Bank Marketing
참여 여부 예측

2분반 10조

역 할

- **배지원 (팀장 & 평가)**
  파일 제출 및 제출 파일 평가
- **김은별 (발표)**
  발표 내용 숙지 및 발표
- **이나경 (데이터 전처리, 모델링, 평가)**
  LabelEncoder() 전처리, LightBGM 모델링 및 평가
- **장가람 (EDA, 데이터 전처리, 모델링, 평가, PPT 작업, 코드 정리)**
  데이터 EDA, 변수 스케일링 및 변수 생성, LightBGM 모델링 및 평가, 발표 자료 준비, 코드 정리

목 차

1. 데이터 탐색 (EDA) - 데이터 구조 파악

```
df_trn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31647 entries, 0 to 31646
Data columns (total 18 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   ID         31647 non-null  object
 1   age        31647 non-null  int64
 2   job        31647 non-null  object
 3   marital    31647 non-null  object
 4   education  31647 non-null  object
 5   default    31647 non-null  object
 6   balance    31647 non-null  int64
 7   housing    31647 non-null  object
 8   loan       31647 non-null  object
 9   contact    31647 non-null  object
 10  day        31647 non-null  int64
 11  month      31647 non-null  object
 12  duration   31647 non-null  int64
 13  campaign   31647 non-null  int64
 14  pdays      31647 non-null  int64
 15  previous   31647 non-null  int64
 16  poutcome   31647 non-null  object
 17  label      31647 non-null  int64
dtypes: int64(8), object(10)
memory usage: 4.3+ MB
```

```
df_tst.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13564 entries, 0 to 13563
Data columns (total 18 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   ID         13564 non-null  object
 1   age        13564 non-null  int64
 2   job        13564 non-null  object
 3   marital    13564 non-null  object
 4   education  13564 non-null  object
 5   default    13564 non-null  object
 6   balance    13564 non-null  int64
 7   housing    13564 non-null  object
 8   loan       13564 non-null  object
 9   contact    13564 non-null  object
 10  day        13564 non-null  int64
 11  month      13564 non-null  object
 12  duration   13564 non-null  int64
 13  campaign   13564 non-null  int64
 14  pdays      13564 non-null  int64
 15  previous   13564 non-null  int64
 16  poutcome   13564 non-null  object
 17  label      0 non-null      float64
dtypes: float64(1), int64(7), object(10)
memory usage: 1.9+ MB
```

```
df_trn.duplicated().sum()
```

```
0
```

```
df_tst.duplicated().sum()
```

```
0
```

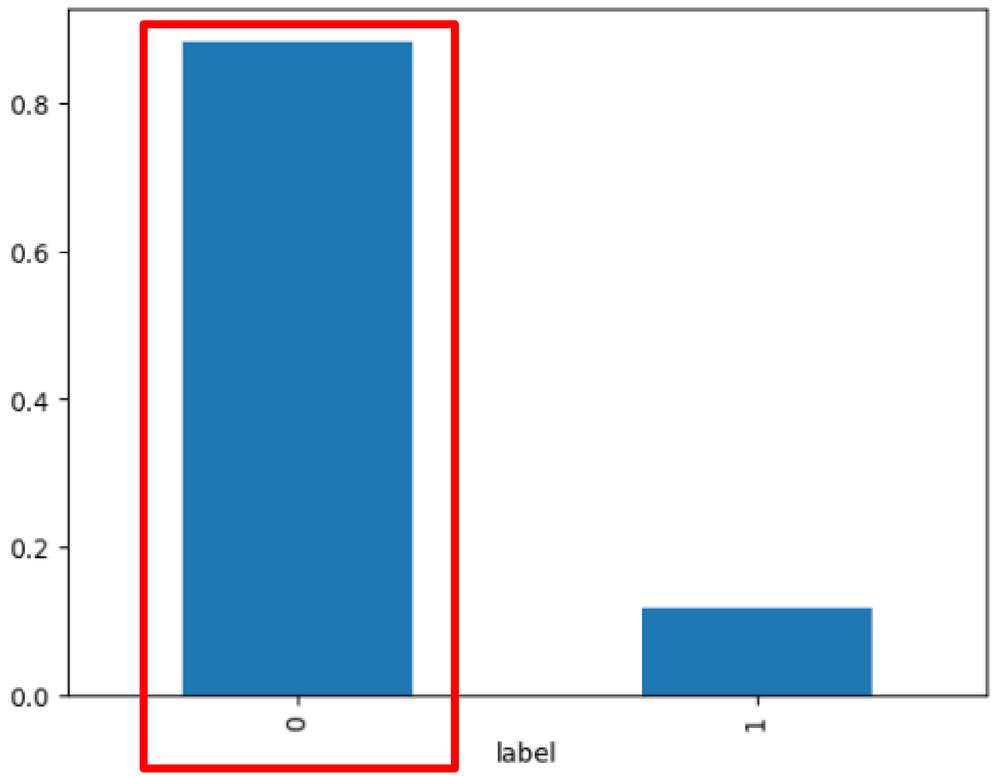
총 16개의 독립변수 (ID, Label 제외)

- 결측치 없음
- 중복 데이터 없음

1. 데이터 탐색 (EDA) - 타겟 변수 분포 확인

```
df_trn['label'].value_counts(normalize=True).plot(kind='bar')
```

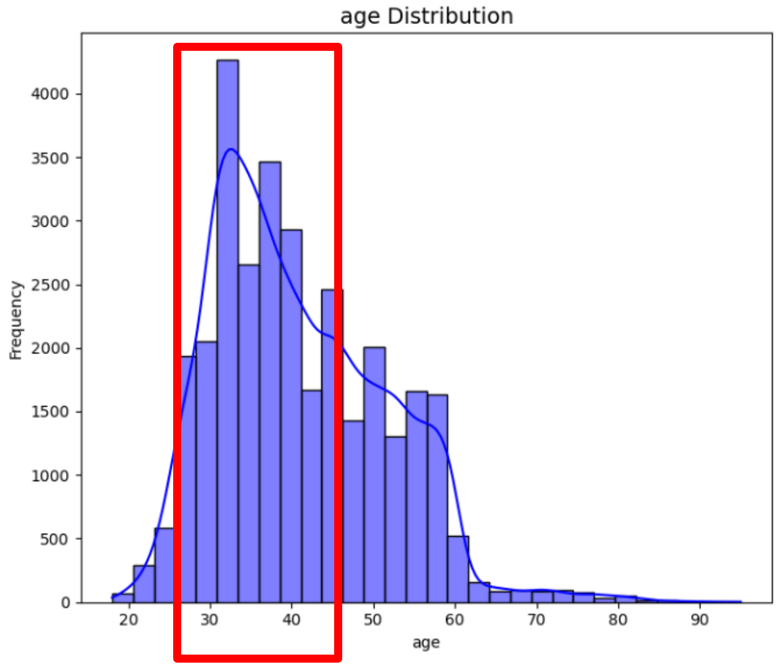
```
<Axes: xlabel='label'>
```

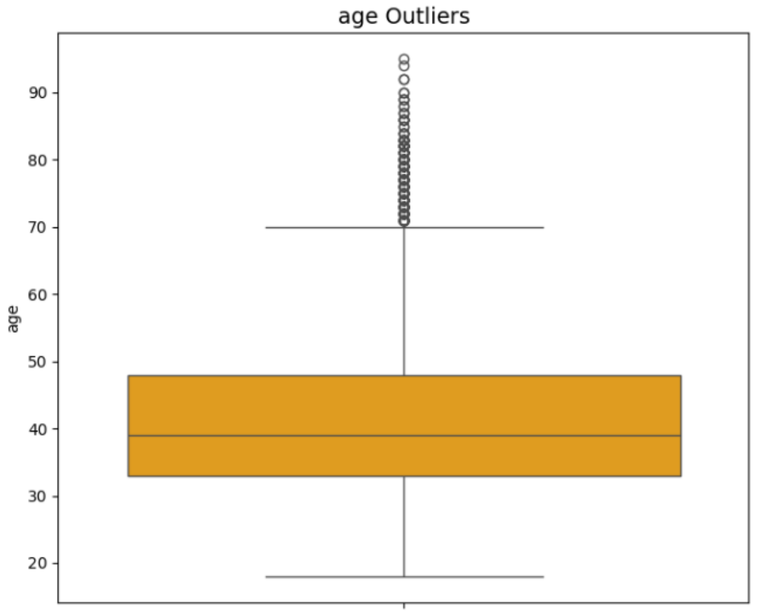


가입 하지 않은 사람(0)의 비율이 높음

-〉 Imbalanced data 처리 필요

1. 데이터 탐색 (EDA) - 수치형 변수 분포

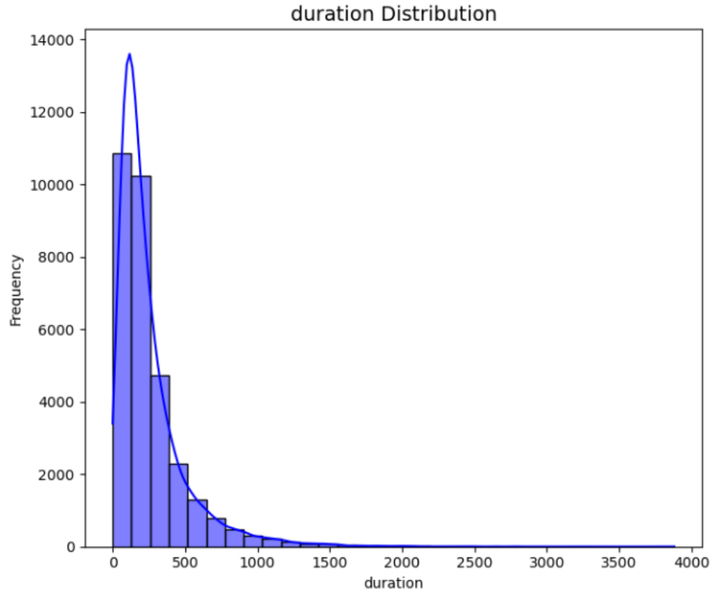




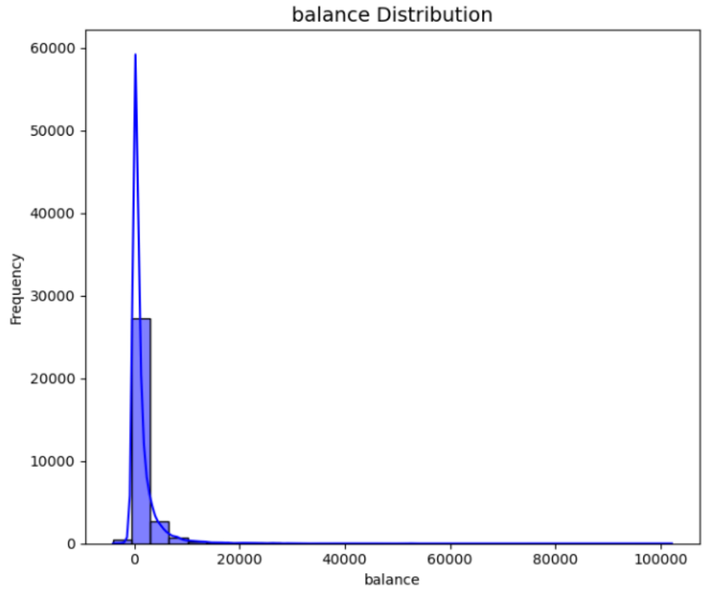
▶ age
- IQR: 15.0
- Lower Bound: 10.5, Upper Bound: 70.5
- Number of Outliers: 337

30, 40 대가 주 고객층

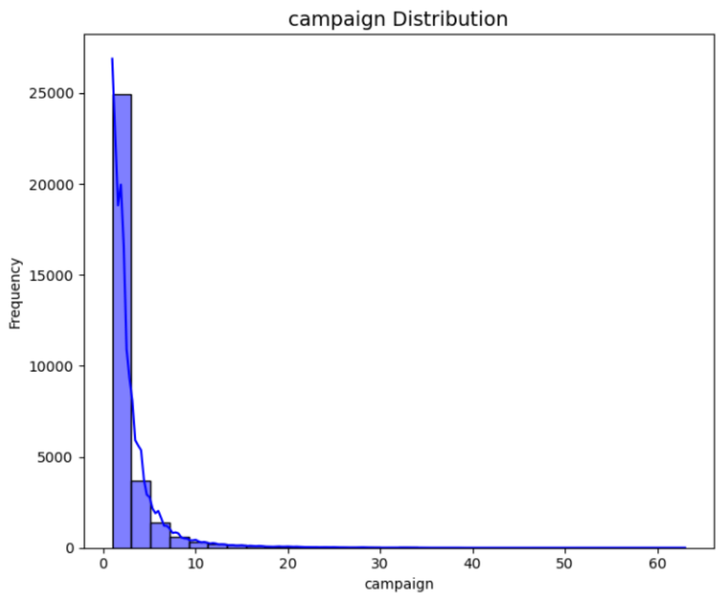
1. 데이터 탐색 (EDA) - 수치형 변수 분포



▶ duration
- IQR: 217.0
- Lower Bound: -222.5, Upper Bound: 645.5
- Number of Outliers: 2242



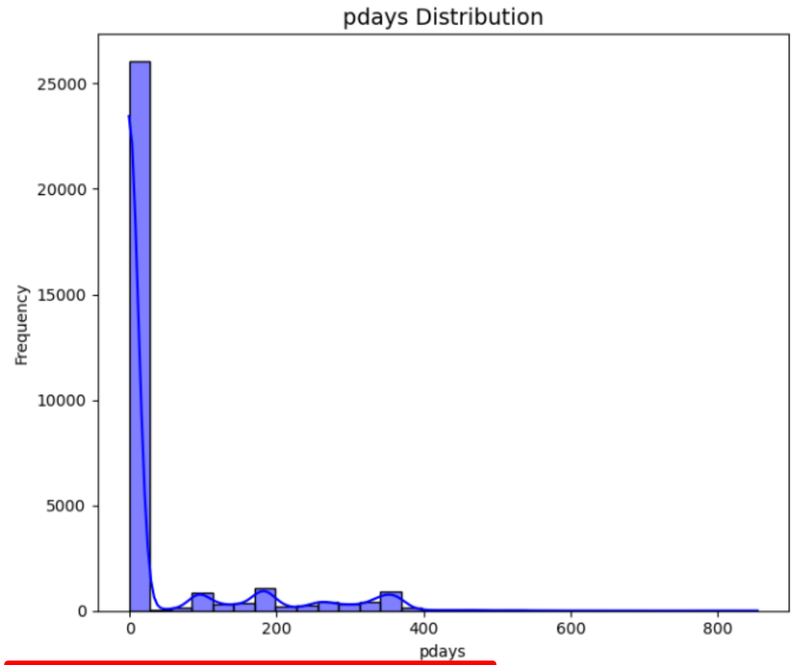
▶ balance
- IQR: 1345.5
- Lower Bound: -1945.25, Upper Bound: 3436.75
- Number of Outliers: 3327



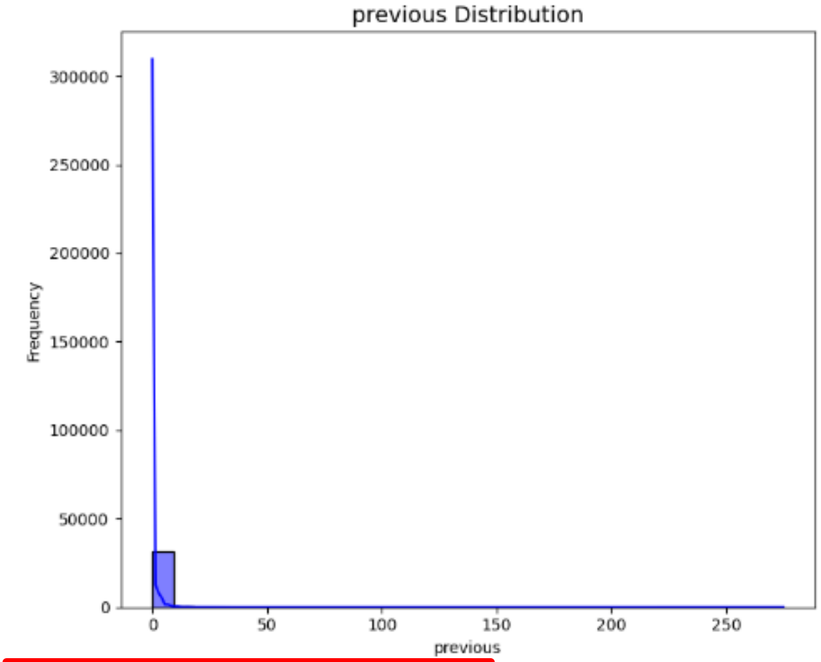
▶ campaign
- IQR: 2.0
- Lower Bound: -2.0, Upper Bound: 6.0
- Number of Outliers: 2134

왼쪽으로 치우쳐 진 그래프
-〉 스케일, 로그 변환 필요

1. 데이터 탐색 (EDA) - 수치형 변수 분포



▶ pdays
- IQR: 0.0
- Lower Bound: -1.0, Upper Bound: -1.0
- Number of Outliers: 5733

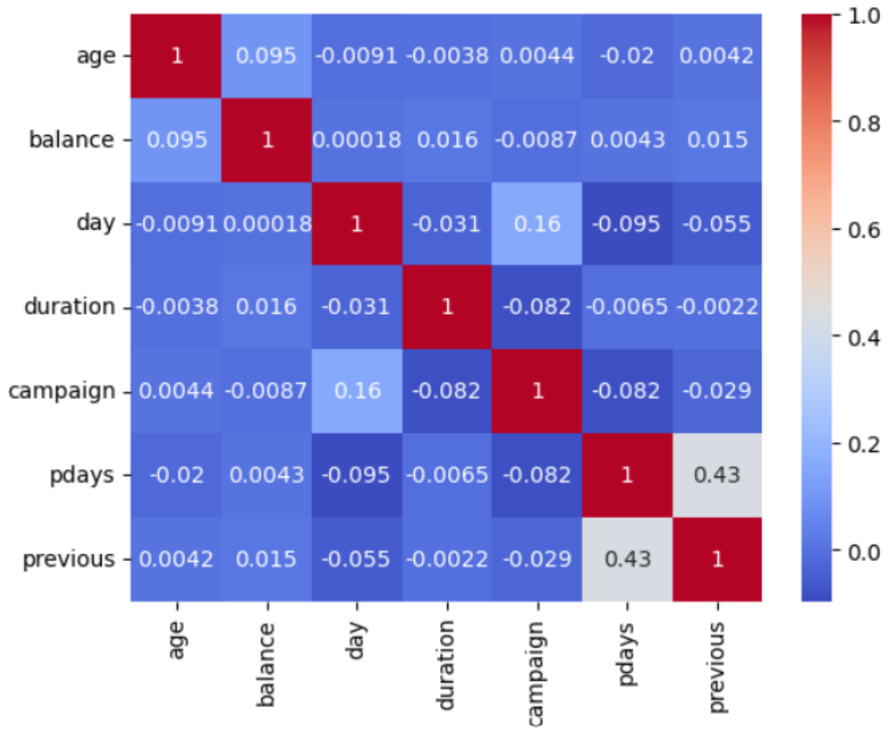


▶ previous
- IQR: 0.0
- Lower Bound: 0.0, Upper Bound: 0.0
- Number of Outliers: 5733

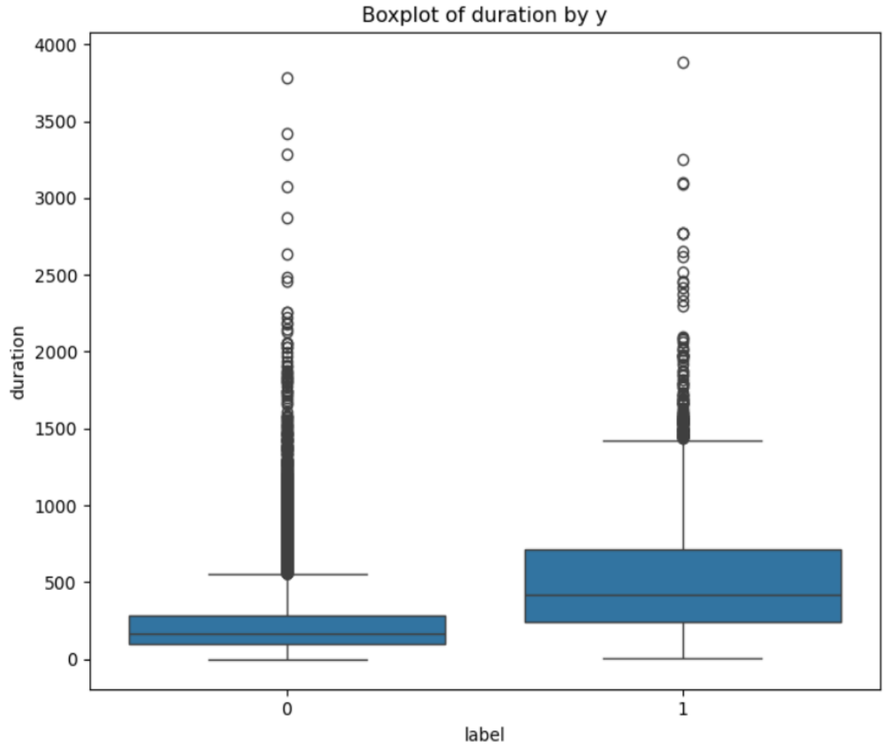
사분위수 범위가 연락하지 않은 데이터 (pdays = -1 / previous = 0)

-> 연락한 데이터는 이상치

1. 데이터 탐색 (EDA) – 수치형 변수 분포

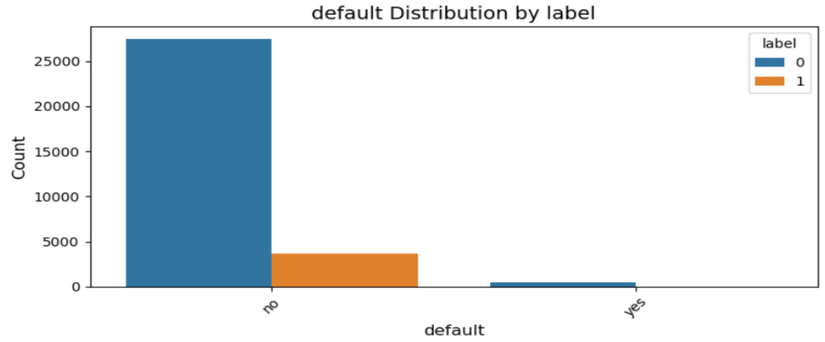


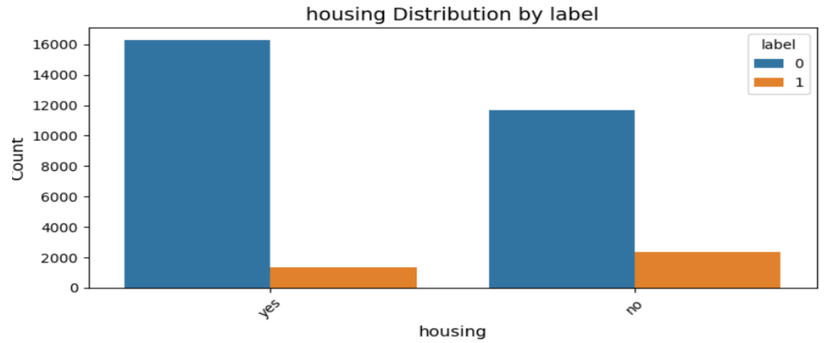
상관계수 히트맵

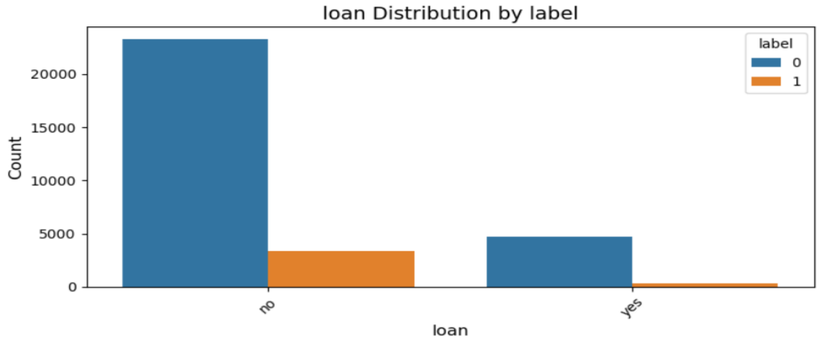


가입 여부에 따른
duration 값 변화

1. 데이터 탐색 (EDA) – 범주형 변수 분포

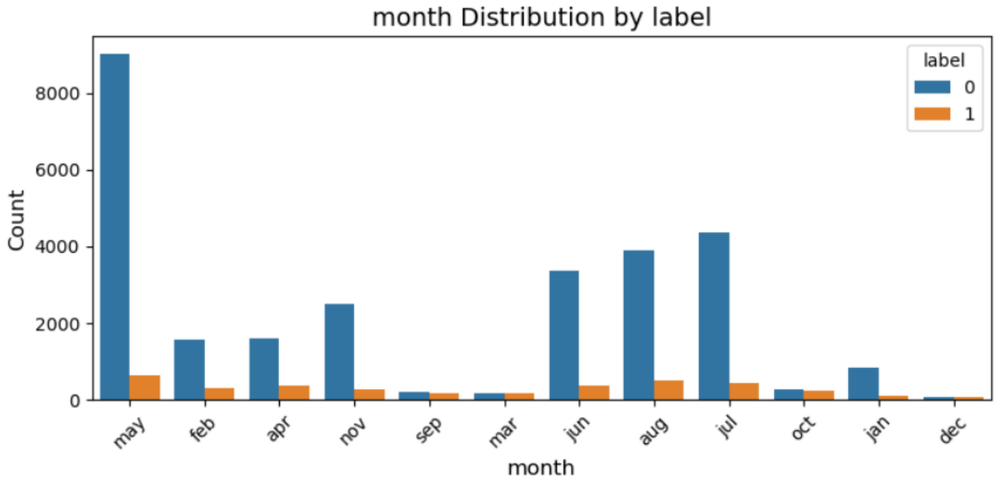


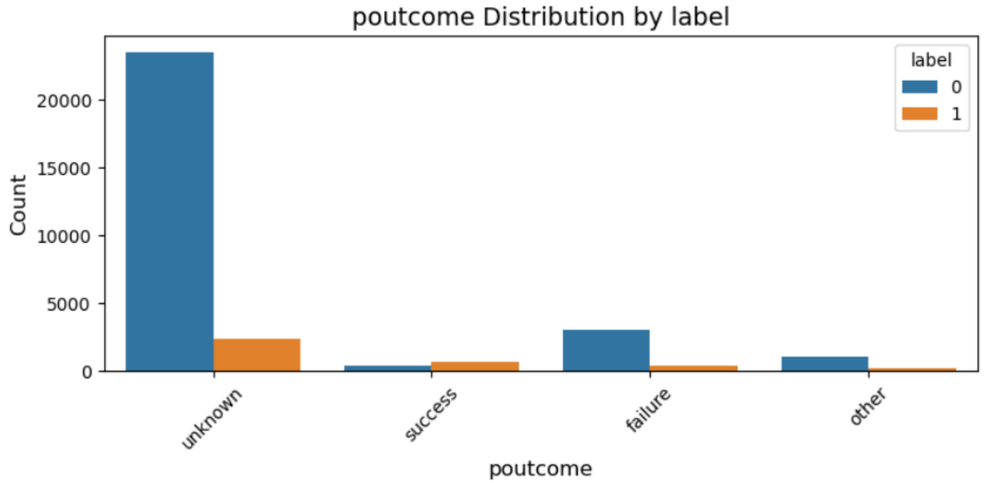





```
default
no     0.117791
yes    0.071685
Name: label, dtype: float64
-----------------------------------
housing
no     0.168425
yes    0.076006
Name: label, dtype: float64
-----------------------------------
loan
no     0.126348
yes    0.067396
Name: label, dtype: float64
-----------------------------------
```

1. 데이터 탐색 (EDA) – 범주형 변수 분포






```
month
mar    0.507003
dec    0.462585
oct    0.451796
sep    0.447761
apr    0.190882
feb    0.168700
aug    0.114616
nov    0.104032
jan    0.103448
jun    0.101178
jul    0.090909
may    0.064994
Name: label, dtype: float64
----------------------------
poutcome
success    0.648571
other      0.166005
failure    0.120760
unknown    0.092561
Name: label, dtype: float64
----------------------------
```

2. 데이터 전처리

수치형 변수

- 결측치 없음
- 스케일링
  -〉 Randomforest 불필요
- Balance 변수
  -〉 로그 변환

범주형 변수

- 결측치 없음
- month / poutcome 변수
  -〉 가입 비율을 추가하는
  특성 엔지니어링
- bool column
  -〉 yes = 1 / no = 0

함수 정의 및 적용

LabelEncoder() 처리

2. 데이터 전처리 - balance 변수

[34]:
```
df_trn['balance'].describe()
```

[34]:
```
count     31647.000000
mean       1370.050084
std        3122.054996
min       -4057.000000
25%          73.000000
50%         451.000000
75%        1418.500000
max      102127.000000
Name: balance, dtype: float64
```

-> balance 값에 음수 값 존재

[35]:
```
# 음수값 행 확인
negative_balance_count = df_trn[df_trn['balance'] < 0].shape[0]
print("음수값 행 개수:",negative_balance_count)
print("음수값 행 비율:",negative_balance_count / df_trn.shape[0])
```

```
음수값 행 개수: 2627
음수값 행 비율: 0.08300944797295162
```

[36]:
```
# 통장 잔고이기 때문에 음수값 그대로 두고, 양수 값은 로그 변환
def transform_balance(row, column_name):
    balance_value = row[column_name]

    if balance_value < 0:
        return balance_value  # 음수는 그대로 두기
    else:
        return np.log(balance_value + 1)  # 양수는 로그 변환 (0의 값을 고려해 1을 더한 후 진행)
```

2. 데이터 전처리 - month / poutcome 변수

[38]:
```
# 각 달별 가입 비율
(df_trn.groupby('month')['label'].mean().sort_values(ascending=False))
```

[38]:
```
month
mar    0.507003
dec    0.462585
oct    0.451796
sep    0.447761
apr    0.190882
feb    0.168700
aug    0.114616
nov    0.104032
jan    0.103448
jun    0.101178
jul    0.090909
may    0.064994
Name: label, dtype: float64
```

[43]:
```
(df_trn.groupby('poutcome')['label'].mean().sort_values(ascending=False))
```

[43]:
```
poutcome
success    0.648571
other      0.166005
failure    0.120760
unknown    0.092561
Name: label, dtype: float64
```

[40]:
```
# 각 달별 가입 비율
month_probability = {
    'jan': 0.103448, 'feb': 0.168700, 'mar': 0.507003, 'apr': 0.190882,
    'may': 0.064994, 'jun': 0.101178, 'jul': 0.090909, 'aug': 0.114616,
    'sep': 0.447761, 'oct': 0.451796, 'nov': 0.104032, 'dec': 0.462585
}
```

[41]:
```
# 가입 확률 추가 함수
def add_month_probability(df, month_probability):
    df['month_prob'] = df['month'].map(month_probability)
    return df
```

[44]:
```
# 'poutcome'에 대한 가입 비율
poutcome_probability = {
    'success': 0.648571,
    'other': 0.166005,
    'failure': 0.120760,
    'unknown': 0.092561
}
```

[45]:
```
# 가입 비율 추가 함수
def add_poutcome_probability(df, poutcome_probability):
    df['poutcome_prob'] = df['poutcome'].map(poutcome_probability)
    return df
```

2. 데이터 전처리 - bool 변수

[47]:
```
# 'yes' -> 1 / 'no' -> 0
def get_dummy_from_bool(row, column_name):
    return 1 if row[column_name] == 'yes' else 0
```

2. 데이터 전처리 – 함수 적용

[49]:
```python
def clean_data(df):

    #데이터 복사 -> 원본 데이터를 변경하지 않기 위해 사용
    cleaned_df = df.copy()

    # 1. balance 변수 처리: 로그 변환
    cleaned_df['balance_log'] = df.apply(lambda row: transform_balance(row, 'balance'), axis=1)
    cleaned_df = cleaned_df.drop(columns=['balance']) # 기존 balance 컬럼 삭제

    # 2. month 변수 처리: 각 월별 가입 확률을 특성으로 추가
    cleaned_df = add_month_probability(cleaned_df, month_probability)

    # 3. poutcome 변수 처리: 각 카테고리 별 가입 확률을 특성으로 추가
    cleaned_df = add_poutcome_probability(cleaned_df, poutcome_probability)

    # 4. bool column 변환
    bool_columns = ['default', 'housing', 'loan'] # bool 컬럼 정의
    for bool_col in bool_columns:
        cleaned_df[bool_col + '_bool'] = df.apply(lambda row: get_dummy_from_bool(row, bool_col),axis=1)
    cleaned_df = cleaned_df.drop(columns = bool_columns) # 기존 bool 칼럼 삭제

    return cleaned_df
```

[51]:
```python
clean_df_trn.head()
```

[51]:

	ID	age	job	marital	education	contact	day	month	duration	campaign	pdays	previous	poutcome	balance_log	month_prob	poutcome_prob	default_bool	housing_bool	loan_bool
0	train00001	34	blue-collar	married	primary	unknown	23	may	100	4	-1	0	unknown	5.883322	0.064994	0.092561	0	1	0
1	train00002	33	blue-collar	married	secondary	unknown	20	may	172	1	-1	0	unknown	-53.000000	0.064994	0.092561	0	1	0
2	train00003	32	management	single	tertiary	cellular	2	feb	56	1	-1	0	unknown	5.337538	0.168700	0.092561	0	1	0
3	train00004	37	blue-collar	divorced	secondary	cellular	8	may	326	1	326	2	success	6.459904	0.064994	0.648571	0	1	0
4	train00005	33	housemaid	married	secondary	cellular	11	may	256	1	-1	0	unknown	6.717805	0.064994	0.092561	0	1	0

2. 데이터 전처리 - LabelEncoder()

[57]:
```
# LabelEncoder 객체 저장용 딕셔너리
# test 데이터 예측시, 같은 Label Encoder를 사용해야 하기 때문에 설정
label_encoders = {}
```

[58]:
```
for col in clean_df_trn.columns:
    if clean_df_trn[col].dtype == 'O' and col != 'ID':  # 문자열/범주형 컬럼에 대해 (ID 컬럼 제외)
        labne = LabelEncoder()
        clean_df_trn[col] = labne.fit_transform(clean_df_trn[col])
        label_encoders[col] = labne  # 각 열에 대한 LabelEncoder 객체 저장
```

[59]:
```
label_encoders
```

[59]:
```
{'job': LabelEncoder(),
 'marital': LabelEncoder(),
 'education': LabelEncoder(),
 'contact': LabelEncoder(),
 'month': LabelEncoder(),
 'poutcome': LabelEncoder()}
```

[60]:
```
clean_df_trn.head()
```

[60]:

	ID	age	job	marital	education	contact	day	month	duration	campaign	pdays	previous	poutcome	abel	balance_log	month_prob	poutcome_prob
0	train00001	34	1	1	0	2	23	8	100	4	-1	0	3	0	5.883322	0.064994	0.092561
1	train00002	33	1	1	1	2	20	8	172	1	-1	0	3	0	-53.000000	0.064994	0.092561
2	train00003	32	4	2	2	0	2	3	56	1	-1	0	3	0	5.337538	0.168700	0.092561
3	train00004	37	1	0	1	0	8	8	326	1	326	2	2	0	6.459904	0.064994	0.648571
4	train00005	33	3	1	1	0	11	8	256	1	-1	0	3	0	6.717805	0.064994	0.092561

3. 모델 학습

[67]:
```
total_cols = [x for x in clean_df_trn.columns]

id_cols = ['ID']
y_cols = ['label']
x_cols = [col for col in total_cols if col not in id_cols + y_cols]
```

[68]:
```
X = clean_df_trn[x_cols]
y = clean_df_trn[y_cols]
```

[70]:
```
sm = SMOTE(random_state=17)
```

[71]:
```
X_sm, y_sm = sm.fit_resample(X, y)
```

[72]:
```
print(y.value_counts()) # SMOTE 적용 전
print('--------------------')
print('--------------------')
print(y_sm.value_counts()) # SMOTE 적용 후
```

```
label
0        27945
1         3702
Name: count, dtype: int64
--------------------
--------------------
label
0        27945
1        27945
Name: count, dtype: int64
```

[73]:
```
print(X.shape) # SMOTE 적용 전
print('--------------------')
print('--------------------')
print(X_sm.shape) # SMOTE 적용 후
```

```
(31647, 18)
--------------------
--------------------
(55890, 18)
```

[75]:
```
X_train, X_test, y_train, y_test = train_test_split(X_sm, y_sm, test_size=0.2, random_state=17,  stratify = y_sm)
```

3. 모델 학습

[77]:
```python
def objective(trial):

    n_estimators = trial.suggest_int('n_estimators', 10, 200) # 트리 개수 10~200 사이 탐색
    max_depth = trial.suggest_int('max_depth', 5, 200) # 트리의 최대 깊이를 5~200 사이 탐색
    min_samples_split = trial.suggest_int('min_samples_split', 2, 50) # 최소 샘플 수 2~50 사이 탐색
    criterion = trial.suggest_categorical('criterion', ['gini', 'entropy', 'log_loss'] ) # 분할 품질 측정 기준
    #train.sugget_* 메서드를 사용해 각 하이퍼파라미터의 범위 또는 값 집합을 지정

    params = {
        'n_estimators': n_estimators,
        'max_depth': max_depth,
        'min_samples_split': min_samples_split,
        'criterion': criterion
    }

    model = RandomForestClassifier(random_state=17, **params) # 지정된 하이퍼파라미터로 모델 생성

    score = cross_val_score(model, X_train, y_train, n_jobs=-1 , cv=5).mean() # 5-fold 교차 검증 수행하여 모델의 평균 점수 계산

    return score # score을 반환하여 Optunar가 최대화 하도록 실행
```

[78]:
```python
# 최적화 과정을 관리하는 study 객체 생성
# 랜덤 샘플링
study = optuna.create_study(direction='maximize', sampler=optuna.samplers.RandomSampler(seed=17))
```

[79]:
```python
# 함수를 50번 호출하며 최적의 하이퍼파라미터 탐색
study.optimize(objective, n_trials=50)
```

3. 모델 학습

[80]:
```
# 최적의 파라미터 확인
study.best_params
```

[80]:
```
{'n_estimators': 131,
 'max_depth': 117,
 'min_samples_split': 3,
 'criterion': 'entropy'}
```

[82]:
```
rfc_model = RandomForestClassifier(random_state=17, **study.best_params)
```

[83]:
```
rfc_model.fit(X_train, y_train)
```

[83]:
```
RandomForestClassifier(criterion='entropy', max_depth=117, min_samples_split=3,
                       n_estimators=131, random_state=17)
```

4. 모델 예측 및 평가

```
[85]: y_test_pred = rfc_model.predict(X_test)
```

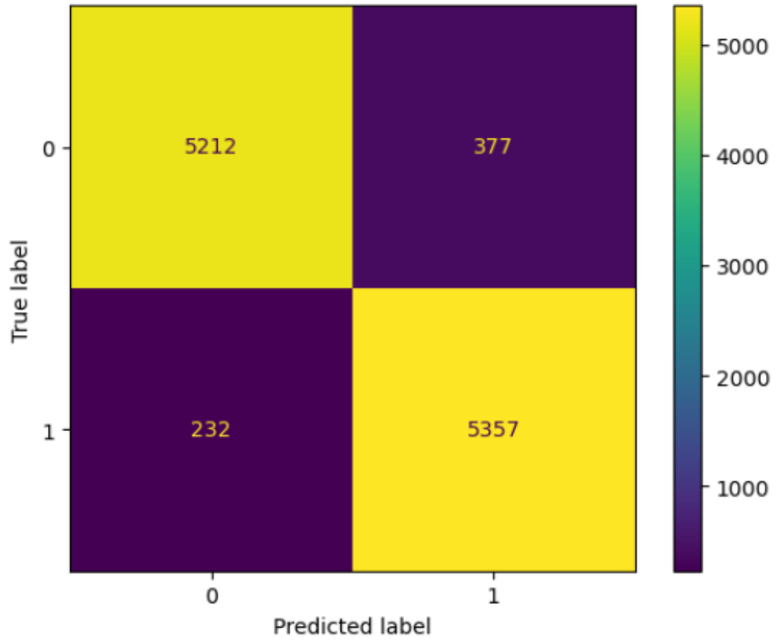
```
[86]: y_test_score = rfc_model.predict_proba(X_test)
```

```
[88]: print(f'In test set, accuracy={tst_acc:.4f}, precision={tst_pre:.4f}, recall={tst_rec:.4f}, f1={tst_f1:.4f}')
      print(f'In test set, AUROC={tst_auroc:.4f}')
```

```
In test set, accuracy=0.9455, precision=0.9343, recall=0.9589, f1=0.9462
In test set, AUROC=0.9907
```

```
[89]: ConfusionMatrixDisplay.from_predictions(y_true=y_test,
                                              y_pred=y_test_pred)
```

```
[89]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1ae6ae837a0>
```



4. 모델 예측 및 평가

[91]:
```
clean_df_tst.head()
```

[91]:

	ID	age	job	marital	education	contact	day	month	duration	campaign	pdays	previous	poutcome	label	balance_log	month_prob	poutcome_prob	de
0	test00001	54	4	0	2	0	20	9	101	2	-1	0	3	NaN	8.765771	0.104032	0.092561	
1	test00002	56	10	0	0	0	28	4	156	2	-1	0	3	NaN	5.645447	0.103448	0.092561	
2	test00003	34	4	1	1	0	21	1	314	3	-1	0	3	NaN	5.874931	0.114616	0.092561	
3	test00004	55	4	1	2	0	12	1	940	10	-1	0	3	NaN	6.343880	0.114616	0.092561	
4	test00005	38	9	1	1	0	21	1	252	4	-1	0	3	NaN	8.814182	0.114616	0.092561	

[92]:
```
X = clean_df_tst[x_cols]
```

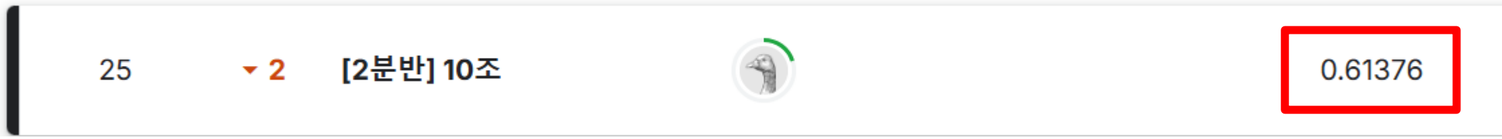
[93]:
```
y_pred = rfc_model.predict(X)
```

[94]:
```
clean_df_tst['label'] = y_pred
```

[95]:
```
clean_df_tst['label'].value_counts()
```

[95]:
```
label
0    11782
1     1782
Name: count, dtype: int64
```

5. 보완점




```
[88]: print(f'In test set, accuracy={tst_acc:.4f}, precision={tst_pre:.4f}, recall={tst_rec:.4f}, f1={tst_f1:.4f}')
      print(f'In test set, AUROC={tst_auroc:.4f}')
```

```
In test set, accuracy=0.9455, precision=0.9343, recall=0.9585, f1=0.9462
In test set, AUROC=0.9907
```

모델 과적합 문제

5. 보완점

```
[38]: # 각 달별 가입 비율
      (df_trn.groupby('month')['label'].mean().sort_values(ascending=False))
```

```
[38]: month
      mar    0.507003
      dec    0.462585
      oct    0.451796
      sep    0.447761
      apr    0.190882
      feb    0.168700
      aug    0.114616
      nov    0.104032
      jan    0.103448
      jun    0.101178
      jul    0.090909
      may    0.064994
      Name: label, dtype: float64
```

```
[43]: (df_trn.groupby('poutcome')['label'].mean().sort_values(ascending=False))
```

```
[43]: poutcome
      success    0.648571
      other      0.166005
      failure    0.120760
      unknown    0.092561
      Name: label, dtype: float64
```

```
# 각 달의 데이터 수
df_trn['month'].value_counts()
```

```
month
may    9647
jul    4807
aug    4406
jun    3736
nov    2778
apr    1996
feb    1885
jan     957
oct     529
sep     402
mar     357
dec     147
Name: count, dtype: int64
```

```
# 각 카테고리별 데이터 수
df_trn['poutcome'].value_counts()
```

```
poutcome
unknown    25918
failure     3420
other       1259
success     1050
Name: count, dtype: int64
```

가입 비율이 높은 카테고리가
데이터 수가 적음
-> 신뢰하기 어려운 비율

5. 보완점

스무딩 적용

- 계산식:

$$\text{smoothed\_rate} = \frac{\text{가입자 수} + \alpha}{\text{총 샘플 수} + \beta}$$

  - α와 β는 조정 가능한 스무딩 파라미터입니다.
    일반적으로 $\alpha = 1, \beta = 2$를 시작점으로 사용할 수 있습니다.

```
# 달별 가입 비율 계산
monthly_rate = df.groupby('month')['label'].mean()

# 스무딩 적용
alpha = 1
beta = 2
total_mean = df['label'].mean()
smoothed_rate = (df.groupby('month')['label'].sum() + alpha) / \
                (df.groupby('month')['label'].count() + beta)

# 원 데이터에 특성 추가
df['monthly_rate'] = df['month'].map(monthly_rate)
df['smoothed_rate'] = df['month'].map(smoothed_rate)
```

5. 보완점

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31647 entries, 0 to 31646
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              31647 non-null  object
 1   age             31647 non-null  int64
 2   job             31647 non-null  object
 3   marital         31647 non-null  object
 4   education       31647 non-null  object
 5   contact         31647 non-null  object
 6   day             31647 non-null  int64
 7   month           31647 non-null  object
 8   duration        31647 non-null  int64
 9   campaign        31647 non-null  int64
 10  pdays           31647 non-null  int64
 11  previous        31647 non-null  int64
 12  poutcome        31647 non-null  object
 13  label           31647 non-null  int64
 14  balance_log     31647 non-null  float64
 15  month_prob      31647 non-null  float64
 16  poutcome_prob   31647 non-null  float64
 17  default_bool    31647 non-null  int64
 18  housing_bool    31647 non-null  int64
 19  loan_bool       31647 non-null  int64
dtypes: float64(3), int64(10), object(7)
```

너무 많은 변수
-> 변수 선택 필요

Q&A