

카드사 이탈 고객 예측 머신러닝

모델 2조_김희진/김용호/문제공/박단치/안영훈/안성준/



multicampus



1.프로젝트 배경 2.팀 구성 및 역할 3.수행 절차 및 방법 4.결론 및 향후 광제 **5.** 느낀점

01. 프로젝트 배경

1.프로젝트

위한 머신러닝 모델링

2.프로젝트

- **개요** 카드 회사들은 고객 유치를 위해 많은 프로모션을 진행
- 새로운 고객을 유치하는 것보다 기존 고객을 유지하는 것이 경제적 효과큼
- 기존 고객의 이탈 여부를 사전에 예측 가능한 모델 구축

··· 굿모닝경제 2021.08.05.

카드사 실질회원 2%p 감소...휴면카드 1200만장 돌파

58%)의 경우 오히려 실질회원 비중이 커졌다. 휴면신용카드 수 증가 규모로도 하 나카드 18만2000장(22.6... 한편 신규 체크카드 발급 수도 올해 2분기 중 1조749...



01. 프로젝트 배경

3.프로젝트 구조





출처 : Kaggle



데이터 수집



- Numpy 및 Pandas
- Seaborn 및 matplotlib
- sklearn

데이터 처리 및 분석







Git 프로젝트 저장소



02. 팀 구성 및 역할

1.팀 구성 및 역할

김 희 진

- 데이터 **투잡**)
- 스케일링 / 전처리 함수화

김 용

- 지도학습 모델링
- ppt 디자인

문 세

- 수치형 변**수**전처리
- 상관관계 도출

박 민

- 범주형 변수 전처리
- 시각화

안 성

- 통계 분<mark>작</mark>
- 깃 헙 협업 관리

안 영

- 발표
- 하이퍼 파라미터 튜닝 및 성능 향상

1. 데이터 정보

데이터 정보

- ~'CLIENTNUM' : 고객 식별 번호~
- 'Attrition_Flag' : 신용 카드 이탈 여부 Target 값
 - Existing Customer : 잔류
 - Attrited Customer : 이탈
- 'Customer_Age' : 고객 나이
- 'Gender' : 성별
- 'Dependent_count' : 부양 가족 수
- 'Education_Level' : 학력 수준
- 'Marital_Status' : 결혼 여부
- 'Income_Category' : 연 소득 구간
- 'Card_Category' : 카드 등급
- 'Months_on_book' : 카드 할부 기간
- 'Total_Relationship_Count' : 가입 상품 수
- 'Months_Inactive_12_mon': 1년 동안 카드 결재 내역이 없는 비활성 기간(개월)
- 'Contacts_Count_12_mon' : 연락 빈도
- 'Credit_Limit' : 신용 한도
- 'Total_Revolving_Bal' : 할부 잔액
- ~'Avg_Open_To_Buy' : 평균 실 사용 가능 금액 : 'Credit_Limit' 'Total_Revolving_Bal'~
- ~'Total_Amt_Chng_Q4_Q1'~: 결제 대금 기준 1분기 대비 4분기 (비율)
- ~'Total_Trans_Amt'~: 실제 사용 총액
- 'Total_Trans_Ct' : 실제 사용 횟수
- 'Total_Ct_Chng_Q4_Q1': 1분기 대비 4분기 결제 대금 횟수 비율
- ~'Avg_Utilization_Ratio' : 'Total_Revolving_Bal'/ 'Credit_Limit' (할부 비율)~

2. 결측치 확인

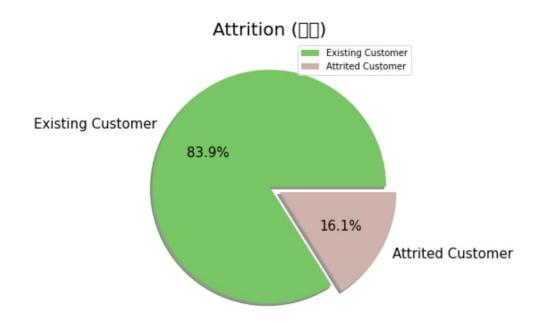
결측치 확인

- isnull().sum() 코드에서는 결측치가 존재하지 않는 것을 확인
- 각각의 피처를 분석해서 결측치 존재 여부 확인 필요

결측치 존재 X (각 피쳐마다 추가적인 정보 확인 필요) df.isnull().sum()

CLIENTNUM	0	
Attrition_Flag	0	
Customer_Age	0	
Gender	0	
Dependent_count	0	
Education_Level	0	
Marital_Status	0	
Income_Category	0	
Card_Category	0	
Months_on_book	0	
Total_Relationship_Count	0	
Months_Inactive_12_mon	0	
Contacts_Count_12_mon	0	
Credit_Limit	0	
Total_Revolving_Bal	0	
Avg_Open_To_Buy	0	
Total_Amt_Chng_Q4_Q1	0	
Total_Trans_Amt	0	
Total_Trans_Ct	0	
Total_Ct_Chng_Q4_Q1	0	
Avg_Utilization_Ratio	0	
dtype: int64		

3. 타겟 데이터 라벨 인코딩



레이블 인코딩

"Existing Cumstomer" : 0 (카드 잔존)"Attrited Customer" : 1 (카드 탈퇴)

주의하여 관측해아할 것이 "탈퇴"여부이기 때문에 탈퇴를 1, 잔존을 0으로 하여 인코딩

4. Unknown 값 결측치 처리

```
In [57]: df["Education Level"].replace({"Unknown":np.nan,
                                          "Graduate":0,
                                          "Post-Graduate":1,
                                          "Uneducated":2,
                                          "College":3,
                                          "Doctorate":4,
                                          "High School":5,
                                           },inplace=True)
In [58]: df["Marital_Status"].replace({"Unknown":np.nan,
                                          "Married":0,
                                          "Single":1,
                                          "Divorced":2,
                                           },inplace=True)
In [59]: df["Income Category"].replace({"Unknown":np.nan,
                                          "Less than $40K":0,
                                          "$40K - $60K":1,
                                          "$60K - $80K":2,
                                          "$80K - $120K":3,
                                          "$120K +":4,
                                           },inplace=True)
```

"Unknown"에 대한 처리 방법은

- 1. "Unknown"도 하나의 category로 해석
- 2. "Unknown"값이 있는 행을 삭제하거나, 칼럼 자체(피처)를 삭제
- 3. 모델링을 활용하여 대체
- 4. 최빈값으로 대체

여러 가지 시도후 최적화된 값 구하기

5. 결측치로 처리된 Unknown을 최빈값으로 대체

6. 수치형 변수 간 상관관계 파악 (1)

plt.subplots(figsize=(20,9))
plt.tick_params(axis='x',labelcolor='white')
plt.tick_params(axis='y',labelcolor='white')
sns.heatmap(df[Numerics].corr(),annot = True)

<AxesSubplot:>



상관계수는 0.79로 높지만, 삭제 X 각각의 피처 할부기간, 고객 나이를 의미 도메인 관점에서 분석하면 각각의 상관관계가 없기때문에 삭제하지 않음

- 2) 같은 변수에서 파생된 변수 들은 상관관계 낮더라도 drop
 - 1. 'Avg_Open_To_Buy' = 'Credit_Limit' 'Total_Revolving_Bal'
 - 2. 'Avg_Utilization_Ratio' = 'Total_Revolving_Bal'/ 'Credit_Limit'
 - 3. Total_Trans_Amt와 Total_Trans_Ct

6. 수치형 변수 간 상관관계 파악 (2) 파생변수와 타겟 변수간의 상관과계인 Attrition_Flag(범주형)과 수치형 피처 변수들의 상관관계를 파악

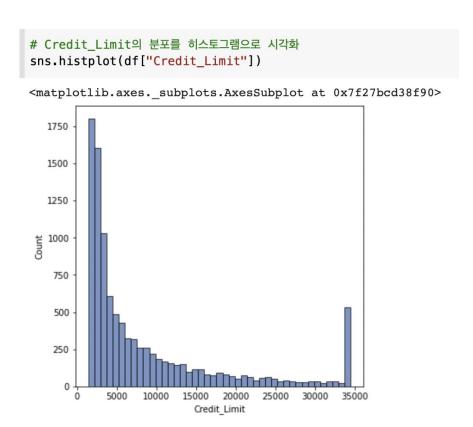
●범주형 - 수치형 상관계수를 파악하기 위해 pointbiserialr 사용

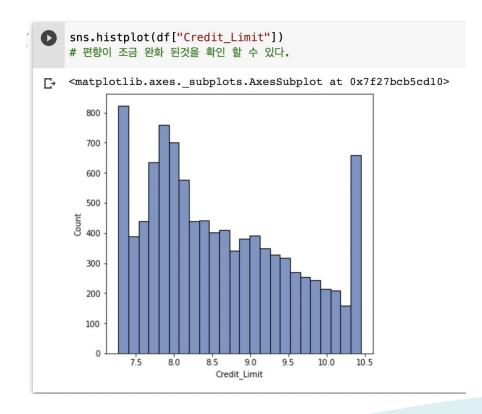
```
from scipy.stats import pointbiserialr
features = ['Avg_Open_To_Buy', 'Credit_Limit', 'Total_Revolving_Bal', 'Avg_Utilization_Ratio', 'Total_T
print("Attrition_Flag를 기준으로 상관관계 분석")
print('----
print('')
 for feature in features:
    target feature corr, target feature pvalue = pointbiserialr(df['Attrition Flag'], df[feature])
    print(feature, "와 Attrition Flag의")
    print('상관관계 : {0:.4f}'.format(target_feature_corr))
    print('P-value : {0:.8f}'.format(target_feature_pvalue))
    print('')
Attrition_Flag를 기준으로 상관관계 분석
Avg_Open_To_Buy 와 Attrition_Flag의
상관관계 : -0.0003
P-value : 0.97711609
Credit_Limit 와 Attrition_Flag의
상관관계 : -0.0239
P-value: 0.01628536
Total Revolving Bal 와 Attrition Flag의
사과과게 • _0 7631
```

6. 수치형 변수 간 상관관계 파악 (3) 타겟과의 상관도가 낮은 피처는 drop

7. 편향된 피처 데이터 로그 변환# 편향이 심한 Credit_Limit은 로그 변환을 해주도록한다.

df["Credit_Limit"]=np.log1p(df["Credit_Limit"])





8. Standard Scaling

수치형 변수를 표준화 스케일링을 진행

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

for numeric in ['Customer_Age','Credit_Limit','Total_Revolving_Bal','Total_Ct_Chng_Q4_Q1','Total_Trans_Ct']:
    df[numeric] = scaler.fit_transform(df[numeric].values.reshape(-1,1))
```

수치형 데이터 표준화 확인 df[Numerics]

Customer_Age	Dependent_count	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit	Total_Revo
-0.165406	3.0	39.0	5.0	1.0	3.0	0.905210	-0.473422
0.333570	5.0	44.0	6.0	1.0	2.0	0.444695	-0.366667
0.583058	3.0	36.0	4.0	1.0	0.0	-0.499786	-1.426858
-0.789126	4.0	34.0	3.0	4.0	1.0	-0.533199	1.661686
-0.789126	3.0	21.0	5.0	1.0	0.0	-0.155051	-1.426858
						•••	
9.458314	2.0	40.0	3.0	2.0	3.0	-0.330596	0.844455
-0.664382	2.0	25.0	4.0	2.0	3.0	-0.259693	1.255524
-0.290150	1.0	36.0	5.0	3.0	4.0	-0.008217	-1.426858
-2.036565	2.0	36.0	4.0	3.0	3.0	-0.033865	-1.426858
-0.414894	2.0	25.0	6.0	2.0	4.0	0.690732	0.979433

8. 전처리 과정을 함수화

피처엔지니어링 함수화

```
# Unknown을 결측치로 처리 함수
def Unknown to nan(df):
   df.replace({"Unknown": np.nan},inplace=True) # 데이터 전체에 존재하는 "Unknown"을 결측치로 처리
    return df
# 레이블 인코딩 수행 함수
def format features(df):
    from sklearn.preprocessing import LabelEncoder
    features=['Education Level','Income Category','Card Category','Marital Status','Gender']
    for feature in features:
           le = LabelEncoder()
           le = le.fit(df[feature])
           df[feature] = le.transform(df[feature])
    return df
# # Unknown도 무응답으로 인지 후 더미화
# def format features(df):
     features=['Education_Level','Income_Category','Card_Category','Marital_Status','Gender']
     for feature in features:
         feature OH = pd.get dummies(df[feature])
         df = pd.concat([df, feature OH], axis=1)
         df.drop(feature,axis=1,inplace=True)
     return df
# 결측치를 대표값으로 대체하는 함수
def fill(df):
    # 대표값 이용 결측치 대체 모듈
    from sklearn.impute import SimpleImputer
    # 각 데이터에 사용할 인스턴스 생성
    SI mode =SimpleImputer(strategy = 'most frequent') # 대표값 중 최빈값으로 결측치를 대체해준다.
    SI mode.fit(df)
    df = pd.DataFrame(SI mode.transform(df),
                          columns = df.columns)
    return df
# 데이타 타입을 수치형으로 바꿔주는 함수
def toNumerics(df):
    df[Numerics]=df[Numerics].astype("float")
    df[Labels]=df[Labels].astype("int")
    df[Orders]=df[Orders].astype("int")
    df["Attrition_Flag"]=df["Attrition_Flag"].astype("int")
    return df
```

```
# 머신러닝 알고리즘에 불필요한 속성 제거
def drop features(df):
   df.drop(["CLIENTNUM", # 식별자 삭제
        "Avg Open To Buy",
         "Avg Utilization Ratio",
         "Total Trans Amt",
         "Total Amt Chng Q4 Q1"], axis=1,inplace=True)
   return df
# 로그변환 함수
def log transformation(df):
  df["Credit_Limit"]=np.log1p(df["Credit_Limit"])
  return df
# 수치형 변수 표준화 스케일링
def Standared scaling(df):
    from sklearn.preprocessing import StandardScaler
   scaler = StandardScaler()
   for numeric in ['Customer Age', 'Credit Limit', 'Total Revolving Bal', 'Total Ct Chng Q4 Q1', 'Total Trans Ct']:
       df[numeric] = scaler.fit_transform(df[numeric].values.reshape(-1,1))
   return df
# 전처리 함수 호출
def transform features(df):
   #df = Unknown to nan(df)
   df = format features(df)
   df = fill(df)
   df = toNumerics(df)
   df = drop features(df)
   df = log transformation(df)
   df = Standared_scaling(df)
   return df
```

향후 입력되는 데이터를 바로 전처리 할 수 있도록 전처리 과정을 하나의 함수화

10. 모델링 (1) Score

from sklearn.metrics import roc auc score

```
# 수정된 get clf eval() 함수
def get clf eval(y test, pred=None, pred proba=None):
   confusion = confusion matrix(y test, pred)
   accuracy = accuracy score(y test , pred)
   precision = precision score(y test, pred)
   recall = recall score(y test, pred)
   f1 = f1 score(y test,pred)
     # ROC-AUC 추가
    roc auc = roc auc score(y test, pred proba)
   print('오차 행렬')
   print(confusion)
   # ROC-AUC print 추가
   print('정확도: {0:.4f}, 정밀도: {1:.4f}, 재현율: {2:.4f},\
    F1: {3:.4f}'.format(accuracy, precision, recall, f1)) #, roc auc))
# 테스트를 수행할 모든 임곗값을 리스트 객체로 저장.
thresholds = [0.05, 0.11, 0.2, 0.3, 0.4, 0.45, 0.50, 0.55, 0.60]
def get eval by threshold(y test , pred proba c1, thresholds):
   # thresholds list객체내의 값을 차례로 iteration하면서 Evaluation 수행.
   for custom threshold in thresholds:
       binarizer = Binarizer(threshold=custom threshold).fit(pred proba c1)
       custom predict = binarizer.transform(pred proba c1)
       print('\n임곗값:',custom threshold)
       get clf eval(y test , custom predict)
```

from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recal get_clf_eval 함수

모델의 성능 (정확도, 재현율, 정밀도, F1)을 측정할 함수를 미리 설정

성능 측정 지표 설정 기준

정확도 신뢰 불가

우리가 선택한 데이터는 타겟 비율이 85:15 정도로 편향이 있어서 정확도는 높게 나올수 밖에 없는 데이터라 정확도는 신뢰할 수 없기 때문에 정확도 보다는 재현율과 정밀도를 관찰하였다.

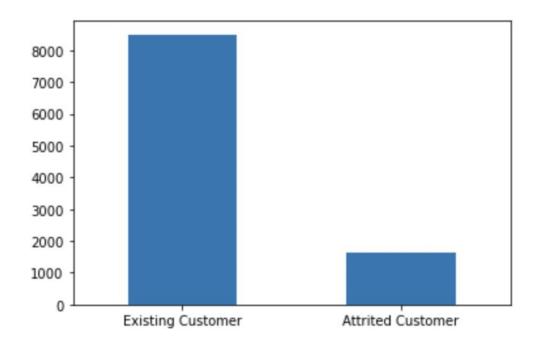
정밀도 0.75 유지 & 재현율을 높이는 방향

잔존고객을 이탈고객으로 판단하였을 경우에는 손실이 없으나, 이탈고객 (양성)을 잔존(음성)으로 잘못 판단하였을 경우에는 손실이 크기 때문에, 재현율을 중점으로 관측하였다. 정밀도가 0.75이상이면서도 재현율의 성능을 높이기 위해 하이퍼 파라미터를 조정하였다.

10. 모델링 (1) 기본 모델링

	정확도	재현율	정밀도
DecisionTreeClassifier	0.8890	0.6591	0.6413
RandomForestClassifier	0.9246	0.8750	0.6192
LogisticRegression	0.8965	0.7736	0.5037
KNeighborsClassifier	0.8894	0.8159	0.4029
LGBMClassifier	0.9293	0.8393	0.6929
XGBClassifier	0.9273	0.8369	0.6806
SVC	0.8867	0.8659	0.3489

10. 모델링 (2) 업샘플링 (smote)



타겟 변수의 비율이 85:15 로 한쪽으로 치우쳐져 있기 때문에 업샘플링이나 다운샘플링으로 비율을 1:1로 맞춤. 데이터가 만개 정도로 크지 않기 때문에 업샘플링은 진행

10. 모델링 (1) 기본 모델링

	정확도	재현율	정밀도
DecisionTreeClassifier	0.8890	0.6591	0.6413
RandomForestClassifier	0.9246	0.8750	0.6192
LogisticRegression	0.8965	0.7736	0.5037
KNeighborsClassifier	0.8894	0.8159	0.4029
LGBMClassifier	0.9293	0.8393	0.6929
XGBClassifier	0.9273	0.8369	0.6806
SVC	0.8867	0.8659	0.3489

Thank U

데이터 출처

- 프로젝트 GIT 저장소 : <u>https://github.com/Ahn-seongjun/DS_team2</u>
- < 데이터 및 코드 >
- KAGGLE: https://www.kaggle.com/sakshigoyal7/credit-card-customers