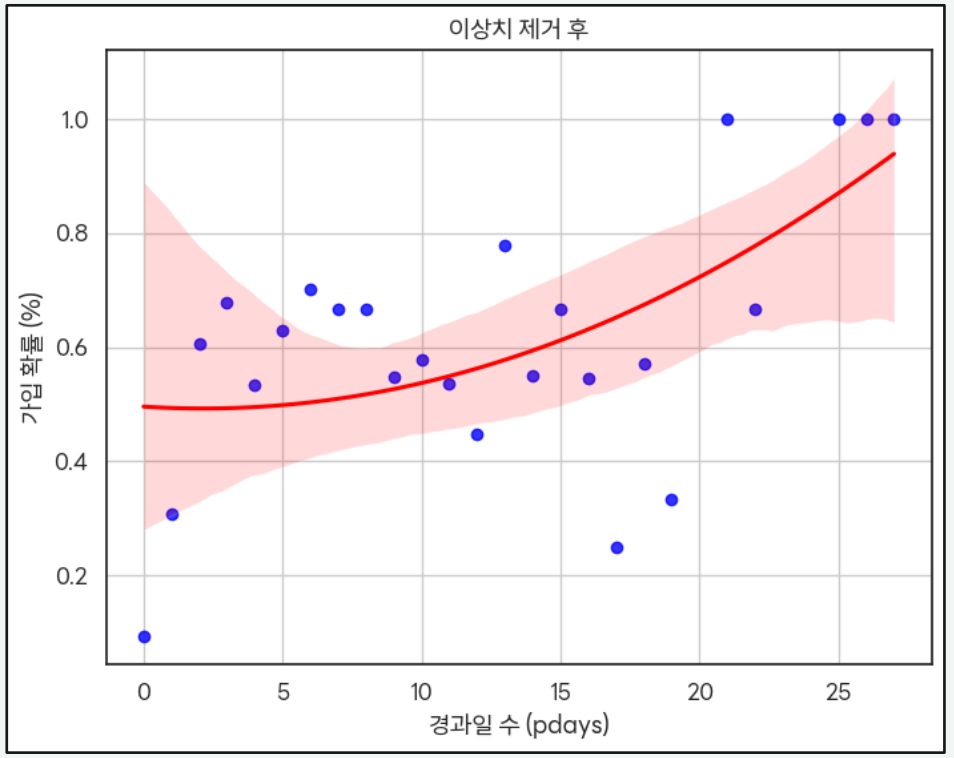



- **하강 추세** : 캠페인 횟수가 증가할수록 가입 확률이 감소하는 경향을 보인다. 이는 소비자 피로도나 관심 감소로 인해 반복적인 캠페인이 효과가 줄어드는 것으로 추측된다.
- **최적의 캠페인 횟수**: 약 5~10 회의 캠페인에서 가입 확률이 상대적으로 높은 경향을 보인다. 이 범위 내에서 최적의 캠페인 전략을 취할 필요가 있다.
- **신뢰 구간**: 빨간색 음영 영역은 신뢰 구간을 나타내며, 이 구간이 좁아지면서 캠페인 횟수가 많아질수록 데이터의 변동성이 줄어드는 것으로 보인다.

EDA

# 경과일 수('pdays')에 따른 정기 예금 가입('y') 확률



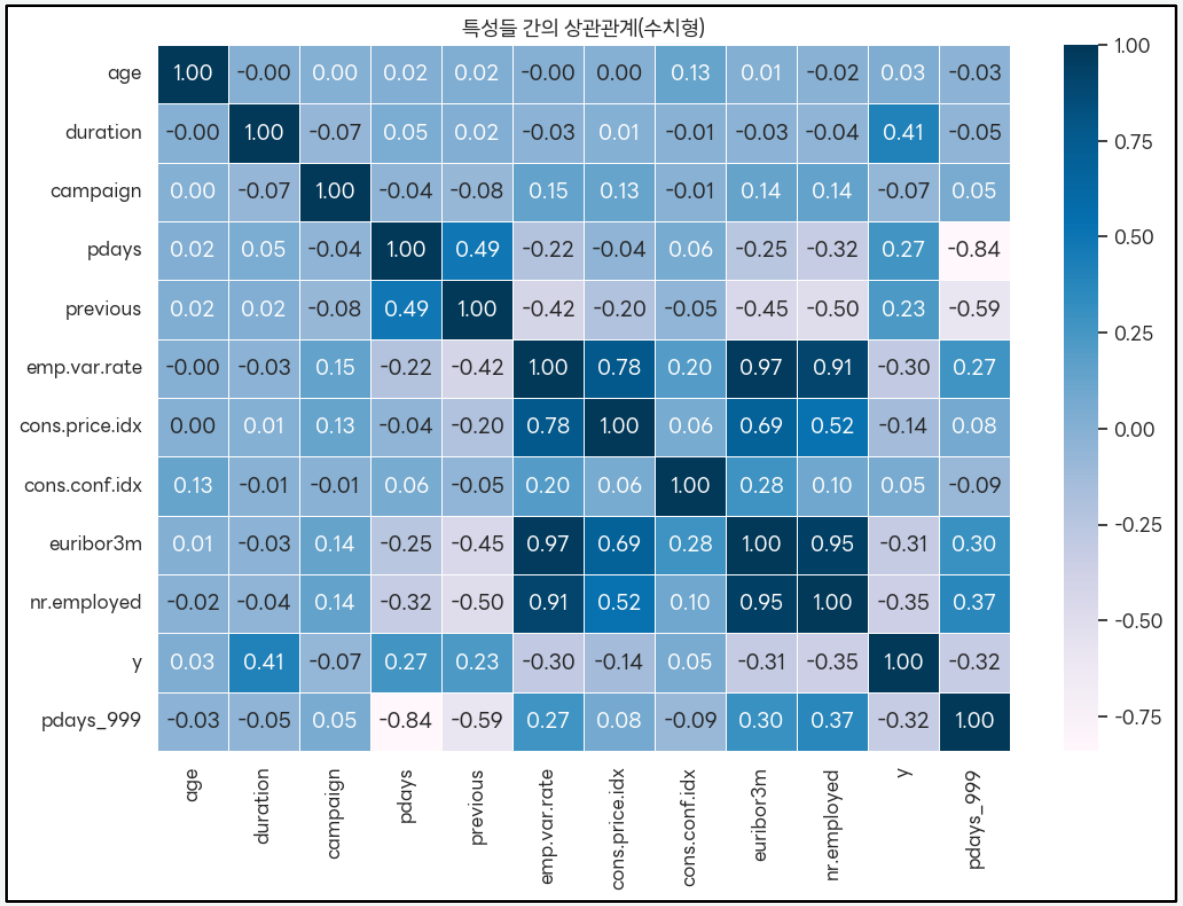

- **len(data[data['pdays'] == 999]) >> 39661**
  : 경과일 수가 999일은 지난 날짜라기 보다는 특별한 의미가 있는 듯하여 'pday_999' 컬럼을 생성하고 'pdays'에는 0으로 변경했다.

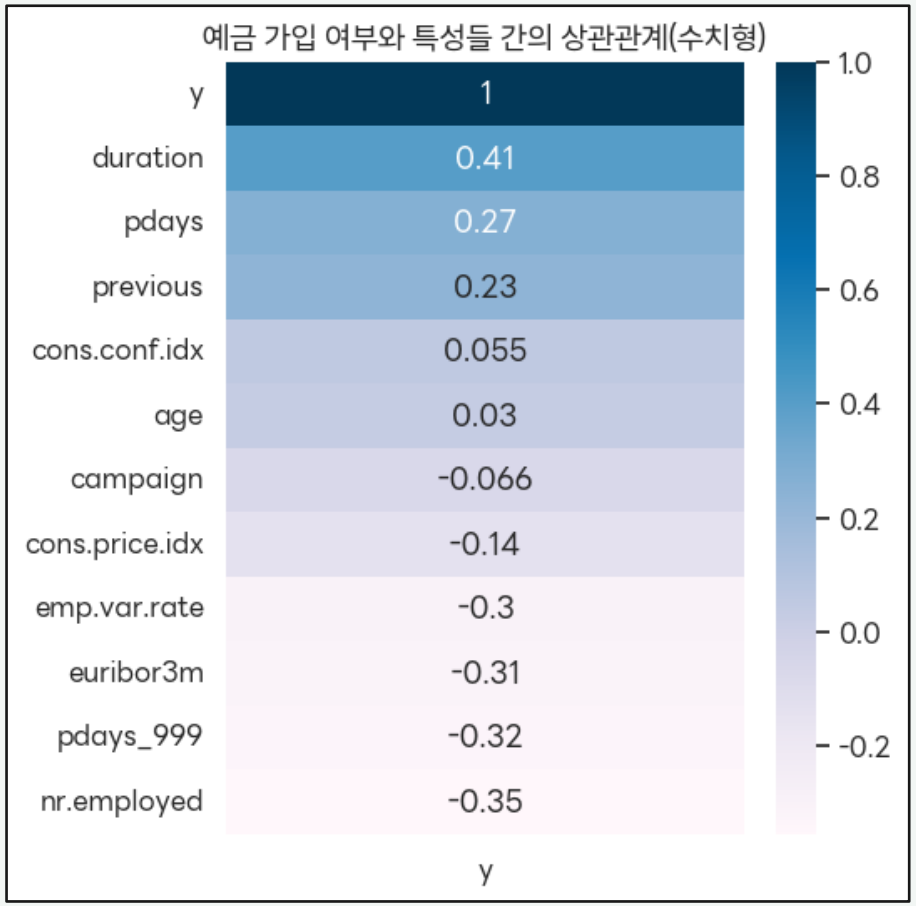
- **경과일 수가 증가함에 따라 가입 확률이 일반적으로 증가하는 것으로 보인다.**
  : 초기에는 가입 확률이 낮지만, 경과일 수가 10일을 넘어서면서 확률이 상승하는 경향이 있다.

- **비선형 패턴**
  : 가입 확률이 도달하는 특정 값에 따라 시간이 지남에 따라 더 빨리 증가할 수 있음을 시사한다.

EDA

# 수치형 특성들 간의 상관관계 / 'y'컬럼과의 상관관계



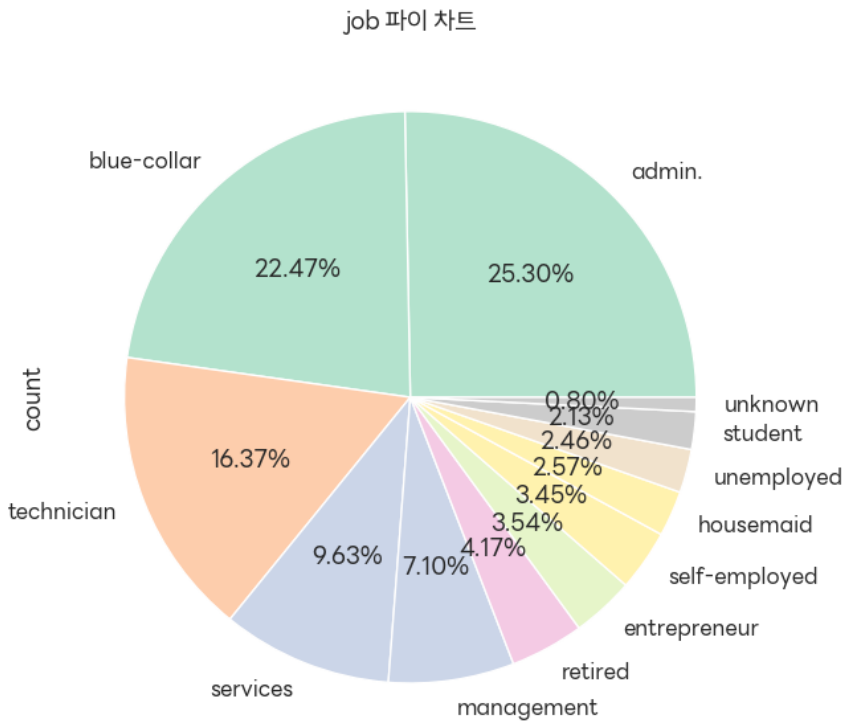



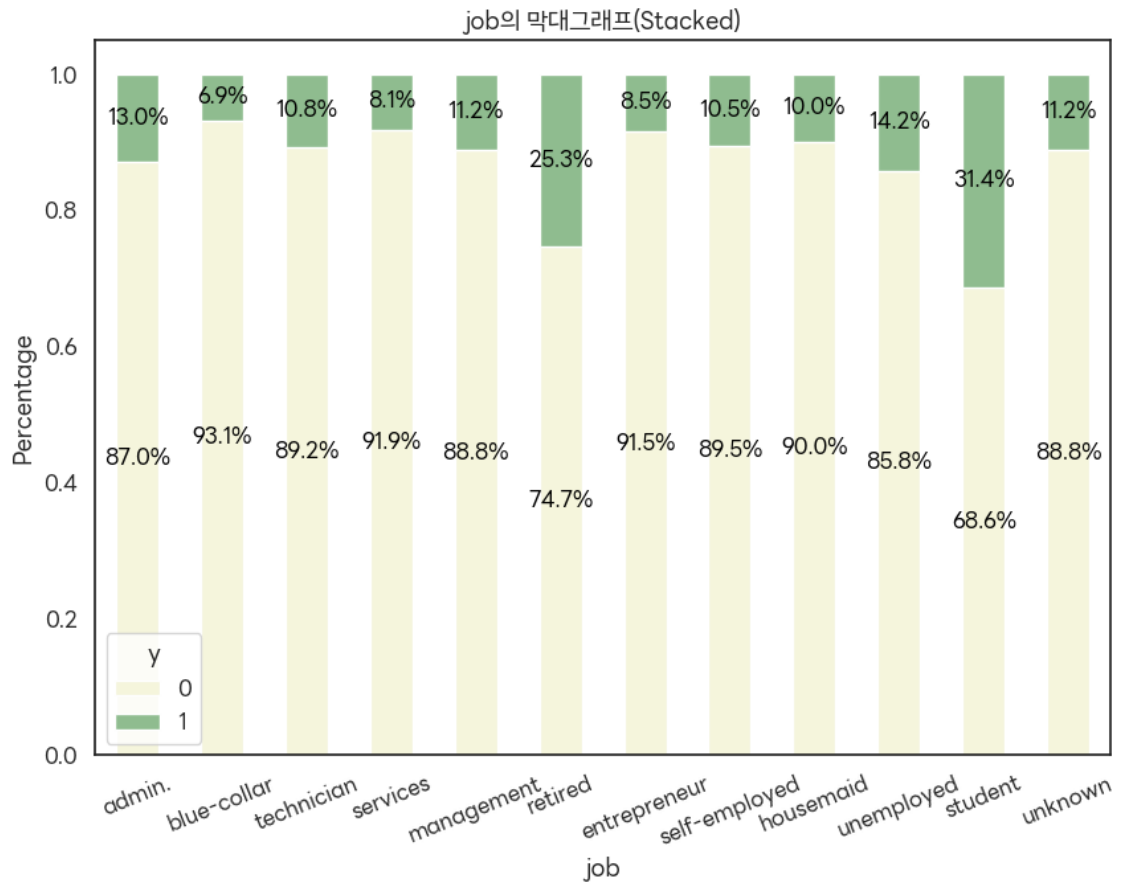

- **'duration'와 'y '** : 두 변수 간의 **상관계수는 0.41**로, 'duration'이 길어질수록 긍정적인 결과(y)에 대한 예측 가능성이 높아진다는 것을 의미한다.
- **'pdays'와 'y '** : 두 변수 간의 **상관계수는 -0.84**로, 'pdays' 값이 작을수록 긍정적인 결과(y)에 더 가까워짐을 나타낸다. 즉, **예전 연락이 짧았던 고객이 재방문의 가능성이 더 높다**는 의미이다.
- **'previous'와 'y ' : 상관계수 -0.59**로, **이전의 연락 횟수가 많을수록 긍정적인 결과가 낮아짐**을 보여준다. 이는 고객이 여러 번 연락을 받았던 경우 결과가 부정적일 수 있음을 나타낸다.
- **'pdays'**와 **'previous ' : 두 변수 간의 상관계수 0.49**로, **이전 연락이 많았던 경우 'pdays' 값이 더 높을 수 있음**을 알 수 있다.
- **'emp.var.rate'와 'cons.price.idx '** : 각각 다른 경제 지표로, **양의 상관관계**를 가지며(상관계수 0.78) **서로 긍정적인 영향**을 미칠 수 있음을 알 수 있고, 경제 지표가 안정적일수록 소비자 신뢰도 또한 높아질 가능성이 있다.
- **'campaign'**의 상관관계는 상대적으로 낮은 편이며**(최대 0.15)** 캠페인의 효과가 다른 변수들에 비해 뚜렷하지 않음을 나타낸다.

EDA

# PI Chart : 컬럼의 카테고리별 비율 / Stack Bar Chart : 예금 가입 여부별 비율

## ‘job’ 컬럼



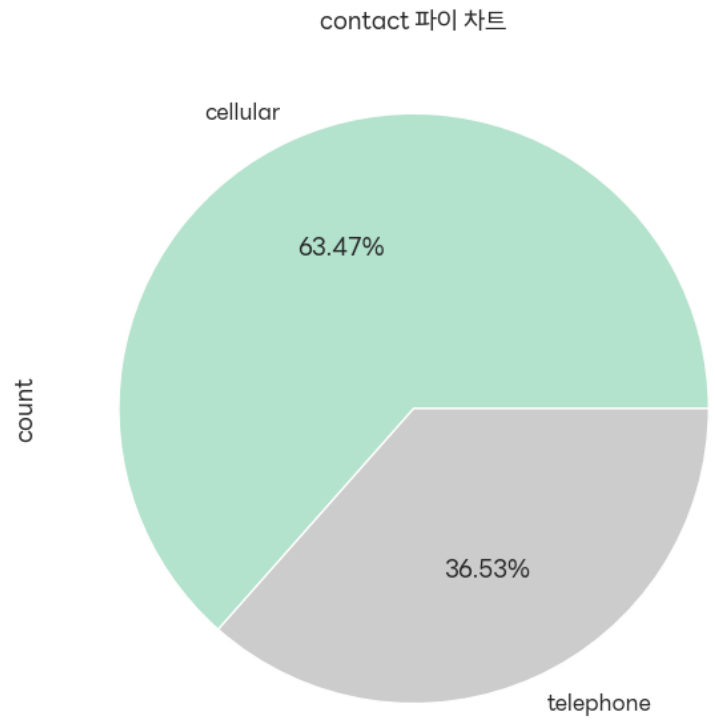



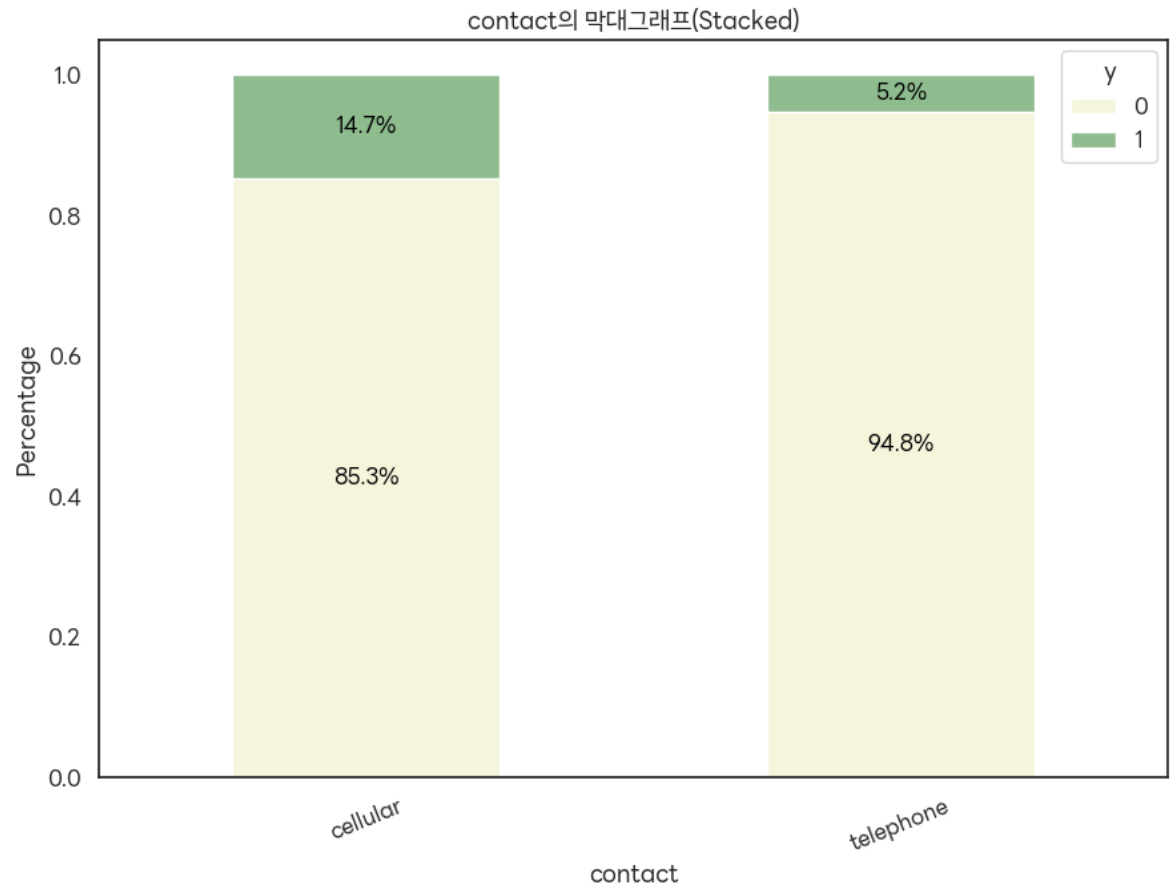

- 소득이 낮거나 불안정한 직업군에서 가입률이 상대적으로 높음 → 예금 상품의 안정성이 어필된 것일 수 있음

EDA

# PI Chart : 컬럼의 카테고리별 비율 / Stack Bar Chart : 예금 가입 여부별 비율

## ‘contact’ 컬럼



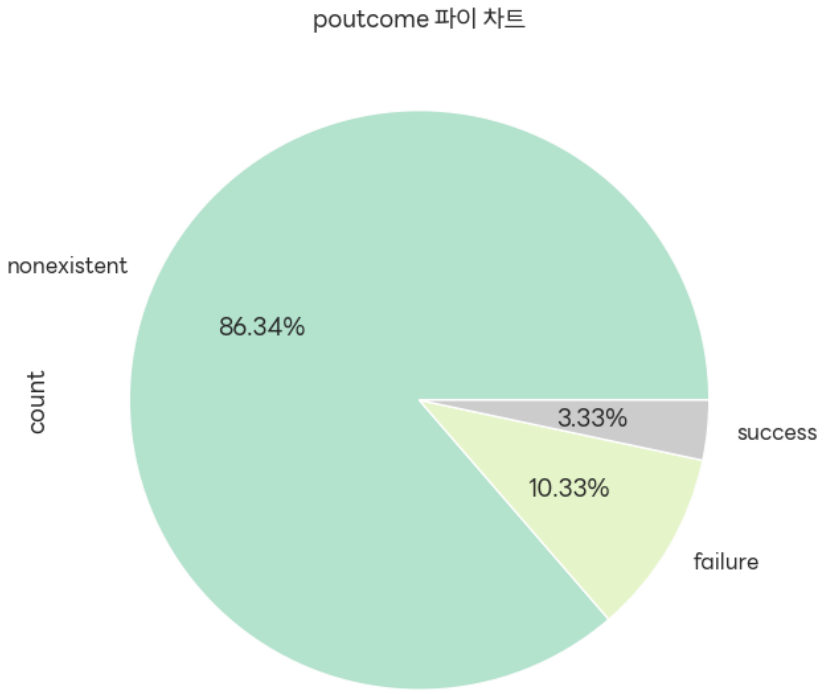



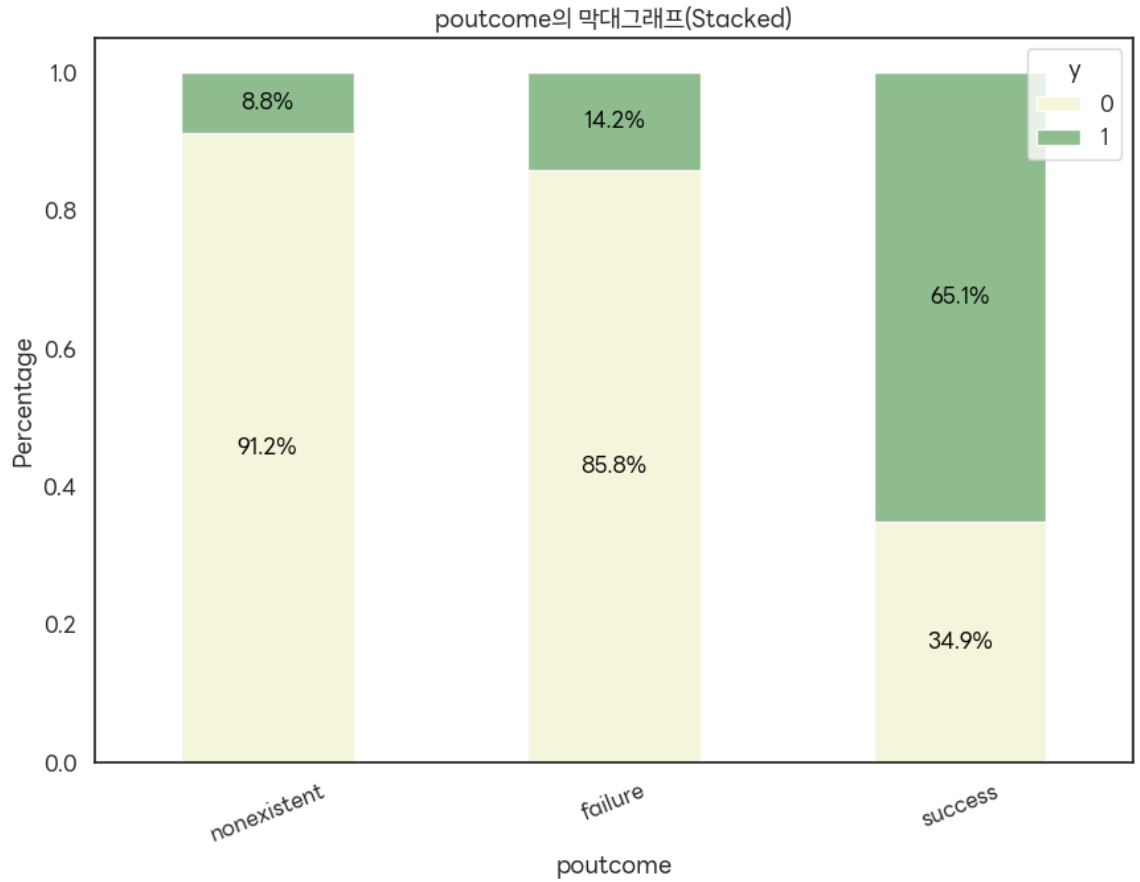

- cellular로 연락한 경우 가입률이 더 높음 → 마케팅 채널로 cellular 활용이 효과적일 수 있음.

EDA

# PI Chart : 컬럼의 카테고리별 비율 / Stack Bar Chart : 예금 가입 여부별 비율

## ‘poutcome’ 컬럼



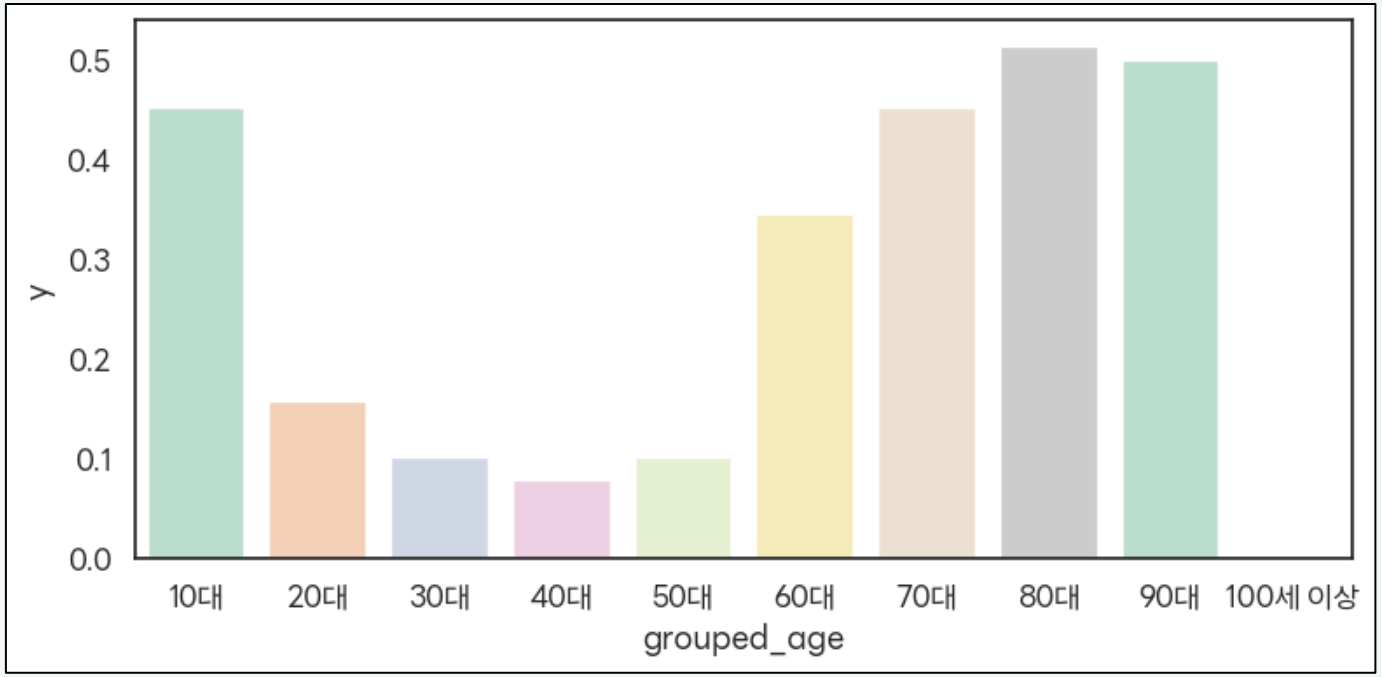

- 과거 마케팅에 성공한 고객에게 재접촉하면 가입 가능성이 높음.

EDA

## 연령대별 예금 가입 여부

```
# 구간과 라벨 설정
bins = [0, 19, 29, 39, 49, 59, 69, 79, 89, 99, float('inf')]
labels = ['10대', '20대', '30대', '40대', '50대', '60대', '70대', '80대', '90대', '100세 이상']

# Categorical dtype으로 age_group 생성
data['grouped_age'] = pd.cut(data['age'], bins=bins, labels=labels)
```

```
plt.figure(figsize= (8,5))
sns.barplot(data, x='grouped_age', y='y', errorbar=None, palette='Pastel2')
```




- 10대, 70대, 80대, 90대에서 가입률이 높음 (약 45~52%)
- 젊은 층(10대)과 고령층(70대 이상)에서 예금 상품에 대한 관심이 높음

→ 각 세대에 맞는 마케팅 전략 필요

# Feature Engineering

## ‘job’ 컬럼 그룹화

```
data['job'].value_counts()
```

```
job
admin.           10419
blue-collar       9253
technician        6739
services          3967
management        2924
retired           1718
entrepreneur      1456
self-employed     1421
housemaid         1060
unemployed        1014
student            875
unknown            330
Name: count, dtype: int64
```

```
# white_collar(사무직) : admin., management,
# blue_collar(현장직) : blue-collar, services, housemaid
# business(사업 관련 직업) : entrepreneur, self-employed
# technician(기술직) 유지
# special(특수한 경우) : student, retired
# etc(기타) : unemployed, unknown

def categorize_job(job):
    if job in ['admin.','management']:
        return 'white_collar'

    elif job in ['blue-collar','service','housemaid']:
        return 'bule_collar'

    elif job in ['entrepreneur', 'self-employed']:
        return 'business'

    elif job in ['technician']:
        return 'technician'

    elif job in ['retired','student']:
        return 'special'

    else:
        return 'etc'

data['grouped_job'] = data['job'].apply(categorize_job)
```

```
data.drop('job', axis=1, inplace=True)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 41176 entries, 0 to 41187
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41176 non-null  int64
 1   marital         41176 non-null  object
 2   education       41176 non-null  object
 3   housing         41176 non-null  object
 4   loan            41176 non-null  object
 5   contact         41176 non-null  object
 6   month           41176 non-null  object
 7   day_of_week     41176 non-null  object
 8   duration        41176 non-null  int64
 9   campaign        41176 non-null  int64
 10  pdays           41176 non-null  int64
 11  previous        41176 non-null  int64
 12  poutcome        41176 non-null  object
 13  emp.var.rate    41176 non-null  float64
 14  cons.price.idx  41176 non-null  float64
 15  cons.conf.idx   41176 non-null  float64
 16  euribor3m       41176 non-null  float64
 17  nr.employed     41176 non-null  float64
 18  y               41176 non-null  int64
 19  grouped_job     41176 non-null  object
dtypes: float64(5), int64(6), object(9)
memory usage: 6.6+ MB
```

카테고리 개수가 많아 모델학습에 불리하게 작용할 수 있기에 그룹화해서 카테고리 개수를 줄였다.

# 데이터 분리

## 모델 학습에 사용할 컬럼 선택 / 입력 변수, 출력 변수 분리 / Train, Test 데이터 분할

```
df = data[['age','grouped_job','housing', 'loan', 'contact','pdays','campaign', 'previous',
           'poutcome', 'cons.price.idx', 'emp.var.rate', 'euribor3m','nr.employed','y']]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 41176 entries, 0 to 41187
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41176 non-null  int64
 1   grouped_job     41176 non-null  object
 2   housing         41176 non-null  object
 3   loan            41176 non-null  object
 4   contact         41176 non-null  object
 5   pdays           41176 non-null  int64
 6   campaign        41176 non-null  int64
 7   previous        41176 non-null  int64
 8   poutcome        41176 non-null  object
 9   cons.price.idx  41176 non-null  float64
 10  emp.var.rate    41176 non-null  float64
 11  euribor3m       41176 non-null  float64
 12  nr.employed     41176 non-null  float64
 13  y               41176 non-null  int64
dtypes: float64(4), int64(5), object(5)
memory usage: 4.7+ MB
```

```
from sklearn.model_selection import train_test_split
```

```
X = df.drop(columns=['y'])
y = df['y']
X.shape, y.shape
```

```
((41176, 13), (41176,))
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((32940, 13), (8236, 13), (32940,), (8236,))
```

# 모델 학습에 사용할 컬럼 선택 / 입력 변수, 출력 변수 분리 / Train, Test 데이터 분할

```
df = data[['age','grouped_job','housing', 'loan', 'contact','pdays','campaign', 'previous',
           'poutcome', 'cons.price.idx', 'emp.var.rate', 'euribor3m','nr.employed','y']]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 41176 entries, 0 to 41187
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41176 non-null  int64
 1   grouped_job     41176 non-null  object
 2   housing         41176 non-null  object
 3   loan            41176 non-null  object
 4   contact         41176 non-null  object
 5   pdays           41176 non-null  int64
 6   campaign        41176 non-null  int64
 7   previous        41176 non-null  int64
 8   poutcome        41176 non-null  object
 9   cons.price.idx  41176 non-null  float64
 10  emp.var.rate    41176 non-null  float64
 11  euribor3m       41176 non-null  float64
 12  nr.employed     41176 non-null  float64
 13  y               41176 non-null  int64
dtypes: float64(4), int64(5), object(5)
memory usage: 4.7+ MB
```

```
from sklearn.model_selection import train_test_split
```

```
X = df.drop(columns=['y'])
y = df['y']
X.shape, y.shape
```

```
((41176, 13), (41176,))
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((32940, 13), (8236, 13), (32940,), (8236,))
```

# 라벨 인코딩

## 범주형 데이터 라벨 인코딩하기 (범주형 ➜ 수치형)

```
from sklearn.preprocessing import LabelEncoder
```

```
# 범주형 컬럼만 선택
cat_cols = df.select_dtypes(include='object').columns
cat_cols
```

```
Index(['grouped_job', 'housing', 'loan', 'contact', 'poutcome'], dtype='object')
```

```
label_encoders = {}
for col in cat_cols:
    le = LabelEncoder()
    X_train[col] = le.fit_transform(X_train[col])
    X_test[col] = le.transform(X_test[col])
    label_encoders[col] = le
```

```
# 디코딩
for col in cat_cols:
    print(f"{col} 매핑: {dict(zip(label_encoders[col].classes_, label_encoders[col].transform(label_encoders[col].classes_)))}")
```

```
grouped_job 매핑: {'bule_collar': 0, 'business': 1, 'etc': 2, 'special': 3, 'technician': 4, 'white_collar': 5}
housing 매핑: {'no': 0, 'unknown': 1, 'yes': 2}
loan 매핑: {'no': 0, 'unknown': 1, 'yes': 2}
contact 매핑: {'cellular': 0, 'telephone': 1}
poutcome 매핑: {'failure': 0, 'nonexistent': 1, 'success': 2}
```

# Decision Tree

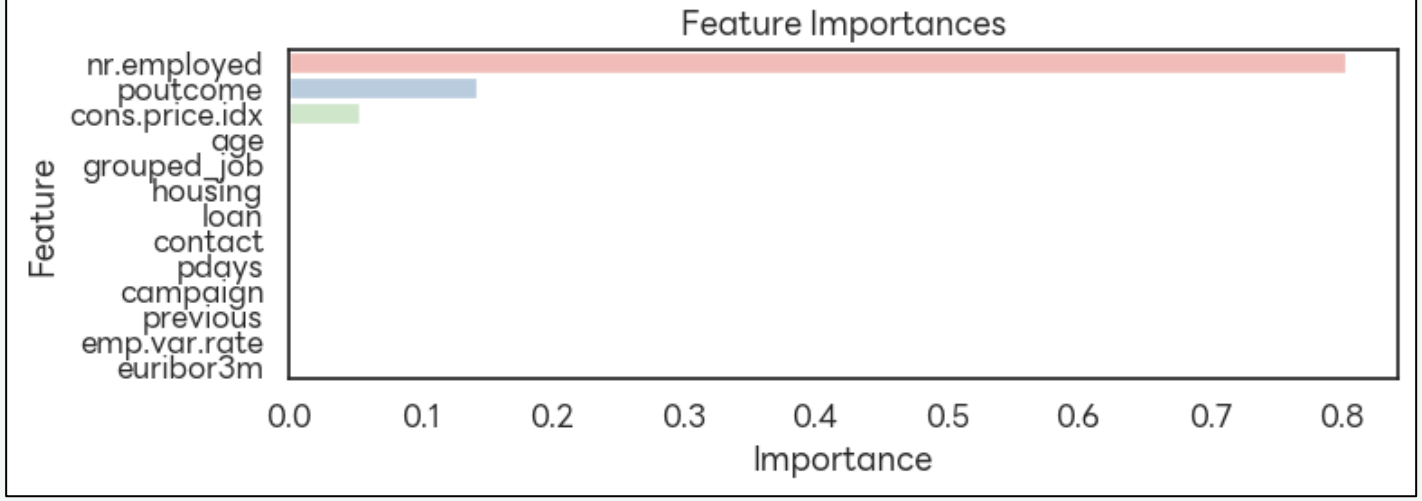
## 결정 트리 모델링과 특성 중요도 시각화

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.99 | 0.94 | 7265 |
| 1 | 0.73 | 0.16 | 0.26 | 971 |
| accuracy | | | 0.89 | 8236 |
| macro avg | 0.81 | 0.58 | 0.60 | 8236 |
| weighted avg | 0.88 | 0.89 | 0.86 | 8236 |

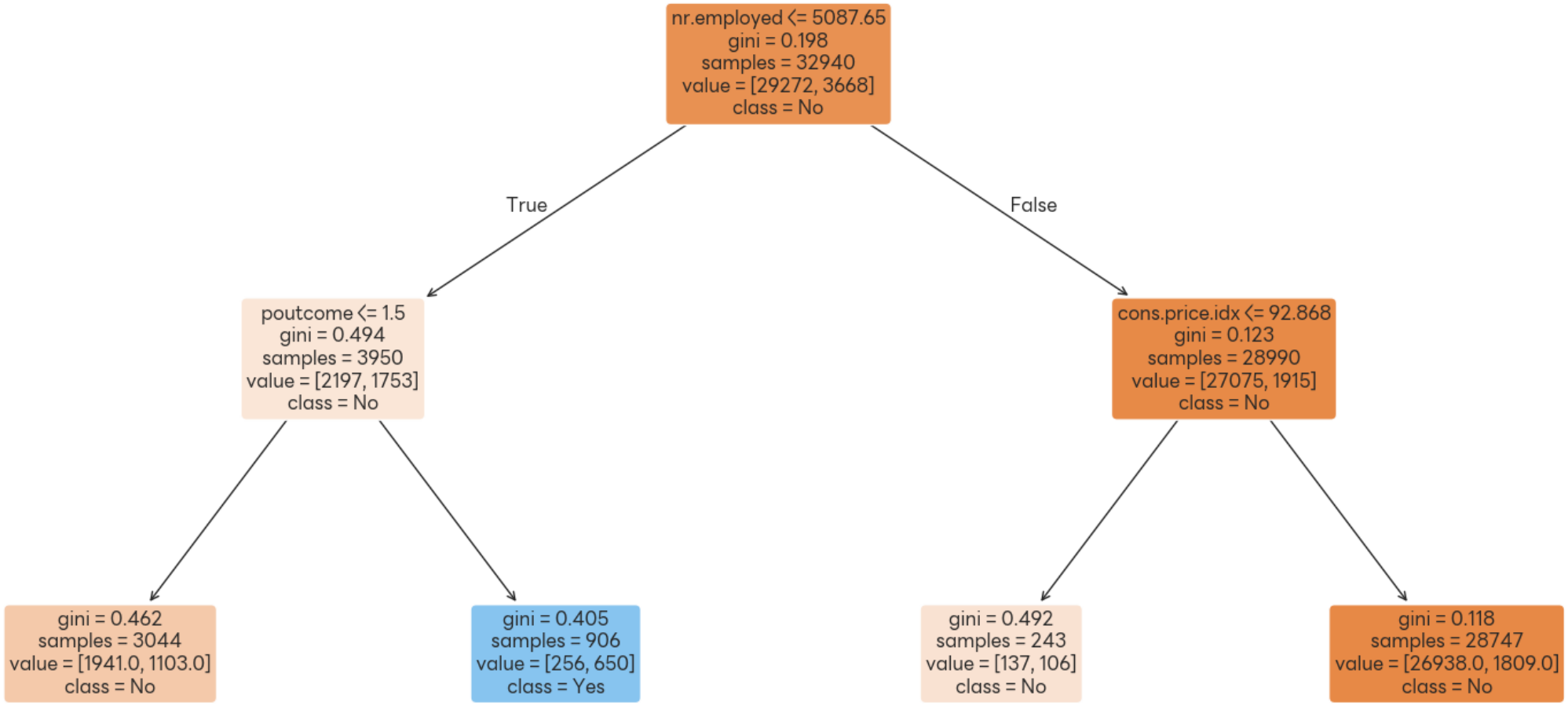
```
(array([0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.14253138, 0.05436234,
        0.        , 0.        , 0.80310628]),
 Index(['age', 'grouped_job', 'housing', 'loan', 'contact', 'pdays', 'campaign',
        'previous', 'poutcome', 'cons.price.idx', 'emp.var.rate', 'euribor3m',
        'nr.employed'],
       dtype='object'))
```

- 모델이 하나의 변수(nr.employed)에 **과도하게 의존.**
- euribor3m, emp.var.rate, previous, campaign 등은 거의 쓰이지 않았고,
  이들을 **제거해도 성능 변화가 없을 수 있음**.
- **의사결정 트리의 단점**이 드러남.
  → 하나의 피처에 지나치게 의존해 **분기가 단순**해짐.
- 가장 중요한 feature = **nr.employed(고용된 사람 수)**
  → (0.8) : 모델이 대부분 이 변수 하나만으로 예측.
- 그 다음으로 중요한 변수는 **poutcome (과거 캠페인 결과),**
  **cons.price.idx(소비자 물가 지수)**이고, 비중이 0.1 이하로 작음.

- 나머지 변수들은 0에 가깝고, **거의 의사 결정 트리에서 사용 X.**



# 결정 트리 시각화

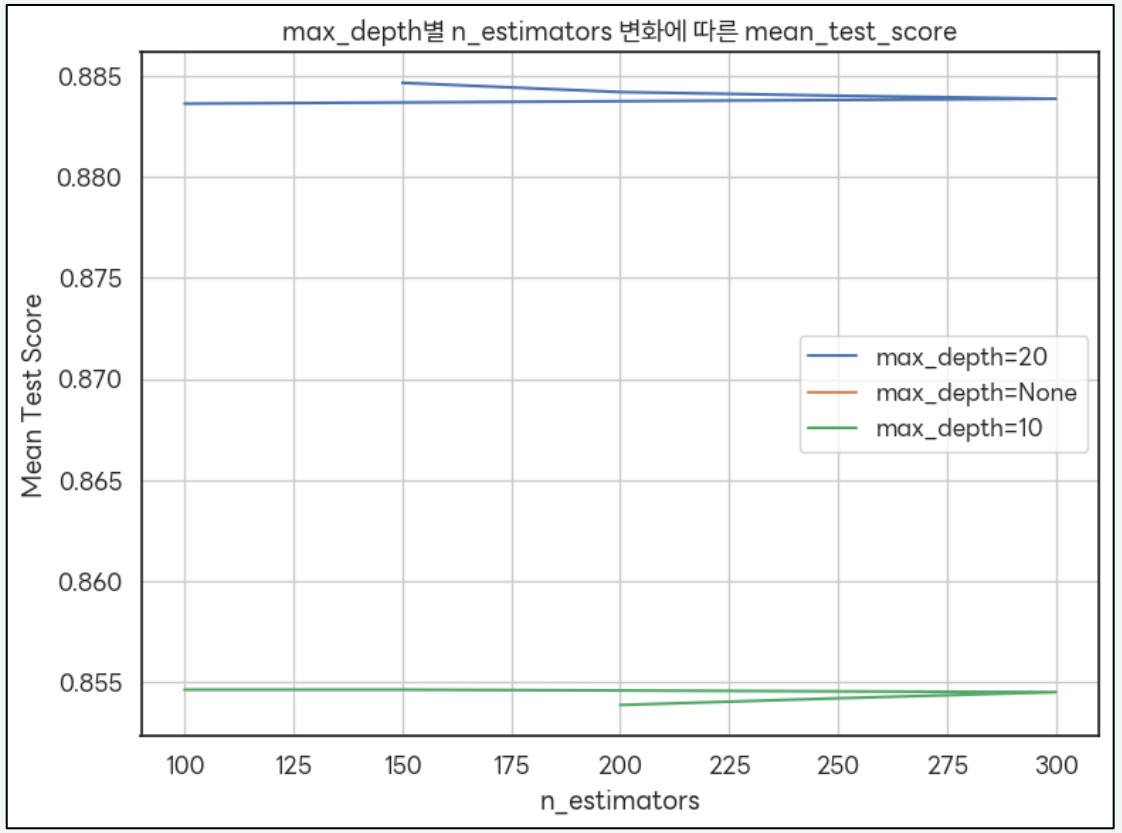

nr.employed <= 5087.65
gini = 0.198
samples = 32940
value = [29272, 3668]
class = No
True
False
poutcome <= 1.5
gini = 0.494
samples = 3950
value = [2197, 1753]
class = No
cons.price.idx <= 92.868
gini = 0.123
samples = 28990
value = [27075, 1915]
class = No
gini = 0.462
samples = 3044
value = [1941.0, 1103.0]
class = No
gini = 0.405
samples = 906
value = [256, 650]
class = Yes
gini = 0.492
samples = 243
value = [137, 106]
class = No
gini = 0.118
samples = 28747
value = [26938.0, 1809.0]
class = No

# Random Forest

## 랜덤 포레스트 모델링과 특성 중요도 시각화

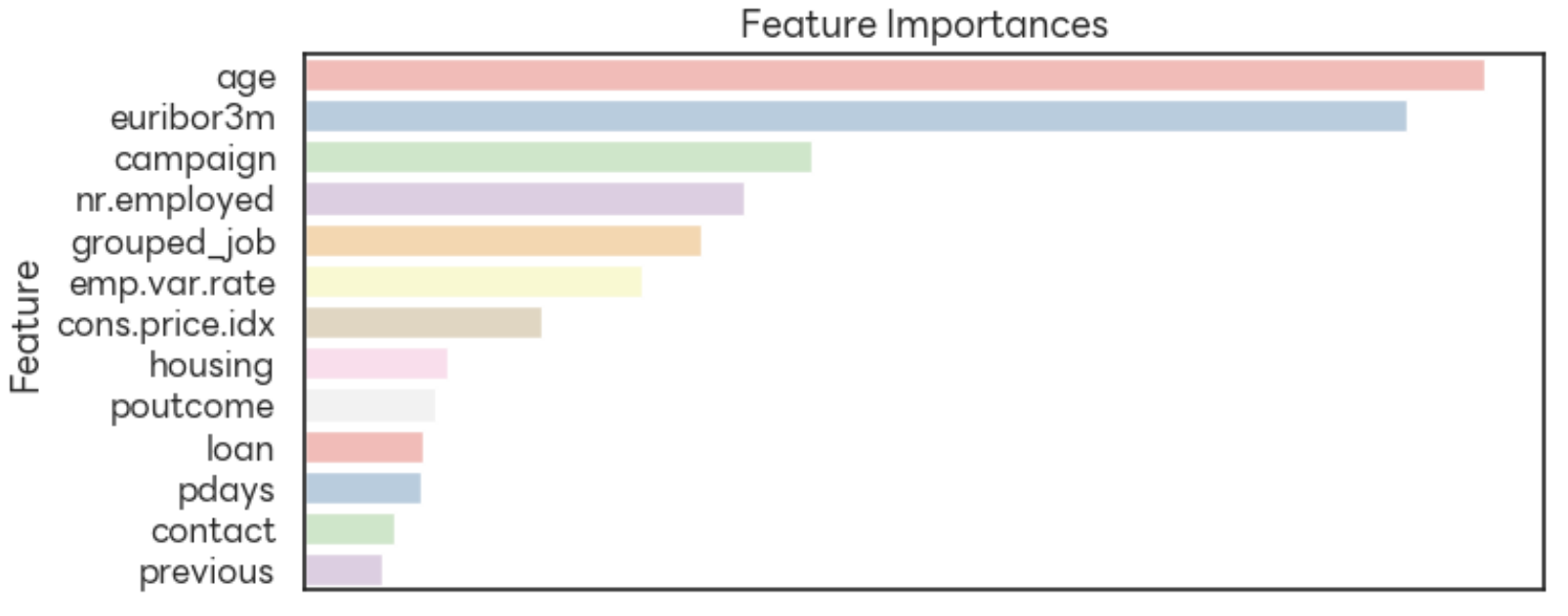
```
최적 파라미터: {'class_weight': 'balanced', 'max_depth': 20, 'n_estimators': 150}
최고 교차검증 점수: 0.8847
테스트 정확도: 0.8763

Classification Report:

              precision    recall  f1-score   support

           0     0.9132    0.9500    0.9313      7265
           1     0.4646    0.3244    0.3820       971

    accuracy                         0.8763      8236
   macro avg     0.6889    0.6372    0.6567      8236
weighted avg     0.8603    0.8763    0.8665      8236
```

| | param_n_estimators | param_max_depth | mean_test_score | rank_test_score |
|---|---|---|---|---|
| **11** | 150 | 20 | 0.884669 | 1 |
| **12** | 200 | 20 | 0.884214 | 2 |
| **13** | 250 | 20 | 0.884032 | 3 |
| **14** | 300 | 20 | 0.883880 | 4 |
| **10** | 100 | 20 | 0.883637 | 5 |



# Random Forest 모델에서의 특성 중요도

```
(array([0.24158799, 0.08140226, 0.02943513, 0.02445145, 0.01843864,
        0.02413919, 0.1037706 , 0.0162146 , 0.02676174, 0.04864956,
        0.06932082, 0.22576968, 0.09005834]),
 Index(['age', 'grouped_job', 'housing', 'loan', 'contact', 'pdays', 'campaign',
        'previous', 'poutcome', 'cons.price.idx', 'emp.var.rate', 'euribor3m',
        'nr.employed'],
       dtype='object'))
```




**('age')**

: 나이가 많고 적음이 예금 가입 여부에 가장 큰 영향을 주는 것으로 보여짐.

**('euribor3m')**

: 유럽 기준금리로 경제적 요인이 큰 영향을 끼치는 것으로 보여줌.

**('campaign')**

: 마케팅 접촉 횟수가 중요 요인인 것 같음.

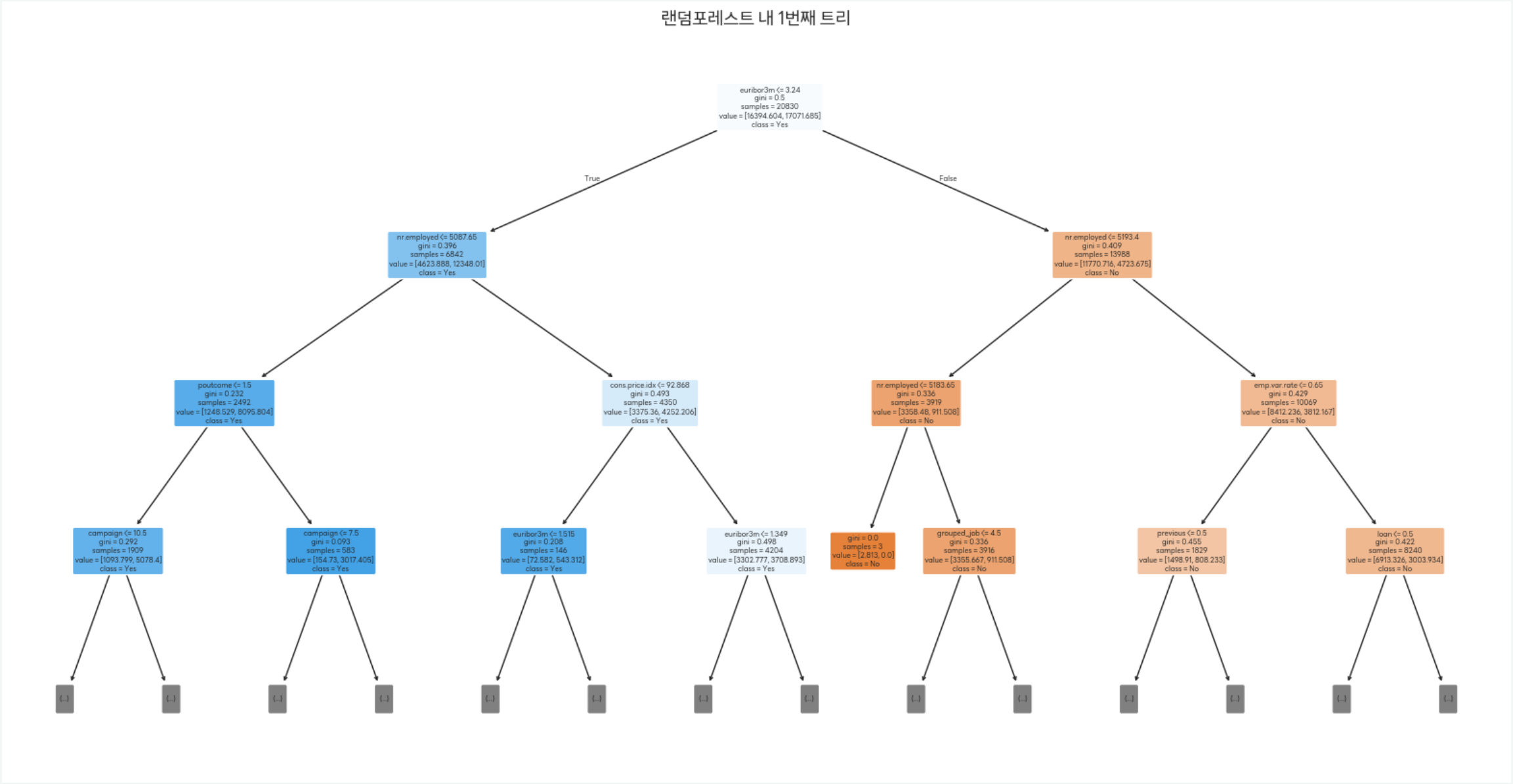
**('grouped_job')**

: 직업군에 따른 행동 패턴 차이가 있을 수 있음.

---

- 개인 특성 (age, job)과 경제적 지표 (euribor3m, nr.employed, emp.var.rate), 그리고 마케팅 전략 (campaign)에 민감하게 반응함.

# 랜덤 포레스트 내의 1번째 트리



# XGBoost

## XGBoost 모델링과 특성 중요도 시각화

```
최적 파라미터: {'max_depth': 3, 'min_child_weight': 5, 'n_estimators': 250, 'reg_alpha': 0.1, 'reg_lambda': 3, 'scale_pos_weigh
t': 7}
최고 교차검증 점수: 0.8477
테스트 정확도: 0.8458
```

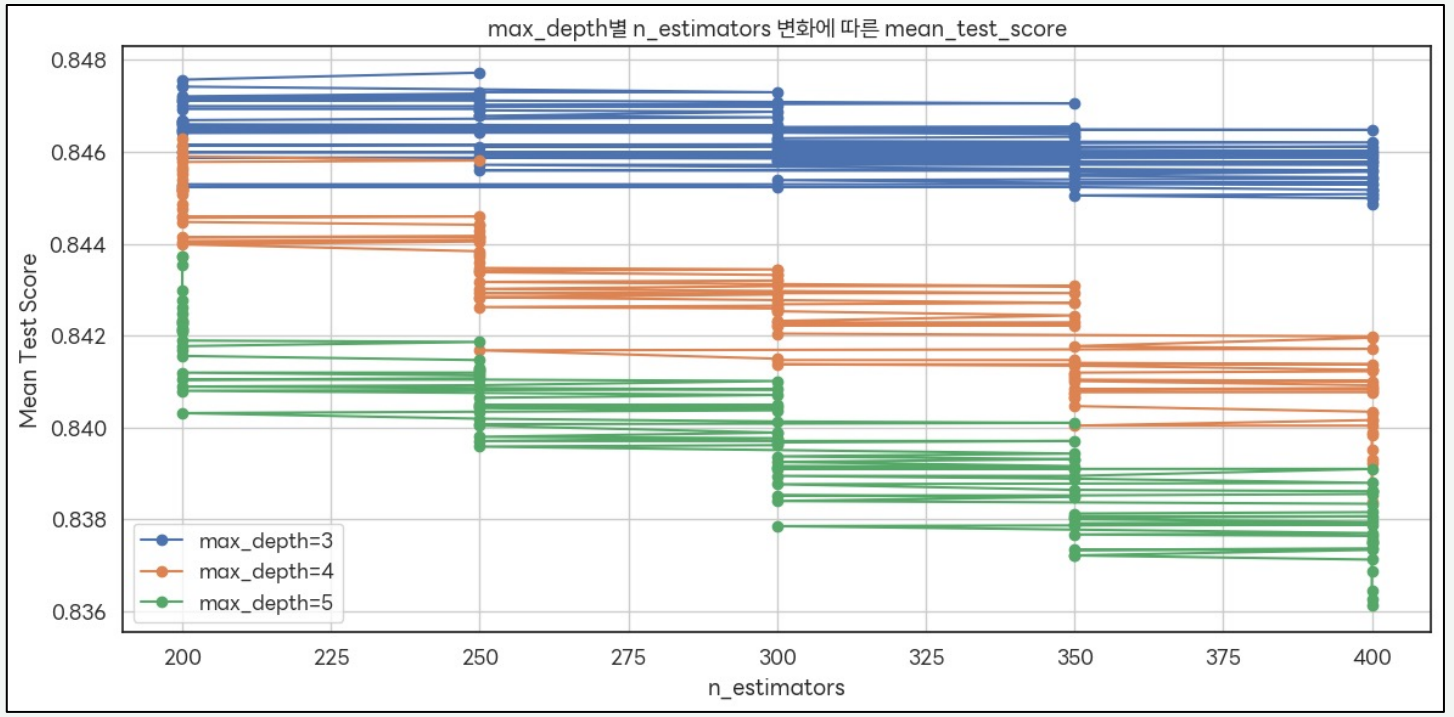
```
Classification Report:

              precision    recall  f1-score   support

           0     0.9430    0.8783    0.9095      7265
           1     0.3982    0.6025    0.4795       971

    accuracy                         0.8458      8236
   macro avg     0.6706    0.7404    0.6945      8236
weighted avg     0.8787    0.8458    0.8588      8236
```

**(max_depth = 3)**

: 가장 높은 mean_test_score를 보이며, 특히 n_estimators가 200-300 사이일 때 성능이 좋음.

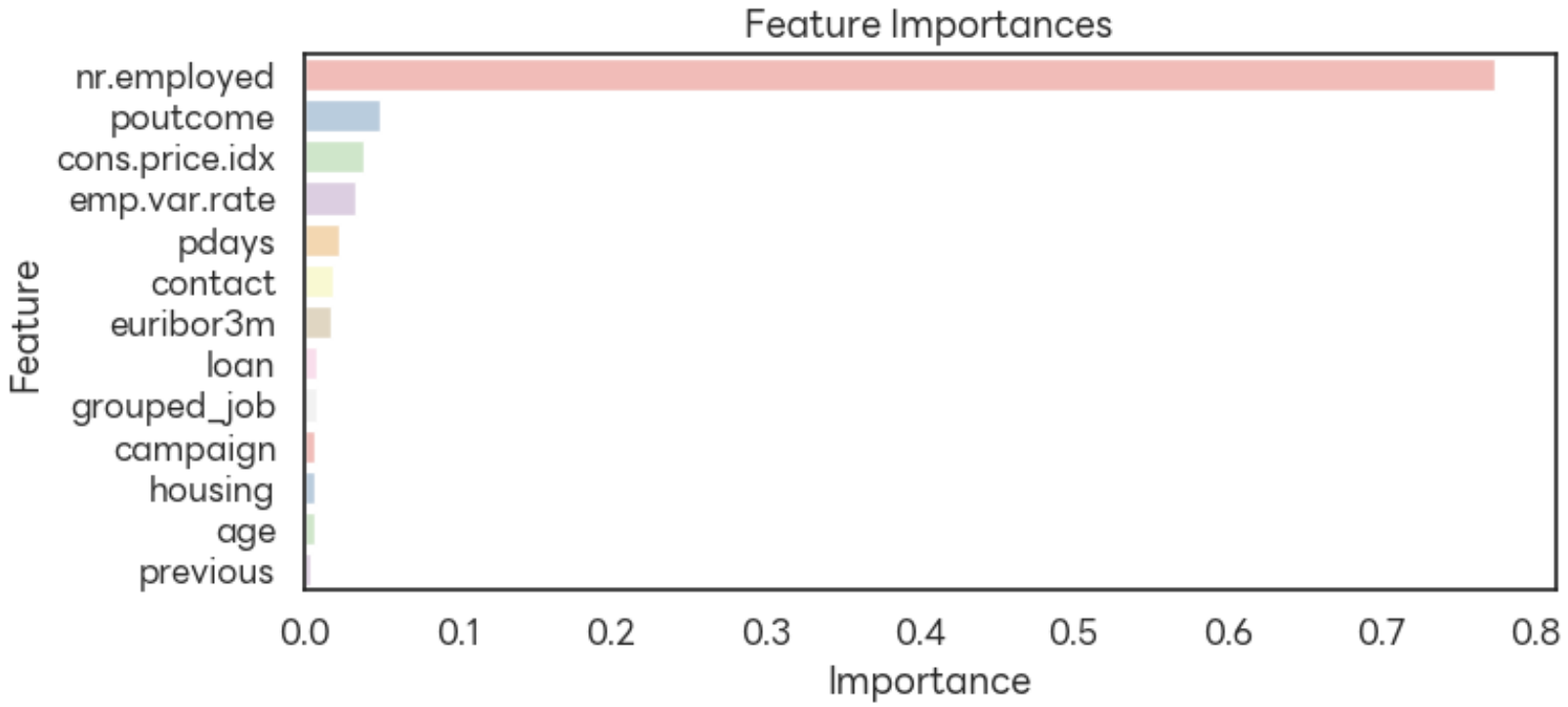
**(최적의 모델 선택)**

- max_depth=3과 max_depth=4의 경우가 유리하지만, 복잡도와 안정성을 고려할 때 max_depth=3가 상대적으로 더 좋은 선택으로 보임.



# XGBoost모델에서의 특성 중요도

```
(array([0.00667099, 0.00785508, 0.00723684, 0.00882173, 0.01877515,
        0.023552  , 0.00774257, 0.00492985, 0.04908089, 0.03898495,
        0.03375421, 0.01842375, 0.774172  ], dtype=float32),
 Index(['age', 'grouped_job', 'housing', 'loan', 'contact', 'pdays', 'campaign',
        'previous', 'poutcome', 'cons.price.idx', 'emp.var.rate', 'euribor3m',
        'nr.employed'],
       dtype='object'))
```




**(nr.employed)**

- nr.employed **변수 하나만**으로도 **모델 예측의 대부분을 설명**하고 있음 (0.77)
- **전체 고용 수치(nr.employed)가 고객의 예금 가입 여부**와 **매우 밀접하게 연관**되어 있다는 것을 의미.

**(모델의 학습 편향 가능성이 의심됨)**

- nr.employed를 제외한 특성들의 중요도가 미미함.

# 결론

| 모델 | 정확도 | Precision | Recall | F1-score | 평가 요약 |
|---|---|---|---|---|---|
| Decision Tree | 0.89 | 0.73 | 0.16 | 0.26 | 과적합 의심<br>실질적 성능이 떨어짐 |
| Random Forest | 0.8763 | 0.4646 | 0.4646 | 0.3820 | 보수적 예측<br>Precision 우위 |
| XGBoost | 0.8458 | 0.3982 | 0.6025 | 0.4795 | 가장 균형적인 성능<br>실제 가입자를 잘 포착함 |

1. F1-score가 가장 높아 모델의 전반적 성능이 가장 우수 / 실제 예금 가입자를 최대한 포착하고 싶은 경우
→ XGBoost가 가장 적합

2. 반대로 예측 정확도와 해석 가능성을 중시하고, FP(False Positive)을 줄이는 게 중요한 경우
→ Random Forest 선택 가능

3. Decision Tree는 간단한 모델 해석에는 유용할 수 있으나, 성능상 선택 우선순위는 낮음

# 비즈니스 모델

## (특정 연령대에 집중된 마케팅 전략)

- age 변수가 가장 중요한 특성으로 나타남
  → 나이에 따라 예금 가입 성향이 크게 달라짐.

- 고령층(60세 이상)에게는 안정성과 고정 수익을 강조하는 상품을 제공.
- EX) "안정적 수익의 정기예금", "노후 대비 플랜"

- 젊은 층(20~30대)에게는 유연하고 유동성 높은 예금 상품 제공.
  - EX) "단기 수시입출금 혜택", "목돈 모으기 챌린지 예금"

## (금리(유럽 기준금리 euribor3m)에 따른 프로모션)

- euribor3m은 경제 상황을 반영하는 변수로, 금리가 높거나 낮을 때 가입 여부가 달라짐.

- **금리 상승 시기**: 고객이 예금에 관심을 가질 확률이 높음
  → 이 시점에 마케팅 집중.
- EX) “금리 인상 기념, 고금리 특판 예금 한정 출시!”