은행 고객의 정기 예금 가입 여부 예측 모델링

통계학과 2020020344 정해성

Ⅰ. 개요

기업이 마케팅에 데이터 분석을 사용하는 목적 중 하나는 적은 비용으로 마케팅을 진행하여 최대한의 수익을 창출하기 위함이다. 이는 현재 진행 중이거나 완료한 마케팅의 성과와 원인을 이해하는 것으로부터 시작한다. 그 후 최적화, 고객 세분화. 예측 및 평가 단계를 거쳐 실질적으로 수익 창출을 달성할 수 있다.
이러한 이유로 고급통계적머신러닝 과목을 수강하며 배운 통계적 머신러닝 기법을 마케팅 데이터에 적용해보고자 고객의 정기 예금 가입 여부 예측을 프로젝트의 주제로 선정하였다. 데이터는 UCI Machine Learning Repository의 Bank Marketing Data Set을 사용했으며 총 20개의 features와 1개의 binary target variable이 존재한다. 샘플은 총 41188개이며 모형의 성능을 평가하기 위하여 EDA 및 전처리 전 train_test_split 함수를 사용하여 약 10%가량의 4119개의 테스트 데이터를 별도로 준비하였다. 그 후 나머지 37069개의 훈련용 데이터로 EDA 및 전처리, 그리고 모형 적합을 진행하고, 테스트 데이터로 평가를 진행하였다. 또한, Oversampling 기법인 SMOTE의 효과를 먼저 살펴본 뒤, 최종 모형을 적합했다. 최종 앙상블 모형의 기초 모형으로 accuarcy가 높은 5가지의 모형들을 선택하고, 각 모형의 적절한 초모수를 recall 기준으로 GridSearchCV를 이용해 결정했다.

Ⅱ. 데이터

데이터는 포르투갈 은행 기관의 전화 통화를 통해 이루어진 마케팅 데이터이다. 이번 과제의 목표는 마케팅을 통해 수집한 데이터를 이용하여 고객이 정기 예금에 가입할지 여부를 예측하는 모형을 만드는 것이다. 데이터는 크게 4가지의 정보가 들어있으며 각 범주마다의 특성 변수들은 다음과 같다.

범주	특성 변수	설명
Bank client data	Age	나이
	Job	직업
	Marital	결혼 여부
	Education	교육 수준
	Default	신용 불이행 여부
	Housing	주택담보대출여부
	Loan	개인 융자 여부
Related with the last contact of the current campaign	Contact	연락 유형
	Month	마지막으로 연락한 월
	Day_of_week	마지막으로 연락한 요일
	Duration	마지막 연락으로부터 지난 시간(단위:초)
Other attributes	Campaign	이번 마케팅 기간동안 연락한 횟수
	Pdays	이전 마케팅에서 마지막 연락 후 경과 일수 (999:이전 연락이 없음)
	Previous	이전 마케팅에서 연락한 횟수
	Poutcome	이전 마케팅의 결과
Social and economic context attributes	Emp.var.rate	고용 변동률-분기별
	Cons.price.idx	소비자 물가지수-월간
	Cons.conf.idx	소비자 신뢰지수-월간
	Euribor3m	3개월 만기 유리보율-일간
	Nr.employed	고용자 수-분기별

*유리보: 유료화를 단일통화로 하는 유럽연합 12개국의 시중은행 간 단기차입 금리

범주	변수	설명
Target	y	고객의 정기 예금 가입 여부 ('yes', 'no')

대부분의 특성 변수들은 범주형 또는 순서형으로 이루어져 있으며 target 변수 y는 이항 변수이므로 주어진 문제는 이항 분류 문제라고 정의 내릴 수 있다.

Ⅲ. EDA 및 전처리: 시험 데이터는 unseen이므로 훈련 데이터에 대해서만 EDA를 진행했다.

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	

우선 y값이 no와 yes인 문자형으로 코딩되어 있어 LabelEncoder를 사용해 수치형으로 전환했다. 정기 예금 가입을 의미하는 'yes'는 1로, 가입하지 않았음을 의미하는 'no'는 0으로 코딩했다. 또한, 데이터 전처리 및 EDA에 앞서 train_test_split 함수를 사용해 Train set은 (37069, 21), Test set은 (4119, 20) 차원으로 분할했으며, 테스트용 (4119,)차원의 y_test를 따로 분리했다.

```
df_train.info()
```

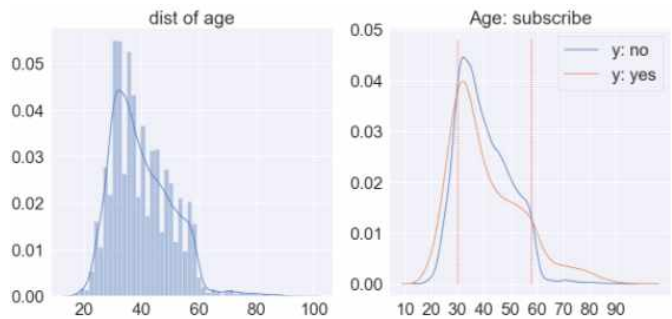
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 37069 entries, 26367 to 18945
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             37069 non-null  int64
 1   job             37069 non-null  object
 2   marital         37069 non-null  object
 3   education       37069 non-null  object
 4   default         37069 non-null  object
 5   housing         37069 non-null  object
 6   loan            37069 non-null  object
 7   contact         37069 non-null  object
 8   month           37069 non-null  object
 9   day_of_week     37069 non-null  object
 10  duration        37069 non-null  int64
 11  campaign        37069 non-null  int64
 12  pdays           37069 non-null  int64
 13  previous        37069 non-null  int64
 14  poutcome        37069 non-null  object
 15  emp.var.rate    37069 non-null  float64
 16  cons.price.idx  37069 non-null  float64
 17  cons.conf.idx   37069 non-null  float64
 18  euribor3m       37069 non-null  float64
 19  nr.employed     37069 non-null  float64
 20  y               37069 non-null  int32
dtypes: float64(5), int32(1), int64(5), object(10)
```

```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4119 entries, 0 to 4118
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             4119 non-null   int64
 1   job             4119 non-null   object
 2   marital         4119 non-null   object
 3   education       4119 non-null   object
 4   default         4119 non-null   object
 5   housing         4119 non-null   object
 6   loan            4119 non-null   object
 7   contact         4119 non-null   object
 8   month           4119 non-null   object
 9   day_of_week     4119 non-null   object
 10  duration        4119 non-null   int64
 11  campaign        4119 non-null   int64
 12  pdays           4119 non-null   int64
 13  previous        4119 non-null   int64
 14  poutcome        4119 non-null   object
 15  emp.var.rate    4119 non-null   float64
 16  cons.price.idx  4119 non-null   float64
 17  cons.conf.idx   4119 non-null   float64
 18  euribor3m       4119 non-null   float64
 19  nr.employed     4119 non-null   float64
dtypes: float64(5), int64(5), object(10)
```

< 1. Bank Client Data >

(1) age

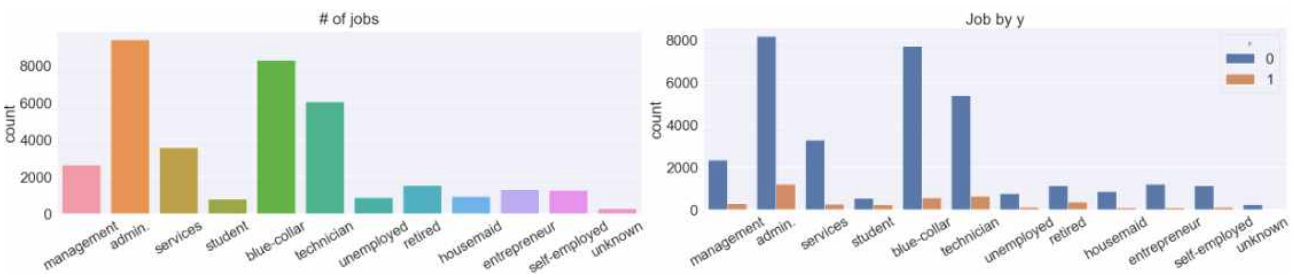


age_cat	y
0	0.15
1	0.09
2	0.32

age 변수를 살펴보면 30세 미만과 60세 이상에서 정기 예금에 가입한 사람이 더 많다. 이처럼 나이 분포에 따른 정기 예금 가입 여부가 차이가 나기 때문에 age는 y를 예측하는 데 중요한 변수라고 생각할 수 있으며, age를 30세 이하, 30 ~ 60세, 60세로 이루어진 age_cat이라는 범주형 변수를 생성했다.

(2) job: Student와 retired 그룹에서 정기 예금 가입률이 높다. 추후 pd.get_dummies로 변환한다.

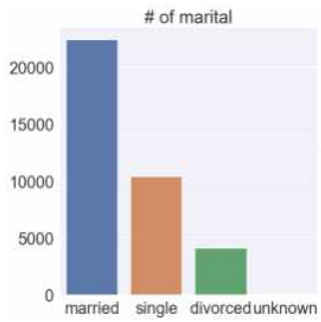
job	student	retired	unemployed	admin.	management	technician	self-employed	unknown	housemaid	entrepreneur	services	blue-collar
y	0.317781	0.246906	0.147651	0.128854	0.111822	0.107532	0.106698	0.105442	0.100832	0.085606	0.081376	0.070404

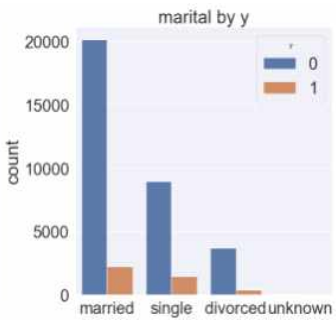


(3) **Marital:** unknown의 수가 매우 적으며, 결혼 여부는 정기 예금 가입률에 큰 영향을 미치지 않는다. 그러나 single일 때 소폭 높은 것을 확인할 수 있다. 추후 pd.get_dummies로 변환한다.

y	0	1	All
marital			
divorced	3709	424	4133
married	20166	2273	22439
single	8962	1469	10431
unknown	56	10	66
All	32893	4176	37069

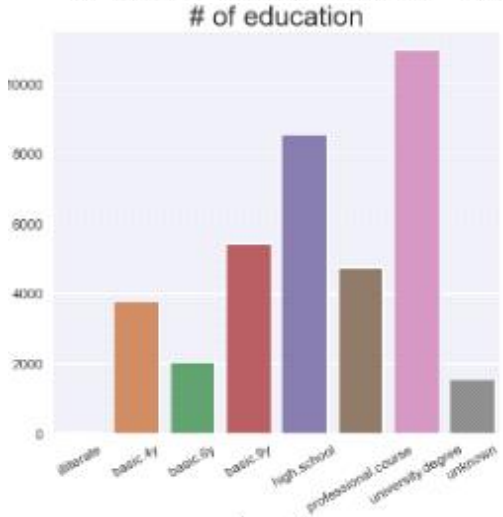
marital	y
unknown	0.151515
single	0.140830
divorced	0.102589
married	0.101297

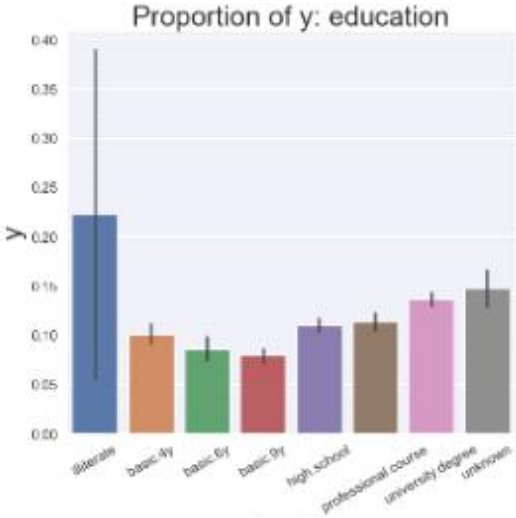




(4) **Education:** 순서형 변수이며 교육수준에 따라 정기 예금 가입률이 감소 후 증가하는 추세가 보인다. 추후 pd.get_dummies로 변환한다.

	0	1	2	3	4	5	6	7
education	basic.4y	basic.6y	basic.9y	high.school	illiterate	professional.course	university.degree	unknown
y	0.100612	0.0856031	0.0788359	0.109658	0.222222	0.113032	0.13582	0.146658





(5) **Default, Housing and Loan:**

- defualt변수는 yes가 오직 3개의 값만 존재하며 이는 전체 훈련 데이터의 0.009%에 불과하고 모두 y=0의 값을 갖고 있다. 또한, 나머지 값은 no 또는 unknown이기 때문에 실질적으로 이 변수는 y에 대한 정보를 주지 않고 있다고 볼 수 있다.
- yes, no, unknown별 y 비율의 차이가 없기 때문에 주택 담보 대출 여부(housing)와 개인 융자 여부는 y에 영향을 거의 미치지 않는 것처럼 보인다. 이를 확인하기 위해 피어슨의 카이제곱 검정을 진행했다.

```
from scipy.stats import chi2_contingency
for col in ['housing', 'loan']:
    stat, p, dof, expected = chi2_contingency(pd.crosstab(df_train['loan'], df_train['y']))
    alpha = 0.05
    print("Pearson Chi-sq Test Result for {}:".format(col))
    if p <= alpha: print('Reject H0: independence\n')
    else: print('Do not reject H0: independence\n')
```

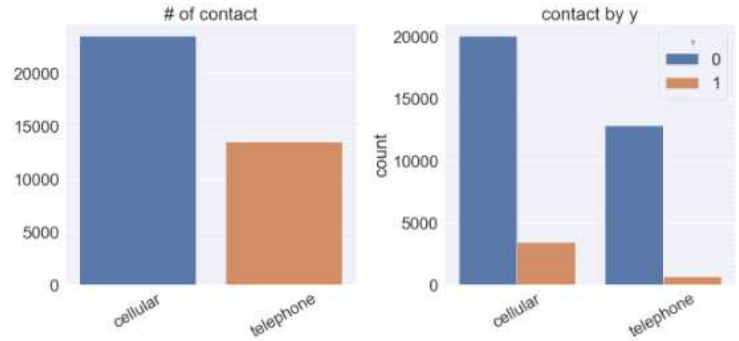
```
Pearson Chi-sq Test Result for housing:
Do not reject H0: independence

Pearson Chi-sq Test Result for loan:
Do not reject H0: independence
```

이 정보들을 바탕으로 default, housing, loan 변수는 모델 적합 전에 제거하기로 결정했다.

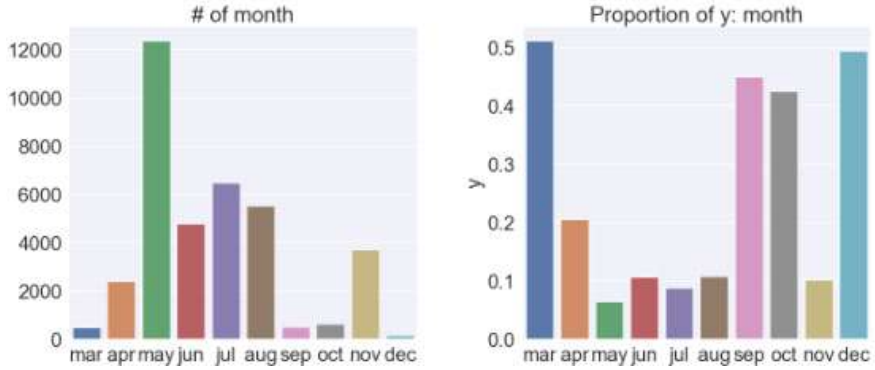
< 2. Related with the last contact of the current campaign >

(1) Contact: 종류에 따라 y의 비율 차이가 크게 나므로 좋은 예측변수가 되리라 생각되며 이항 변수이기 때문에 추후 0과 1로 변환한다.

contact	y
cellular	0.147346
telephone	0.052472

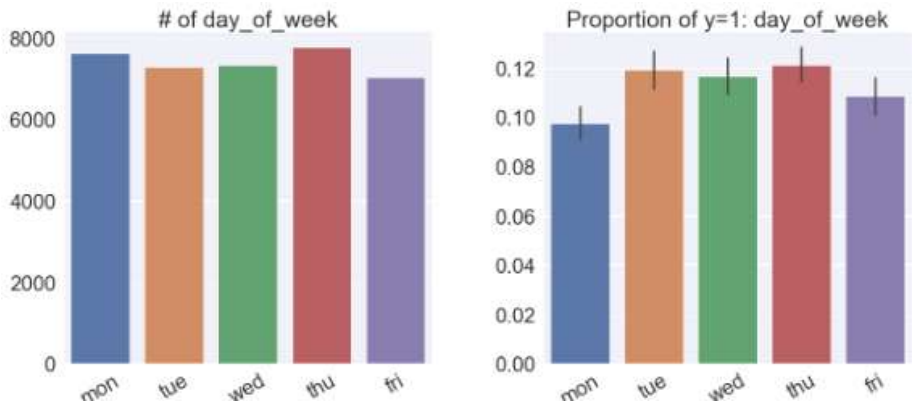


(2) Month: 1월과 2월에는 데이터가 없다. 즉 과거와 현재의 마케팅은 모두 3월-12월에 진행되었다. 또한, 대부분의 연락이 may(5월)에 몰려있다. 하지만 가장 적은 비율 가입률을 보이며, 오히려 mar(3월), sep(9월), oct(10월), dec(12월)은 상대적으로 적은 통화량이나 마케팅이 성공(y=1)일 확률이 훨씬 높다. 그러므로 추후 pd.get_dummies로 더미변수화 한다.

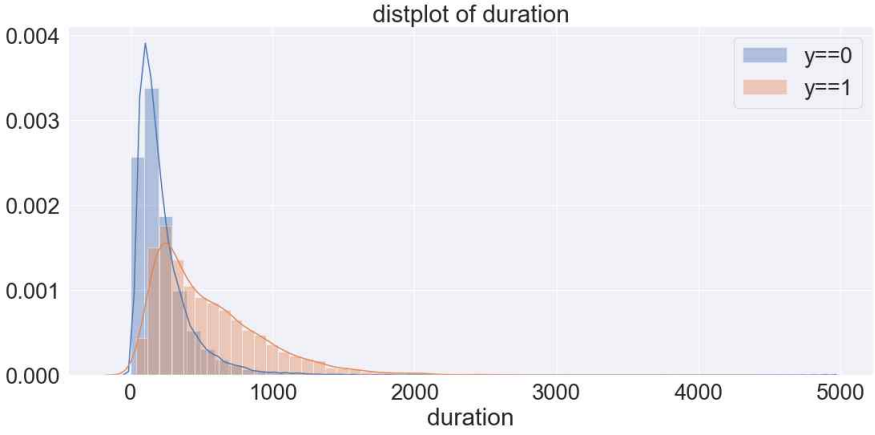


(3) Day_of_week: 주말에는 마케팅을 진행하지 않았으며 요일에 따라 y=1의 비율은 큰 차이가 없다. 그래도 가장 높은 날을 선택하면 목요일이라고 할 수 있다.

day_of_week	y
mon	0.097679
fri	0.108461
wed	0.116637
tue	0.119335
thu	0.121173

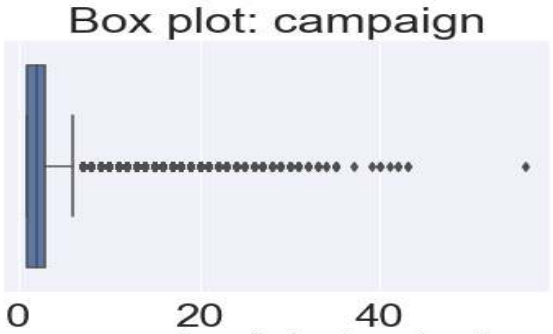


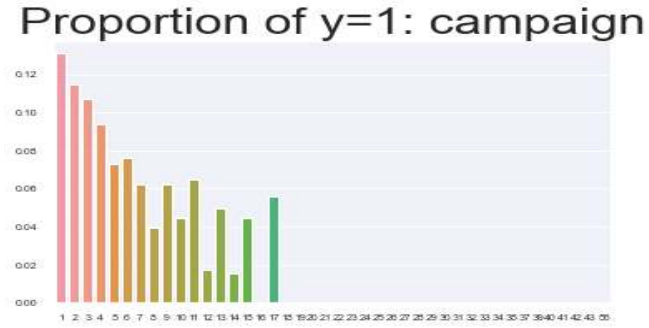
(4) Duration: 마지막 연락으로부터 지난 시간의 분포는 다음과 같으며, y가 0과 1일 때 큰 차이를 보이므로 중요한 변수라고 여길 수 있다. 또한, 좌측으로 치우친 분포이기 때문에 로그 변환을 고려해 본다.

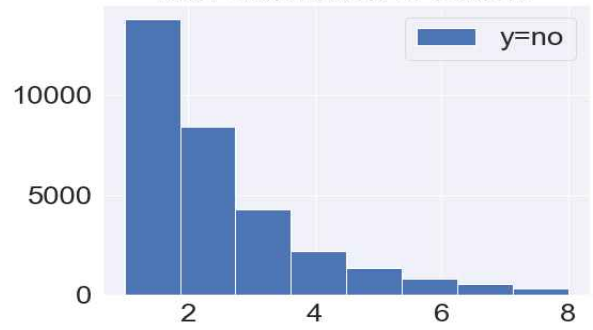


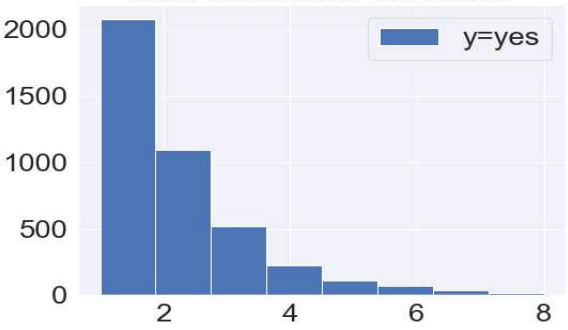
< 3. Other Attributes >

(1) Campaign: 이번 마케팅 기간동안 고객에게 연락한 횟수를 뜻하는 campaign은 대부분의 데이터가 10보다 작은 값을 갖는 매우 편향된 분포를 갖고 있다. 또한 y=1 비율 역시 campaign이 증가할수록 감소하는데 상식적으로 한 번의 마케팅에서 수차례 전화를 하는 것은 상품 가입률을 떨어뜨릴 것이라 예상이 가능하다. 따라서 boxplot의 수염(1.5*IQR)을 벗어나는 데이터(8 이상)를 이상치로 여기고 제거한다.

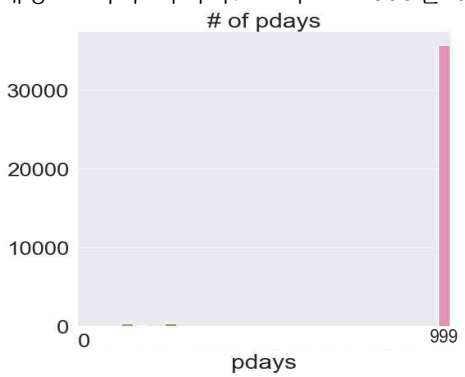


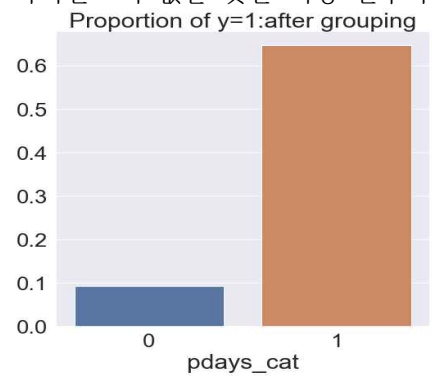






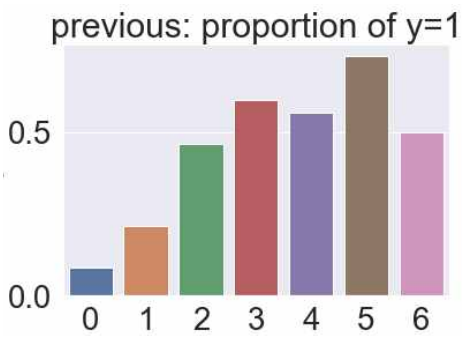
(2) Pdays: 이전 마케팅에서 고객에게 마지막 연락 이후의 경과 일수지만, 대부분의 값이 999로 다수는 이전 마케팅 대상 고객이 아니다. 그러므로 999면 0 아니면 1의 값을 갖는 이항 변수화를 시켜준다.

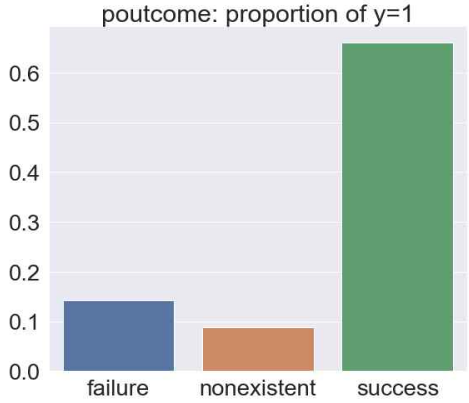




(3) Previous: 순서형 변수로 이전 마케팅에서 고객에게 연락한 횟수이다. 아래 그래프에서 보듯 2 이상의 값에서 y=1의 비율이 높다. 그러므로 previous가 2보다 크면 1 아니면 0을 갖는 이항변수를 생성한다.

(4) Poutcome: 이전 마케팅의 결과를 나타내는 변수로, success의 값에서 y=1의 비율이 높다. 즉, 이전 마케팅이 성공적으로 끝났다면, 이번 마케팅도 성공했을 가능성이 높다. 그러므로 nonexistent와 failure면 0, 성공이면 1의 값을 갖는 변수로 변환한다.



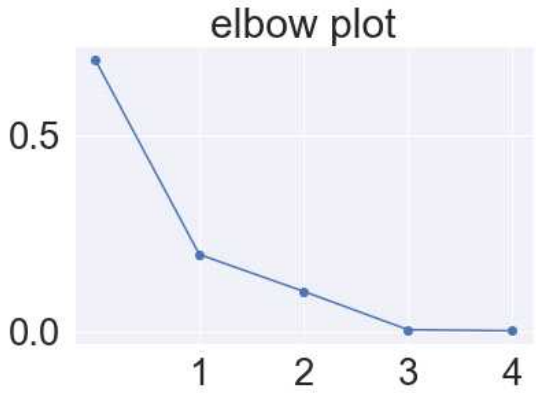


< 4. Social and Economic Context Attributes >

사회 및 경제 지표를 나타내는 변수들의 피어슨 상관계수는 다음과 같다.

	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
emp.var.rate	1.000000	0.776999	0.194333	0.972313	0.907141
cons.price.idx	0.776999	1.000000	0.058233	0.690020	0.524561
cons.conf.idx	0.194333	0.058233	1.000000	0.275865	0.098770
euribor3m	0.972313	0.690020	0.275865	1.000000	0.945269
nr.employed	0.907141	0.524561	0.098770	0.945269	1.000000

매우 높은 상관관계를 갖는 변수들이 있다. 따라서 이 5개의 변수들을 표준화 후 PCA를 통해 차원 축소를 고려할 수 있다. 3개의 주성분만으로도 99%의 분산을 설명하고 있으며, 이와 함께 elbow plot을 고려해서 3개의 주성분을 사용하기로 했다.



<EDA 및 전처리 결과> - 추후 평가를 위해 시험 데이터에도 동일한 처리를 했다.

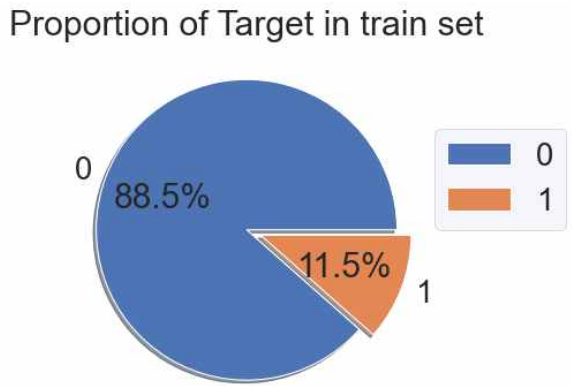
범주	특성 변수	전처리
Bank client data	Age	age_cat(0: ≤ 30, 1: 30~60, 2: 60 ≤)
	Job	더미 변수화
	Marital	더미 변수화
	Education	수치화(순서형 변수)
	Default	제거
	Housing	제거
	Loan	제거
Related with the last contact of the current campaign	Contact	더미 변수화
	Month	더미 변수화
	Day_of_week	더미 변수화
	Duration	로그 변환 및 정규화
Other attributes	Campaign	8 이상인 이상치 제거
	Pdays	pdays_cat(0: 999, 1: o/w)
	Previous	이항 변수화(0: <2, 1: 2 ≤)
	Poutcome	poutcome_cat (0:failure|'nonexistent, 1:'success')
Social and economic context attributes	Emp.var.rate	PCA(n_component=3)
	Cons.price.idx	
	Cons.conf.idx	
	Euribor3m	
	Nr.employed	

< 5. 최종 데이터의 형태 >

최종적으로 완성된 훈련 데이터 X(타겟 y)와 시험 데이터 X_test(타겟 y_test)는 각각 (35821, 38), (3990, 38)의 shape을 갖는다. SMOTE의 효과를 알아보기 위해 train_test_split함수로 훈련 데이터를 다시 각각 (28656, 38), (7165, 39)의 shape을 갖는 훈련 데이터 X_tr와 검증용 데이터 X_vld로 분할하였다. 이 데이터는 최종 모델링 과정에서는 사용하지 않았다. 최종 모델링에는 전체 훈련 데이터 X와 y를 사용하여 GridSearchCV 함수를 적용했기 때문이다.

Ⅳ. SMOTE의 효과

훈련 데이터의 Target variable y는 88.5%가 0을 갖는 매우 불균형한 데이터다.



이러한 경우 모형 훈련 시 불균형한 데이터를 그대로 사용한다면, 모형 역시 불균형한 학습이 이루어질 가능성이 높다. 단적인 예로 accuracy를 기준으로 훈련 데이터의 모든 샘플에 대해서 타겟을 0으로 예측한다면 accuracy는 88.5%를 기록할 것이다. 그러므로 모형 평가의 기준이 accuracy 하나로는 부족하다.

이번 프로젝트의 목표는 고객이 정기 예금에 가입할지 여부를 예측하는 것이다. 이러한 목표 하에서 다음과 같은 2가지의 오류가 발생할 수 있다.

1. 고객이 정기 예금에 가입하지 않았는데 가입했다고 예측하는 False Positive
2. 고객이 정기 예금에 가입했는데 가입하지 않았다고 예측하는 False Negative

우리에게 주어진 상황에서 더 줄여야 하는 오류를 골라야 한다면, False Negative라고 할 수 있다. 이미 가입한 고객을 미가입자로 분류한 뒤 마케팅을 진행하는 것은 비용 측면에서 손해이다. 즉, 실제 가입한 고객들을 제대로 분류하는 것이 더 중요하다.

그러므로 최종모형은 다음과 같은 단계로 생성하기로 결정했다. 우선 accuarcy가 높은 5가지의 모형들을 앙상블의 기초 모형으로 사용하고, 각 모형의 적절한 초모수를 recall 기준으로 GridSearchCV를 이용해 결정한다. 그 후, 적절한 초모수를 선택하는 GridSearchCV 시 가장 우선 시 되는 것은 Recall을 높이는 것이며, 이를 모형 훈련 시 scoring에 전달하기로 했다.

우선, SMOTE를 사용했을 때의 효과를 살펴보자. 이를 실험해보기 위해 35821개의 샘플이 있는 훈련 데이터를 train_test_split(startify=y)로 각각 (28656, 38)과 (7165, 38)의 shape을 갖는 X_tr, X_vld 그리고 (28656,)과 (7165,)의 shape을 갖는 y_tr, y_vld로 나눴다. SMOTE를 적용한 결과는 다음과 같다.

```
print('SMOTE 적용 전 Train data set: ', X_tr.shape, y_tr.shape)
print('SMOTE 적용 후 Train data set: ', X_tr_smote.shape, y_tr_smote.shape)
print('SMOTE 적용 전 레이블 값 분포: ₩n', pd.Series(y_tr).value_counts())
print('SMOTE 적용 후 레이블 값 분포: ₩n', pd.Series(y_tr_smote).value_counts())
```

```
SMOTE 적용 전 Train data set:  (28656, 38) (28656,)
SMOTE 적용 후 Train data set:  (50706, 38) (50706,)
SMOTE 적용 전 레이블 값 분포:
 0    25353
1     3303
Name: y, dtype: int64
SMOTE 적용 후 레이블 값 분포:
 1    25353
0    25353
Name: y, dtype: int64
```

그 후, X_tr과 y_tr로 모형을 훈련하고 X_vld, y_vld로 평가한 결과와, X_tr_smote, y_tr_smote로 모형을 적합하고 X_vld, y_vld로 평가한 결과를 비교했다. 이때 사용한 모형들은 KNN, Logistic Regression, SVM(kernel='rbf'), Decision Tree, Random Forest, XGBoost, LightGBM 등 7개의 모형이며, 각 초모수는 기본적으로 정해져있는 디폴트 값을 사용했다.

<SMOTE 전·후의 평가지표 값>

	accuracy		precision		recall		f1	
	original	sm	original	sm	original	sm	original	sm
knn	0.903	0.851	0.624	0.422	0.406	0.788	0.492	0.550
logit	0.905	0.863	0.637	0.444	0.406	0.754	0.496	0.559
svm	0.906	0.865	0.712	0.455	0.311	0.849	0.433	0.592
decision tree	0.894	0.881	0.538	0.488	0.553	0.643	0.545	0.555
rf	0.911	0.898	0.645	0.548	0.501	0.679	0.564	0.606
xgb	0.905	0.896	0.600	0.535	0.519	0.736	0.557	0.619
lgbm	0.912	0.892	0.636	0.520	0.542	0.820	0.586	0.636

비교한 7개 모형 모두에서 약간의 accuracy와 precision은 감소하지만, 정기 예금 가입자를 가입했다고 올바르게 예측하는 recall, 그리고 precision과 recall의 조화평균인 f1 score가 모두 증가한 것으로 보아 SMOTE를 사용하는 것이 더 좋다고 판단할 수 있다. 그러므로 SMOTE를 훈련 데이터에 사용해 최종 모형을 만들기로 했다.

V. 앙상블의 기초 모형 선정

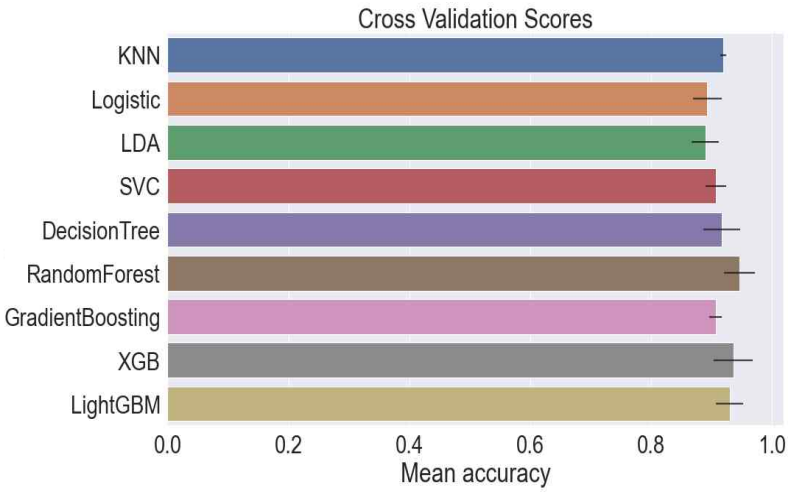
앞서 언급했듯 앙상블의 기초 모형의 기준은 accuracy를 통해 5개의 상위 모델을 결정한 뒤, 이들에 대해 recall을 기준으로 최적의 초모수를 찾기로 했다. 따라서, 전체 훈련 데이터에 SMOTE를 적용한 뒤 모형을 훈련시키기 위해, 다음과 같이 X_smote와 y_smote를 생성했다.

```
# 전체 Train set에 SMOTE 적용
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=0)
X_smote, y_smote = smote.fit_sample(X,y)
```

```
SMOTE 적용 전,후 Train data
:  (35821, 38) (63384, 38)
SMOTE 적용 전 레이블 값 분포:
 0    31692
1     4129
Name: y, dtype: int64
```

```
SMOTE 적용 후 레이블 값 분포:
 1    31692
0    31692
Name: y, dtype: int64
```

그 후, cross_val_score함수에 모형과 X_smote, y_smote를 훈련 데이터로, scoring='accuracy'로 설정한 뒤, StratifiedKFold(n_splits=10)으로 생성한 객체를 인자로 넣어 10-fold CV를 수행했다. 그런 다음 KNN, Logitstic Regression, LDA, SVC(kerenl='rbf'), Decision Tree, Random Forest, Graident Boosting, XGBoost, LightGBM 등 9개의 모형에 대해 교차 검증 평균과 표준편차를 구했다. 결과는 다음과 같다.



Algorithm	CVMeans	CVstd
RandomForest	0.945524	0.025547
XGB	0.935679	0.032169
LightGBM	0.929968	0.022466
KNN	0.919191	0.005001
DecisionTree	0.917252	0.030468
SVC	0.907485	0.016942
GradientBoosting	0.906775	0.010328
Logistic	0.892829	0.023556
LDA	0.889642	0.022323

이 결과를 바탕으로, CV accuracy 평균이 높은 Random Forest, XGBoost, LightGBM, KNN과 Decision Tree를 앙상블 모형의 기초 모형으로 결정했다.

Ⅵ. 각 모형의 초모수 결정

SMOTE를 사용해서 교차검증을 통한 초모수 결정을 시행할 때의 가장 중요한 점은, oversampling이 교차검증 이전에 이루어져서는 안된다는 것이다. 그 이유는 교차검증을 수행하기 전에 SMOTE를 사용했다면, 이를 평가하는 Validation Fold가 Unseen이라는 가정을 만족하지 못하기 때문이다. 그러므로 SMOTE기법은 교차검증 전이 아니라, 교차검증을 수행하면서 이루어져야 한다. 예를 들어 5-fold cv를 진행할 때, 훈련 데이터로 사용되는 4개의 fold에 대해서 SMOTE를 적용한 뒤 모형을 훈련해야 한다는 것이다. 그 후 validation set에 대해 예측하고 평가해야 한다. 이러한 과정을 다음과 같이 for 반복문과 imblearn 패키지 내의 make_pipeline을 사용해 진행했다.

[KNN의 초모수 결정 - 교차검증 수행 중 oversampling(SMOTE)]

```
from imblearn.pipeline import make_pipeline as imb_pipeline

skf = StratifiedKFold(n_splits=5, random_state=0)
accuracy_lst_knn = []
precision_lst_knn = []
recall_lst_knn = []
f1_lst_knn = []
auc_lst_knn = []

knn_sm2 = KNeighborsClassifier()

knn_param = {"n_neighbors":list(range(1,20))}

gs_knn_sm2 = GridSearchCV(knn_sm2,param_grid = knn_param, cv=skf, scoring="recall", n_jobs= -1, verbose = 1)

for train, test in skf.split(X_train_original, y_train_original):
    pipe = imb_pipeline(SMOTE(sampling_strategy='minority'), gs_knn_sm2)
    model = pipe.fit(X_train_original[train], y_train_original[train])
    best_est_knn = gs_knn_sm2.best_estimator_
    pred = best_est_knn.predict(X_train_original[test])
    accuracy_lst_knn.append(pipe.score(X_train_original[test], y_train_original[test]))
    precision_lst_knn.append(precision_score(y_train_original[test], pred))
    recall_lst_knn.append(recall_score(y_train_original[test], pred))
    f1_lst_knn.append(f1_score(y_train_original[test], pred))
    auc_lst_knn.append(roc_auc_score(y_train_original[test], pred))

print("accuracy: {}".format(np.mean(accuracy_lst_knn)))
print("precision: {}".format(np.mean(precision_lst_knn)))
print("recall: {}".format(np.mean(recall_lst_knn)))
print("f1: {}".format(np.mean(f1_lst_knn)))
```

이 과정을 거쳐 얻은 앙상블 모형의 최종 베이스 모형들은 다음과 같다.

모형	초모수
RandomFoestClassifier	n_neighbors=5
XGBClassifier	base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.05, max_depth=9, min_child_weight=1, n_estimators=50
LGBMClassifier	learning_rate=0.05, n_estimators=50, num_leaves=35
KNeighborsClassifier	criterion='entropy', max_features=10, min_samples_leaf=10, n_estimators=200
DecisionTreeClassifier	criterion='entropy', max_depth=3

Ⅶ. 앙상블 모형(최종 모형)

예측력을 높이고자 앞에서 구한 5개의 모형들을 활용해 앙상블 모형 즉, hard 방식의 VotingClassifier를 생성하기로 한다. 과정은 VotingClassifier에 앞서 구한 모형들과 함께 voting='hard'를 전달하고, 전체 훈련 데이터에 적용한 X_smote, y_smote를 활용해 훈련시킨다. 그리고 이 모형을 최종모형으로 정했다.

```
vote_smote = VotingClassifier(estimators=[('knn', best_est_knn),('rf',best_est_rf),('dt', best_est_dt),('lgbm',best_est_lgbm),
                                          ('xgb',best_est_xgb)], voting='hard', n_jobs=-1)
vote_smote.fit(X_smote, y_smote)
```

최종 모형을 사용해 구한 훈련 데이터와 시험 데이터에 대한 예측 결과는 다음과 같다.

Confusion Matrix: Train set			
		Prediction	
		0	1
True	0	28045	3647
	1	630	31062

Confusion Matrix: test set			
		Prediction	
		0	1
True	0	3088	447
	1	43	412

Classification Report: Train set				
	Precision	Recall	F1_score	Support
0	0.98	0.88	0.93	31692
1	0.89	0.98	0.94	31692
accuracy			0.93	63384
macro avg	0.94	0.93	0.93	63384
weighted avg	0.94	0.93	0.93	63384

Classification Report: Test set				
	Precision	Recall	F1_score	Support
0	0.99	0.87	0.93	3535
1	0.48	0.91	0.63	455
accuracy			0.88	3990
macro avg	0.73	0.89	0.78	3990
weighted avg	0.93	0.88	0.89	3990

훈련 데이터에 대한 Accuracy는 0.93이고 시험 데이터에 대한 Accuracy는 0.88이다. 그러나 Target variable y가 불균형한 자료이기 때문에 Recall을 살펴봐야 한다. 우리의 관심은 y=1에 대한 예측이므로 각 Classification Report의 1이 있는 행을 살펴보면, 훈련 데이터에 대한 Recall은 0.98, 시험 데이터에 대한 Recall은 0.91이다. 훈련 데이터에 대한 것만큼은 아니지만 시험 데이터에 대해서도 어느 정도 실제 정기 예금 상품에 가입한 고객들을 모형이 올바르게 예측하는 비율이 꽤 높다는 것을 확인할 수 있다. 그러나 과대적합의 문제가 보인다는 점이 아쉬움으로 남는다.

Ⅷ. 한계점 및 느낀점

배운 이론을 데이터에 응용해보니 많은 것을 배웠다. 우선, 불균형 데이터이다 보니 모형 훈련 시 편향이 생길 수 있어 resampling 기법을 사용했다. 그중에서도 일반적으로 통계적으로 유용한 과대표집 방법 중 SMOTE 방법을 사용했다. 그러나 처음에는 각 모형의 초모수를 결정하는 과정에서 교차검증 수행 전에 SMOTE를 적용하고 심한 과대적합의 결과를 얻었다. 구글링을 통해 이유를 찾아보니 oversamling이나 undersampling 기법은 교차검증을 진행하는 과정에서 사용해야 한다는 것을 배웠다. 교차검증을 수행할 때 validation set은 unseen data여야 하는데, 만약 미리 SMOTE를 사용한다면 그 의미가 퇴색되기 때문이다. 그러나 oversampling 기법 중 과대적합을 피하는 SMOTE 방법과 이를 교차검증 내에서 수행했지만, 여전히 시험 데이터에 대해 과대적합이 보였다는 점은 추후 개선해야 할 점이라고 생각한다.