분류

캐글 경진 대회

은행 이탈 데이터를 사용한 이진 분류 분석

1. 프로젝트 개요

1-1. 주제

- 은행 이탈 데이터 변수 간의 관계 영향 분석 및 시사점 도출

1-2. 주제 선정의 배경

- **배경**
  - 디지털 전환으로 인해 은행에서는 고객 이탈 현상이 대거 발생하고 있다. 경쟁이 치열해지고 있는 금융 시장에서 이러한 변화에 발맞추지 못한 은행들은 고객 이탈로 인한 막대한 손실을 겪고 있으며, 이를 해결하지 못한 은행은 결국 '죽게'된다는 전망도 보인다. 그러므로 은행은 고객들의 만족도를 높이며 고객 이탈을 최소화하는 것이 필수적으로 보임.

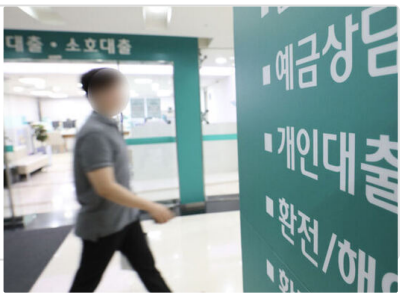
▼ 디지털 전환으로 업종 간 경계가 모호해지고 경쟁이 본격화된 가운데 전통적인 은행의 마케팅 비용은 증가했지만, 고객이 이탈하는 상황에 직면했음.

출처: 뉴데일리경제(23.11)

은행권 '조용한 이탈' 증가... "초개인화 서비스 강화해야"

2024년 은행의 최우선 과제로 예금 확보가 꼽혔다. 지난 2020년만 해도 주요 과제로 부각됐던 대출 증대, 브랜드 인지도 강화 등의 항목보다 예금 확보가 최우선 순위로 부상한 것이다.28일

https://biz.newdaily.co.kr/site/data/html/2023/11/28/2023112800069.html



▼ 디지털 채널 경쟁의 본격화로 고객 이탈이 우려된다는 전망이 나왔다. 국내 은행들은 디지털 경쟁에서 지면 금융 상품의 단순 제조자로 전락할 수 있으므로 경쟁력을 키워야 함.

출처: 서울경제(21.01)

https://www.sedaily.com/NewsView/22H436FENS

▼ 대형 은행이 이러한 변화에 적응하지 못하면 결국 죽게됨.

출처: CIOkorea뉴스(18.02)

"돈 되는 고객이 먼저 이탈한다"… 오픈뱅킹 속 위기의 대형은행

대형 은행이 오픈 뱅킹(Open Banking) 때문에 핵심 고객을 잃게 될 위기에 처했다는 분석이 나왔다. 베인앤드컴퍼니(Bain &

https://www.ciokorea.com/news/37389

- **목적**
  - 고객 이탈을 예측해 주는 모델을 만들어 은행이 고객 이탈을 최소화하고 어떤 상황에서 이탈이 발생하는지 분석하여 경쟁력을 유지하는데 필요한 전략을 논의하고자 함.

1-3. 본 프로젝트의 활용 방안 제시

- **고객 관리 및 예측**
  - 은행 고객 이탈 분류 모델 분석을 통해 고객 이탈에 영향을 주는 요인을 구분하여 가능성을 예측하고, 이를 기반으로 고객 관리 전략을 수립할 수 있음.
- **고객의 주인의식을 향상시키는 전략 수립**
- **마케팅 및 고객 응대 전략의 개선**
  - 분석 모델을 통해 마케팅 및 고객 응대 전략을 수립 및 개선할 수 있음.
    - 연령 별 은행 상품 기획 및 추천
    - 잔고 별 고객이 이용하는 은행 상품(예금, 적금 등) 추천
    - 고객의 개개인의 특성을 고려해 다양한 상품을 기획

2. 프로젝트 수행 절차 및 방법

2-1. 데이터 설명

- 기본 변수 설명

- Feature Data
  - id : 식별 번호
  - Surname : 고객의 성(씨)
  - Customer ID: 각 고객의 고유 식별 번호
  - Credit Score: 고객의 신용 점수
  - Geography: 고객이 거주하는 국가
  - Gender: 고객의 성별
  - Age: 고객의 나이
  - Tenure: 고객이 은행을 이용한 기간
  - Balance: 고객의 계좌 잔액
  - NumOfProducts: 고객이 이용하는 은행 상품의 수
  - HasCrCard: 신용카드 보유 여부
  - IsActiveMember: 활성 회원 여부
  - EstimatedSalary: 고객의 예상 연봉
- Target Data
  - Exited : 고객 이탈 여부

- 파생 변수 생성 계획

  : 고객 이탈 변수에 영향을 끼치는 그리고 끼치지 않는 변수를 예측, 구분

  : 영향을 미칠 것 같은 요인에서 조합하여 파생 변수 생성

  : 영향을 미칠 것 같은 요인 + 없을 것 같은 요인 조합하여 파생 변수 생성

  : Feature Importance + permutation importance + 상관관계 히트맵을 이용하여 제거 혹은 연관성 생성

  1. 이탈 고객에 영향을 미칠 것 같은 요인
     a. CreditScore (고객의 신용 점수)
     b. Age (고객의 나이)
     c. Tenure (고객이 은행을 이용한 기간)

d. NumOfProducts (고객이 이용하는 은행 상품의 수)
e. HasCrCard (신용카드 보유 여부)
f. EstimatedSalary (고객의 예상 연봉)

2. 이탈 고객에 영향이 없을 것 같은 요인
   a. id (식별 번호)
   b. CustomerID (각 고객의 고유 식별 번호)
   c. Surname (고객의 성(씨))
   d. Geography (고객이 거주하는 국가)
   e. Balance (고객의 계좌 잔액)
   f. IsActiveMember (활성 회원 여부)

3. 데이터 전처리

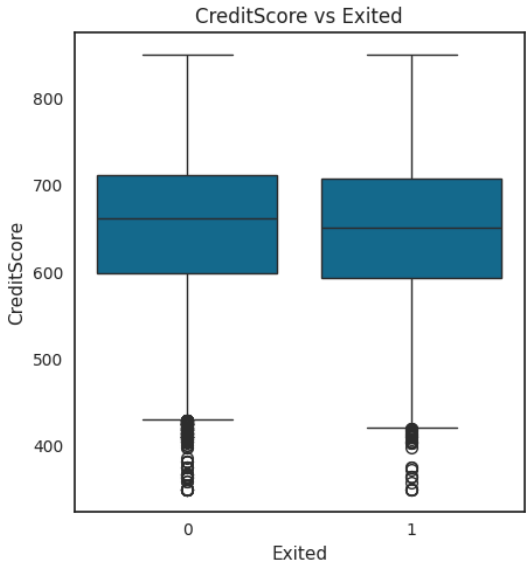
3-1. 데이터 전처리 계획

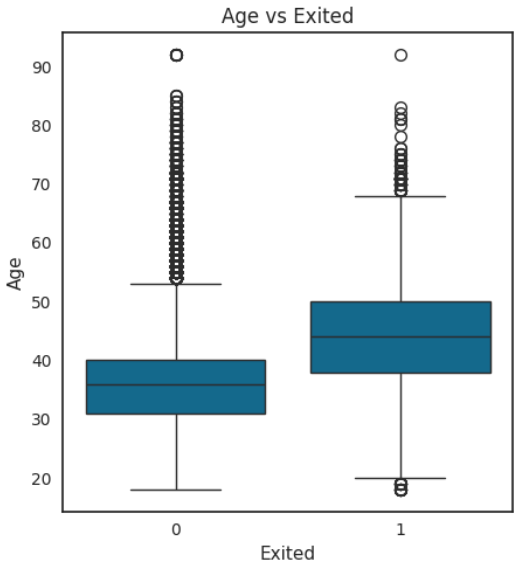
- 변수 제거
  - 불필요한 피처 제거
    - id, CustomerId, Surname 제거
      - id : 인덱스 역할
      - CustomerId : 고객 당 갖게 되는 고유 Id
        - 전체 데이터 165034개 중 고유값 23221 존재
        - 따라서 고유값 기능 상실
      - Surname : 고객의 성(씨)
- 중복 값 제거
  - 데이터 원본(165034개)
    - Surname
      - Surname 고윳값 개수 : 2237개
      - Surname만 중복 값 개수 탐색 : 162237개
      - Surname이 중복 값인 행 중 Balance가 0인 행 : 88318개 (절반 차지)

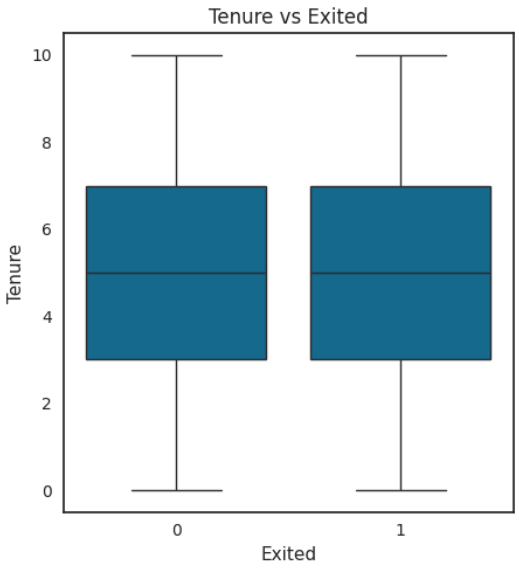
        - CustomerId
            - CustomerId 고윳값 개수 : 23221개
            - CustomerId만 중복 값 개수 탐색 : 141813개
            - CustomerId가 중복 값인 행 중 Balance가 0인 행 : 77810개 (절반 차지)
    - id, Customer 제거 후 Surname + 나머지 값들 모두 동일한 값 중복 값으로 처리 후 제거
    - 완벽한 중복 값 처리가 아닌 것이므로 추후 재확인 예정

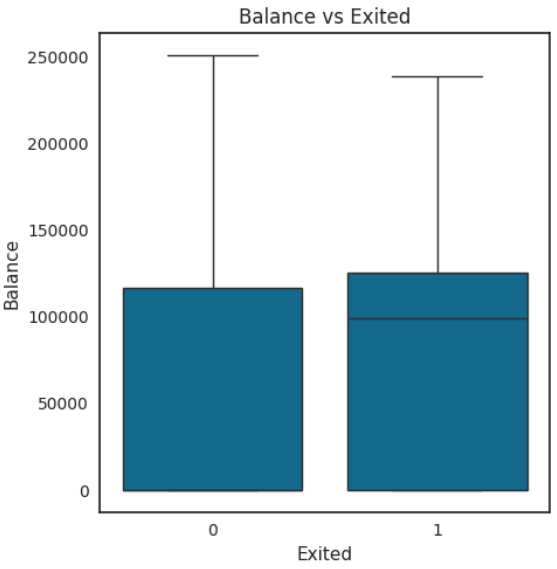
- 변수 변환(type, 단위)
    - HasCrCard, IsActiveMember ：고윳값 1.0 / 0.0
        - 값 int로 변경
    - Balance : float
        - 값 반올림 후 int로 type 변환
            - 품질 향상
    - Age ：float
        - (Age*10) 후 int로 type 변환
            - Age에 존재하는 소수점이 의미가 있을 가능성
            - 품질 향상
            - 차이 극대화
    - EstimatedSalary : float
        - (EstimatedSalary*100) 반올림 후 int로 type 변환
            - 품질 향상
            - 차이 극대화
- 데이터 인코딩
    - ‘Gender’, ‘Geography’ - OneHot (pd.get_dummies) or 직접 변경
        - 고윳값 개수가 많지 않고 값의 순서가 없음으로 OneHot이 적절한 것으로 판단
        - 파생 변수 생성을 위해 OneHot이 아닌 직접 인코딩

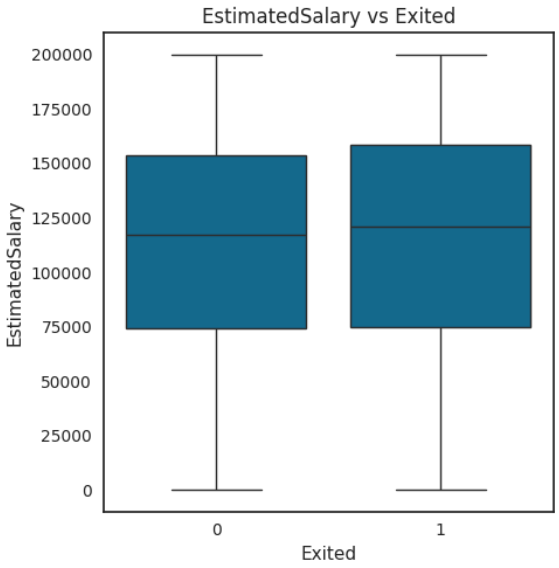
        - 추후에 파생 변수 이용을 위해 인코딩 값 직접 설정
            - Gender의 고윳값 개수 : 2 (‘Male’, ‘Female’)
            - Geography의 고윳값 개수 : 3 (’France’,’Germany’,’Spain’)
- 데이터 스케일링
    - EDA 시각화 및 피처 요약표를 통한 이상치 유무 확인 및 데이터 분포 파악하여 스케일
        - RobustScaler : Age, CreditScore
        - MinMaxScaler : Tenure, EstimatedSalary
        - StandardScaler : Balance + 파생 변수들('CreditScoreByAge' , 'TenureByAge' , 'Engagement' , 'CreditScoreByBalance' , 'BalanceByAge' , 'NumOfProductsByBalance')

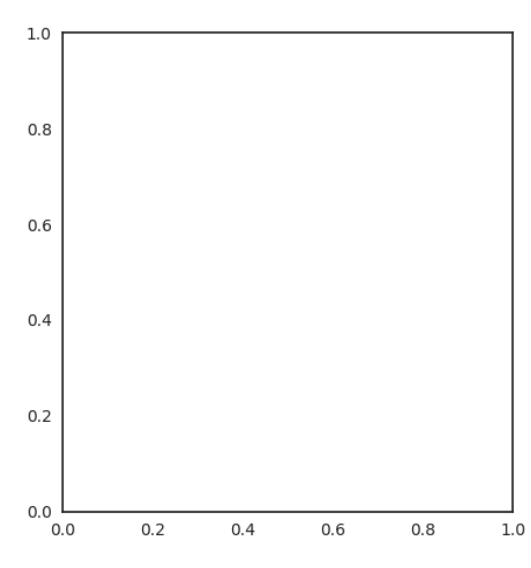

CreditScore vs Exited
CreditScore
800
700
600
500
400
0
1
Exited

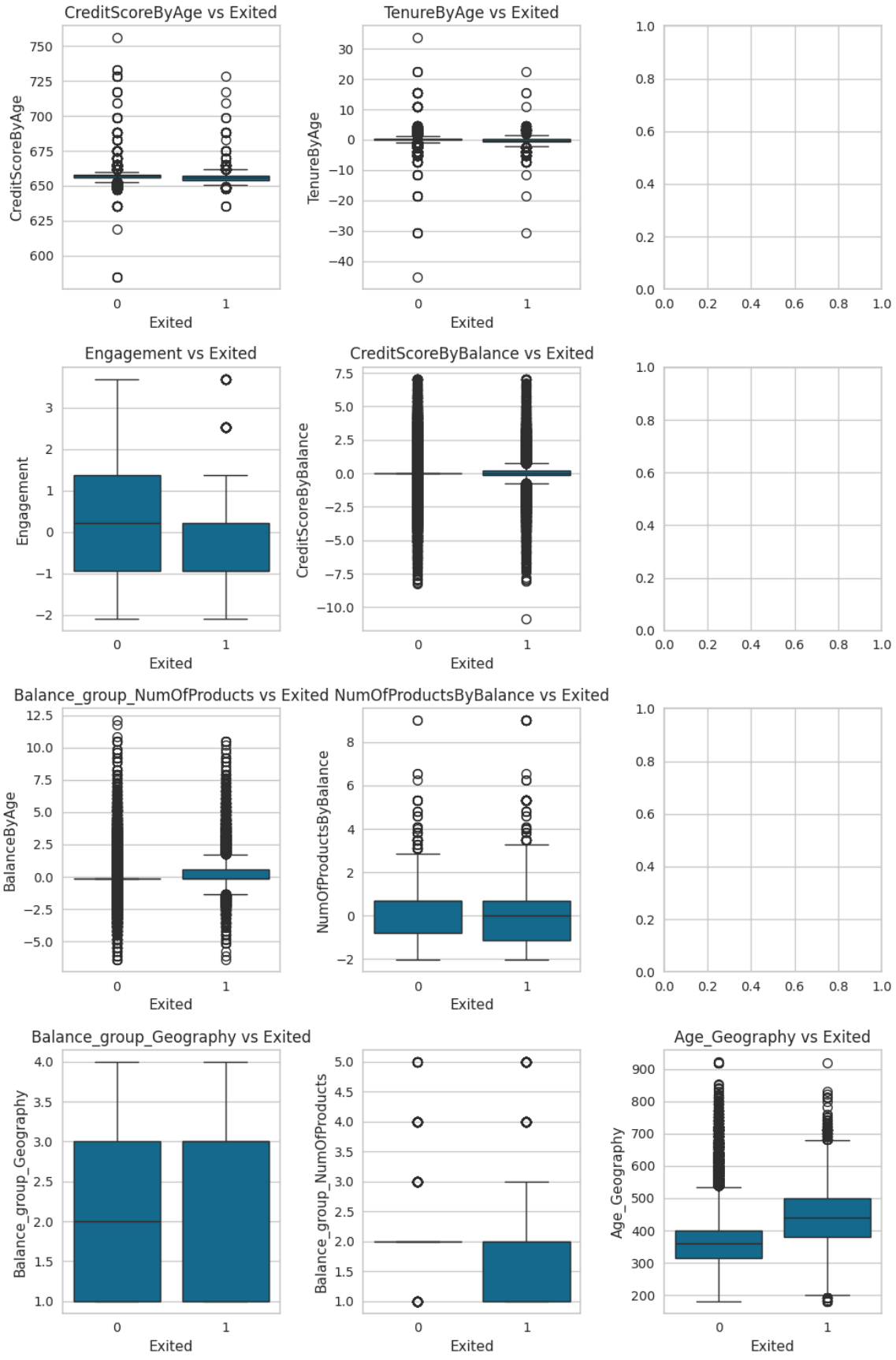

Age vs Exited
Age
90
80
70
60
50
40
30
20
0
1
Exited


Tenure vs Exited
Tenure
10
8
6
4
2
0
0
1
Exited


Balance vs Exited
Balance
250000
200000
150000
100000
50000
0
0
1
Exited


EstimatedSalary vs Exited
EstimatedSalary
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
Exited


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


CreditScoreByAge vs Exited
CreditScoreByAge
Exited
TenureByAge vs Exited
TenureByAge
Exited
Engagement vs Exited
Engagement
Exited
CreditScoreByBalance vs Exited
CreditScoreByBalance
Exited
Balance_group_NumOfProducts vs Exited
BalanceByAge
Exited
NumOfProductsByBalance vs Exited
NumOfProductsByBalance
Exited
Balance_group_Geography vs Exited
Balance_group_Geography
Exited
Balance_group_NumOfProducts
Exited
Age_Geography vs Exited
Age_Geography
Exited

- 파생 변수 생성

- CreditScore By Age : 나이대 별 신용점수
- TenureByAge : 연령 별 고객이 은행을 이용한 기간
- Engagement : NumOfProducts(1,2,3,4) + HasCrCard(0,1) + IsActveMember(0,1)

  -> 1, 2, 3, 4, 5, 6 : 참여도(값이 낮을수록 낮음)
- CreditScore By (Balance or Balance_group) : 잔고 별 신용점수
- Balance By Age : 연령 별 잔고
- NumOfProducts By (Balance or Balance_group) : 잔고 별 등 고객이 이용하는 상품의 수

- 모델링
  - AutoML을 통한 Top5 알고리즘 결정

3-2. 데이터 샘플

▼ Shape

- train.shape : `(165034, 14)`
- test.shape `(110023, 13)`
- submission.shape `(110023, 2)`

```
1 train.shape, test.shape, submission.shape
```

```
((165034, 14), (110023, 13), (110023, 2))
```

▼ Train

	id	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
0	0	15674932	Okwudilichukwu	668	France	Male	33.0	3
1	1	15749177	Okwudiliolisa	627	France	Male	33.0	1
2	2	15694510	Hsueh	678	France	Male	40.0	10
3	3	15741417	Kao	581	France	Male	34.0	2
4	4	15766172	Chiemenam	716	Spain	Male	33.0	5

Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0.00	2	1.0	0.0	181449.97	0
0.00	2	1.0	1.0	49503.50	0
0.00	2	1.0	0.0	184866.69	0
148882.54	1	1.0	1.0	84560.88	0
0.00	2	1.0	1.0	15068.83	0

▼ Test

	id	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
0	165034	15773898	Lucchese	586	France	Female	23.0	2
1	165035	15782418	Nott	683	France	Female	46.0	2
2	165036	15807120	K?	656	France	Female	34.0	7
3	165037	15808905	O'Donnell	681	France	Male	36.0	8
4	165038	15607314	Higgins	752	Germany	Male	38.0	10

Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0.00	2	0.0	1.0	160976.75
0.00	1	1.0	0.0	72549.27
0.00	2	1.0	0.0	138882.09
0.00	1	1.0	0.0	113931.57
121263.62	1	1.0	0.0	139431.00

▼ submission

id	Exited
165034	0.5
165035	0.5
165036	0.5
165037	0.5
165038	0.5

3-3. 데이터 수집 및 전처리

- 변수 제거
  - Id
  - CustomerId
  - Surname
- 중복 값 제거
  - 전체 중복치 제거
- 변수 변환
  - Float → int
    - HasCrCard
    - IsActiveMember
    - Balance
      - 반올림
    - Age
      - *10
    - EstimatedSalary
      - *100
- 데이터 인코딩
  - Gender
    - OneHot
  - Geography
    - OneHot
    - LabelEncoder
- 데이터 스케일링
  - 이상치 개수에 따른 스케일링 방법 조정
    - Standard
    - Robust
    - MinMax

- 파생 변수 생성
  - 총 생성 파생 변수
    - age_group
    - Balance_group
    - CreditScoreByAge
    - CreditScoreByBalance
    - TenureByAge
    - BalanceByAge
    - Engagement
    - NumOfProductsByBalance
    - Balance_group_Geography
    - Balance_group_NumOfProducts
    - Age_Geography

3-4. 활용 라이브러리 등 기술적 요소

- Pandas
- Numpy
- Matplotlib
- Seaborn
- Pycaret
- Sklearn
- LGBMClassifier
- CatBoostClassifier
- XGBClassifier
- GradientBoostingClassifier
- AdaBoostClassifier
- GridSearchCV
- BayesianOptimization

3-5. 프로젝트에서 분석한 내용

▼ 결측치 확인

- train.isna( ).sum( ).sum( ) - 0

```
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   id               165034 non-null  int64
 1   CustomerId       165034 non-null  int64
 2   Surname          165034 non-null  object
 3   CreditScore      165034 non-null  int64
 4   Geography        165034 non-null  object
 5   Gender           165034 non-null  object
 6   Age              165034 non-null  float64
 7   Tenure           165034 non-null  int64
 8   Balance          165034 non-null  float64
 9   NumOfProducts    165034 non-null  int64
 10  HasCrCard        165034 non-null  float64
 11  IsActiveMember   165034 non-null  float64
 12  EstimatedSalary  165034 non-null  float64
 13  Exited           165034 non-null  int64
dtypes: float64(5), int64(6), object(3)
```

- test.isna( ).sum( ).sum( ) - 0

```
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   id               110023 non-null  int64
 1   CustomerId       110023 non-null  int64
 2   Surname          110023 non-null  object
 3   CreditScore      110023 non-null  int64
 4   Geography        110023 non-null  object
 5   Gender           110023 non-null  object
 6   Age              110023 non-null  float64
 7   Tenure           110023 non-null  int64
 8   Balance          110023 non-null  float64
 9   NumOfProducts    110023 non-null  int64
 10  HasCrCard        110023 non-null  float64
 11  IsActiveMember   110023 non-null  float64
 12  EstimatedSalary  110023 non-null  float64
dtypes: float64(5), int64(5), object(3)
```

- submission.isna( ).sum( ).sum( ) - 0

```
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   id      110023 non-null  int64
 1   Exited  110023 non-null  float64
dtypes: float64(1), int64(1)
```

▼ 중복 값 확인

- 전체 데이터
  - 없음
- id, CustomerId, Surname 제거

```
# id, CUstomerId, Surname 제거
train.drop(columns=['id','CustomerId','Surname'],inplace=True)
# 중복 값  제거 - 123개
train[train.duplicated()]

# 중복 값 중 Balance의 고윳값 개수
train[train.duplicated()]['Balance'].value_counts()
# 123개 중
# 0이 69개
# 69/123 -> 0.5609 -> 절반 차지
```

- id, CustomerId 제거

```
# id, CUstomerId 제거
train.drop(columns=['id','CustomerId'],inplace=True)
# 중복 값 제거 - 54개
train[train.duplicated()]

# 중복 값 중 Balance의 고윳값 개수
train[train.duplicated()]['Balance'].value_counts()
# 54개 중
# 0이 36개 나머지는 다 1개씩
# 36/54 -> 0.66 -> 절반 이상 차지
```

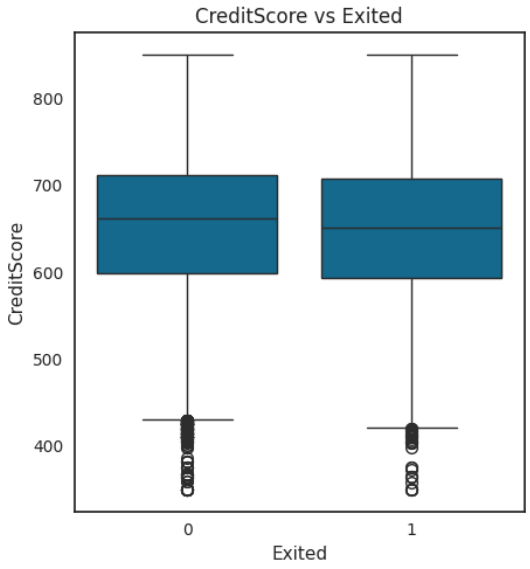
▼ 데이터 타입 확인

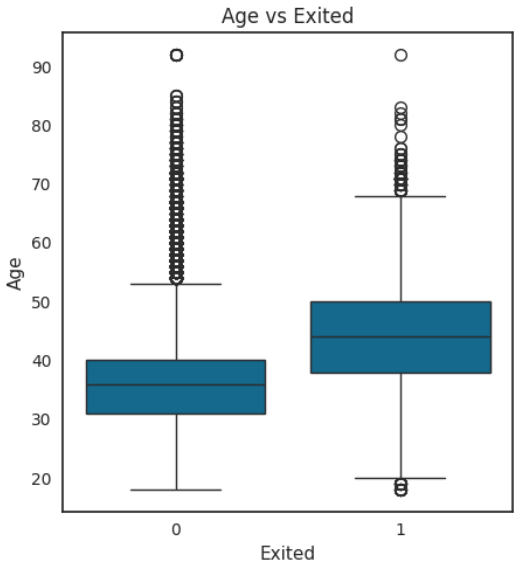
- 피처 요약표 확인
  - 결측치 존재하지 않음.
  - Object 타입 인코딩
    - Geography, Gender
  - float64 → int64 타입 변환

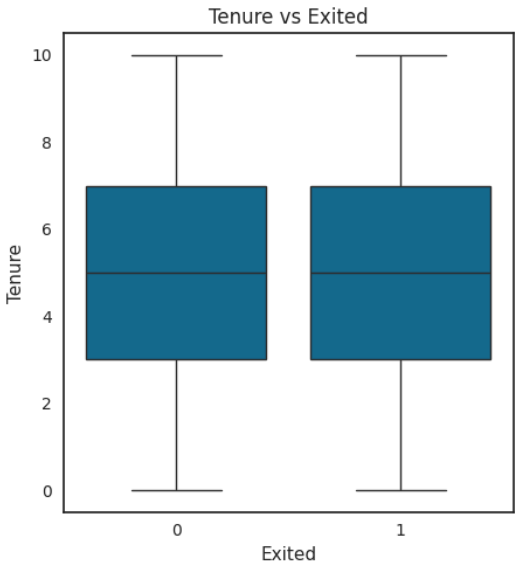
- HasCrCard, IsActiveMember
- Age * 10
  - *10 : 품질 향상, 소수점 의미 존재 가능성 확인
- Balance
  - 반올림 포함
- EstimatedSalary * 100
  - *100 : 품질 향상

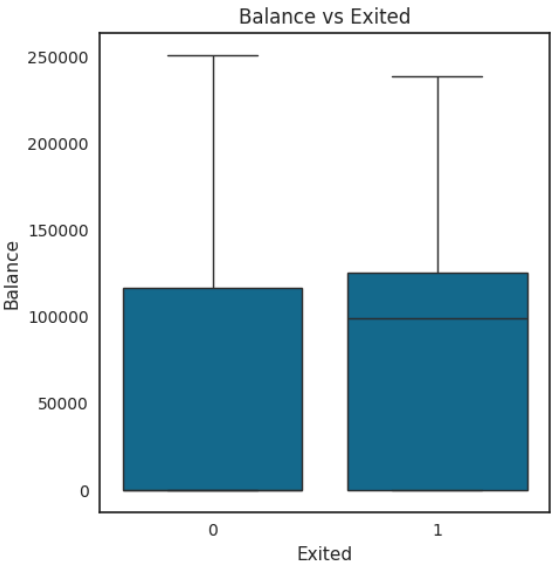
	피처명	타입	결측치수	고유값수	샘플_0	샘플_1	샘플_2
0	id	int64	0	165034	0	1	2
1	CustomerId	int64	0	23221	15674932	15749177	15694510
2	Surname	object	0	2797	Okwudilichukwu	Okwudiliolisa	Hsueh
3	CreditScore	int64	0	457	668	627	678
4	Geography	object	0	3	France	France	France
5	Gender	object	0	2	Male	Male	Male
6	Age	float64	0	71	33.0	33.0	40.0
7	Tenure	int64	0	11	3	1	10
8	Balance	float64	0	30075	0.0	0.0	0.0
9	NumOfProducts	int64	0	4	2	2	2
10	HasCrCard	float64	0	2	1.0	1.0	1.0
11	IsActiveMember	float64	0	2	0.0	1.0	0.0
12	EstimatedSalary	float64	0	55298	181449.97	49503.5	184866.69
13	Exited	int64	0	2	0	0	0

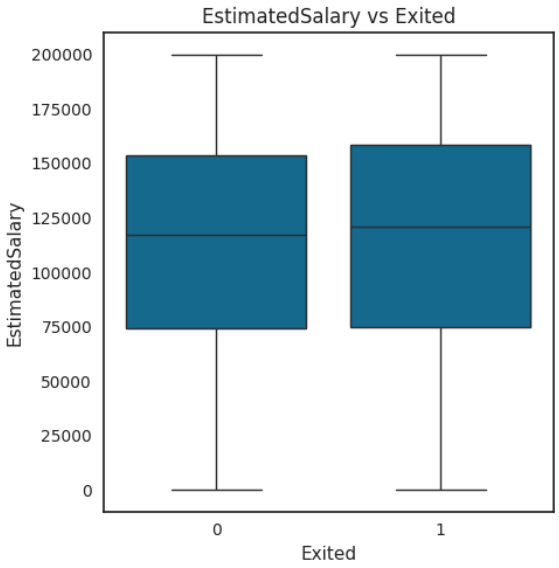
▼ 이상치 확인

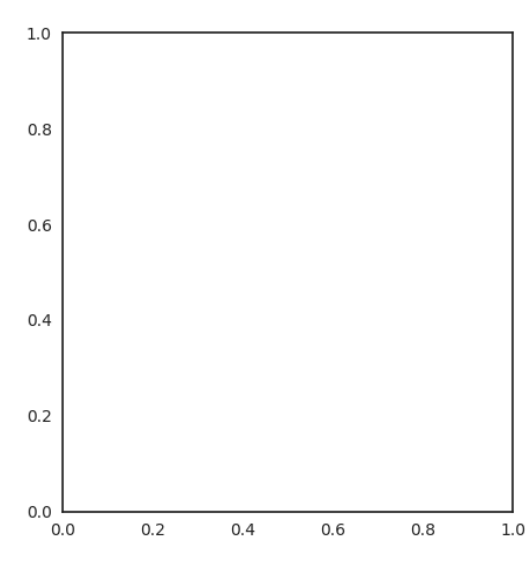

CreditScore vs Exited
CreditScore
800
700
600
500
400
0
1
Exited

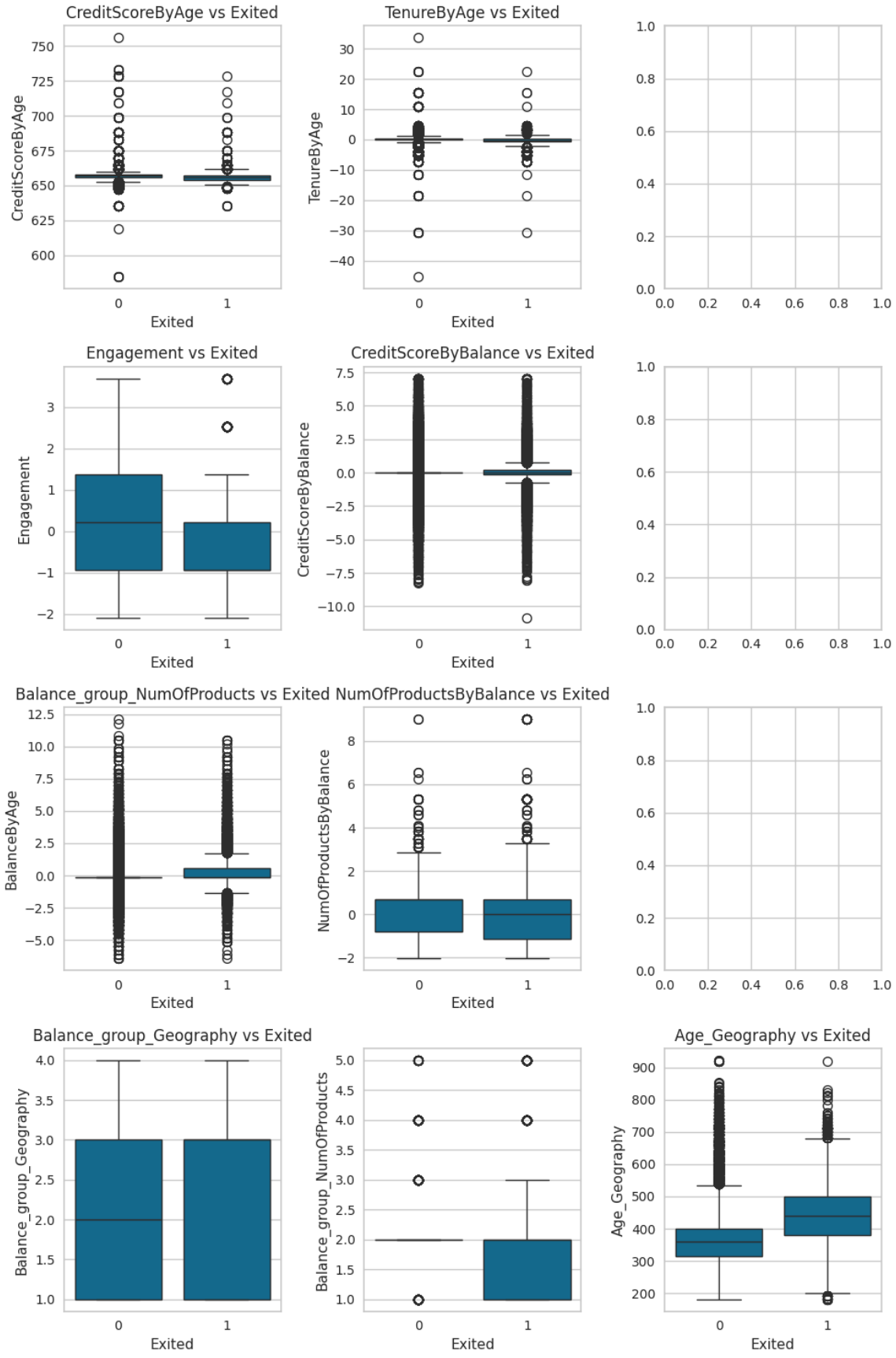

Age vs Exited
Age
90
80
70
60
50
40
30
20
0
1
Exited


Tenure vs Exited
Tenure
10
8
6
4
2
0
0
1
Exited

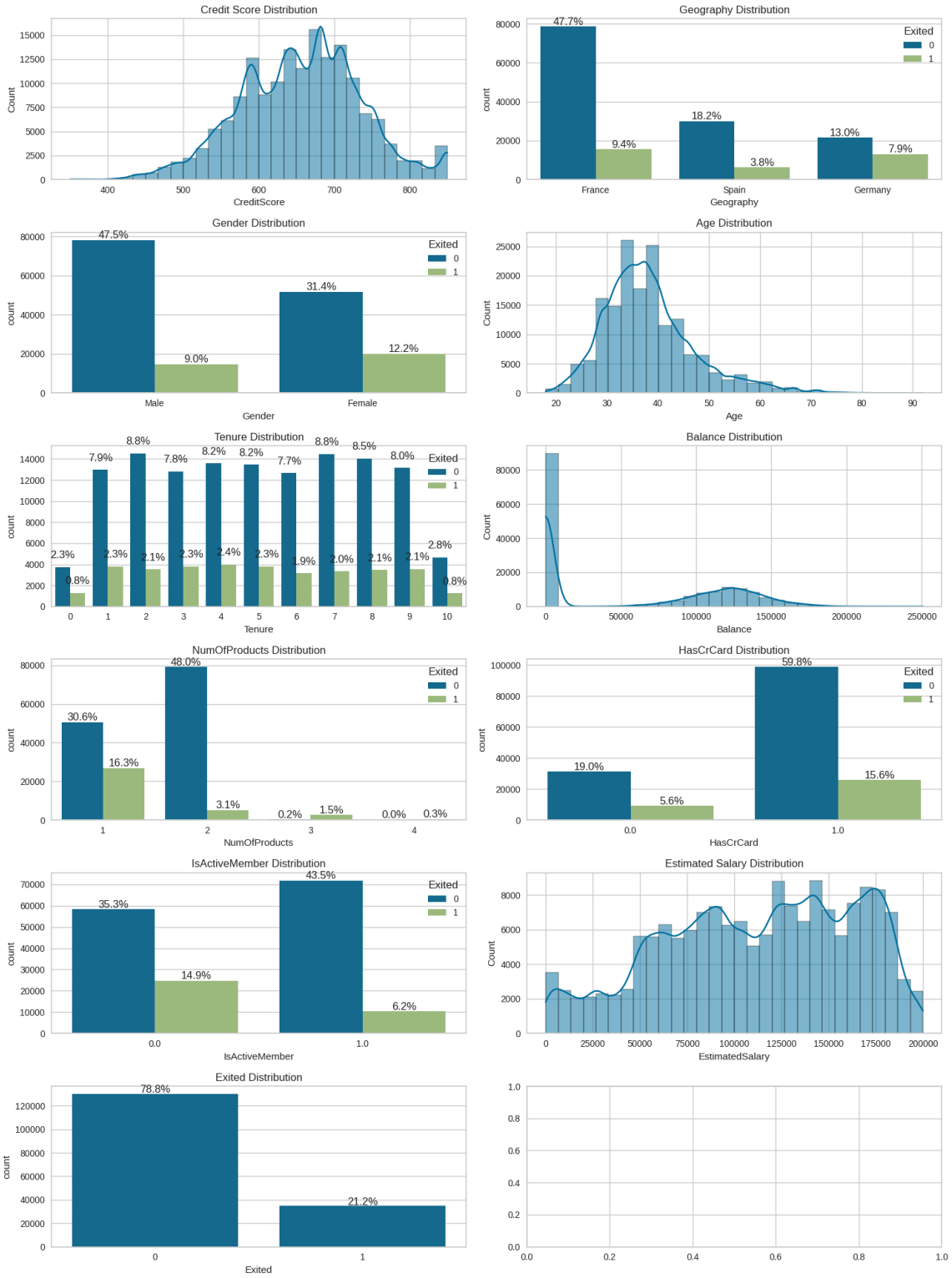

Balance vs Exited
Balance
250000
200000
150000
100000
50000
0
0
1
Exited


EstimatedSalary vs Exited
EstimatedSalary
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
Exited


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0



▼ 전체 수치 변수의 히스토그램 그리기



4. 기초 평가 및 분석

4-1. 탐색 분석(EDA)

▼ 시각화 분석

- Exited (고유값 : 0, 1) - 수치(범주형)

    - 0 : 1 = 80 : 20로 target data 층화에 대한 필요성
- CreditScore - 수치(연속형)
    - 대체로 정규분포 형태로 인식 가능
    - 600 - 700 데이터가 밀집
    - 400 - 500 상대적으로 데이터 부족
    - low, normal, high 범주형 변수로 변환 가능성 확인
- Geography (고유값 : France, Spain, Germany) - 문자
    - France : Spain : Germany = 60 : 20 : 20
    - target data와 차이가 존재
- Gender (고유값 : Male, Female) - 문자
    - Male : Female = 60 : 40
    - target data와 차이가 존재
- Age - 수치(연속)
    - 대체로 정규분포 형태로 인식 가능
    - 30대에 데이터가 몰려있음
    - 나이가 50세 이상부터 상대적으로 데이터가 부족
- Tenure (고유값 0 - 10) - 수치(연속|범주)
    - 0년과 10년을 제외하고 비슷한 수치
    - target data와 차이가 존재
- Balance - 수치(연속)
    - 0의 데이터가 압도적으로 많이 분포
    - 0을 제외하고 대체로 정규분포 형태 인식 가능
    - 0을 이상치로 인식 가능성 여부 고려
- NumOfProducts(고유값 : 1, 2, 3, 4) - 수치(범주)
    - 1 : 2 : 3 : 4 = 50 : 50 : 0 : 0
        - 3과 4의 데이터 크기가 작음
- HasCrCard (고유값 : 0, 1) - 수치(범주)

    - 0 : 1 = 25 : 75
- IsActiveMember (고유값 : 0, 1) - 수치(범주)
    - 0 : 1 = 50 : 50
- EstimatedSalary - 수치(연속)
    - 대체로 정규 분포 형태로 인식 가능
    - 0인 값이 생각보다 많이 존재
- 상관관계 분석
    - 변수간 상관계수가 높은 경우- 타겟 데이터와 비교
        - 양의 상관관계
            - Age(0.34)
            - Balance(0.13)
            - Geography_Germany(0.21)
            - Gender_Female(0.15)
        - 음의 상관관계
            - NumOfProducts(-0.21)
            - IsActiveMember(-0.21)
            - Geography_France(-0.13)
            - Gender_Male(-0.15)
    - 변수간 상관계수가 높은 경우 - 기존 변수와 파생 변수 제외 비교
        - 양의 상관관계
            - Age & Geography_Germany(0.09)
            - Balance & Geography_Germany(0.54)
            - age_group & Germany(0.1)
            - balance_group & Germany(0.56)
        - 음의 상관 관계
            - Age & Geography_France(-0.07)
            - Age & NumOfProducts (-0.1)

- Balance & Fance (-0.33)
- Balance & NumOfProducts (-0.36)
- age_group &- NumOfProducts (-0.11)
- balance_group & NumOfProducts(-0.37)
- Geography_France(-0.35)
- Geography_spain(-0.14)

4-2. 모델 선택 및 훈련

▼ AutoML

- 상위 5개의 모델 추출 - LightGBM, CatBoost, GradientBoost, XGBoost, AdaBoost

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.864	0.887	0.548	0.741	0.630	0.549	0.558	7.680
catboost	CatBoost Classifier	0.863	0.886	0.540	0.744	0.626	0.545	0.555	32.350
gbc	Gradient Boosting Classifier	0.863	0.885	0.526	0.750	0.618	0.538	0.550	20.178
xgboost	Extreme Gradient Boosting	0.862	0.883	0.546	0.732	0.625	0.543	0.551	1.312
ada	Ada Boost Classifier	0.858	0.878	0.518	0.735	0.607	0.524	0.536	4.024
rf	Random Forest Classifier	0.854	0.865	0.530	0.707	0.606	0.519	0.527	18.184
et	Extra Trees Classifier	0.848	0.849	0.531	0.681	0.597	0.505	0.511	14.106
ridge	Ridge Classifier	0.839	0.826	0.377	0.730	0.497	0.412	0.444	0.154
lda	Linear Discriminant Analysis	0.838	0.826	0.451	0.674	0.540	0.446	0.459	0.218
knn	K Neighbors Classifier	0.840	0.813	0.499	0.661	0.568	0.472	0.479	27.694
lr	Logistic Regression	0.827	0.806	0.365	0.665	0.471	0.379	0.403	4.100
nb	Naive Bayes	0.803	0.752	0.394	0.547	0.458	0.341	0.348	0.228
dt	Decision Tree Classifier	0.795	0.699	0.529	0.515	0.522	0.391	0.391	1.338
qda	Quadratic Discriminant Analysis	0.633	0.610	0.311	0.096	0.142	0.022	0.025	0.362
dummy	Dummy Classifier	0.788	0.500	0.000	0.000	0.000	0.000	0.000	0.656
svm	SVM - Linear Kernel	0.673	0.362	0.200	0.042	0.070	0.000	0.000	9.260

4-3. 모델 튜닝

▼ GridSearch CV

- LightGBM

```
# LightGBM
param_grid = {
```

```
    'num_leaves': [31, 64, 128],
    'max_depth': [5, 10, 15],
    'learning_rate': [0.05, 0.1, 0.2],
    'objective':['binary'],
    'n_estimators':[100,200,300]
}
```

- CatBoost

```
# CatBoost
param_grid = {
    'learning_rate': [0.01, 0.05, 0.1, 0.5, 1],
    'depth': [4, 6, 8, 10],
    'l2_leaf_reg': [1, 3, 5, 7, 9],
    'border_count': [32, 64, 128, 256],
    'iterations': [100, 200, 300],
    'bagging_temperature': [0, 1, 5, 10]
}
```

- GradientBoost

```
# GradientBoost
param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5]
    }
```

- XGBoost

```
param_grid = {
    'learning_rate': [0.01, 0.05, 0.1, 0.5, 1],
    'n_estimators': [50, 100, 200, 300],
    'max_depth': [3, 4, 5, 6, 7],
    'min_child_weight': [1, 2, 4, 8],
    'subsample': [0.5, 0.7, 0.9, 1.0],
    'colsample_bytree': [0.5, 0.7, 0.9, 1.0],
    'gamma': [0, 0.1, 0.2, 0.3, 0.4],
```

```
    'reg_alpha': [0, 0.1, 0.5, 1, 10],
    'reg_lambda': [0, 0.1, 0.5, 1, 10]
    }
```

- AdaBoost

```
param_grid = {
    'n_estimators': [50, 100, 200, 300],
    'learning_rate': [0.01, 0.05, 0.1, 0.5, 1]
    }
```

- 모델 평가(공통)

```
# GridSearchCV를 사용하여 그리드 서치 수행
grid_search = GridSearchCV(estimator= 각 모델, param_grid=param_gr
grid_search.fit(X_train_scaled, y_train)

# 최적의 하이퍼 파라미터 조합 확인
print("Best parameters:", grid_search.best_params_)

# 최적 모델에 대한 평가
print("Best score:", grid_search.best_score_)
```

▼ Bayesian Optimization

- LightGBM

```
# LightGBM
param_bounds = {
    'n_estimators'      : (50, 500),
    'learning_rate'     : (0.01, 0.3),
    'num_leaves'        : (20, 100),
    'max_depth'         : (3, 10),
    'min_child_samples' : (10, 50),
    'subsample'         : (0.5, 1.0),
    'colsample_bytree'  : (0.5, 1.0),
    'reg_alpha'         : (0.0, 1.0),
    'reg_lambda'        : (0.0, 1.0),
```

```
    'min_child_weight'  : (0.1, 10.0),
}
```

- CatBoost

```
# CatBoost
param_bounds3 = {
    'iterations'         : (50, 500),
    'learning_rate'      : (0.01, 0.3),
    'depth'              : (3, 10),
    'l2_leaf_reg'        : (0.0, 1.0),
    'border_count'       : (32, 255),
    'random_strength'    : (0.0, 1.0),
    'bagging_temperature' : (0.0, 1.0),
    'min_data_in_leaf'   : (1, 50)
}
```

- GradientBoost

```
# GradientBoostingClassifier
param_bounds = {
    'learning_rate'    : (0.01, 0.3),
    'n_estimators'     : (50, 500),
    'max_depth'        : (3, 10),
    'min_samples_split' : (2, 20),
    'min_samples_leaf'  : (1, 10),
    'max_features'     : (0.1, 1.0),
    'subsample'        : (0.5, 1.0),
}
```

- XGBoost

```
# XGBoost
param_bounds = {
    'n_estimators'     : (100,500),
    'learning_rate'    : (0.01, 0.3),
    'max_depth'        : (3, 10),
```

```
    'subsample'        : (0.5, 1.0),
    'min_samples_split' : (2,5),
    'min_samples_leaf'  : (2,5),
    'max_leaf_nodes'    : (2,100),
    'colsample_bytree'  : (0.5, 1.0),
}
```

- AdaBoost

```
# CatBoost
param_bounds = {
    'n_estimators'  :(50, 200),
    'learning_rate' :(0.01, 1.0),
}
```

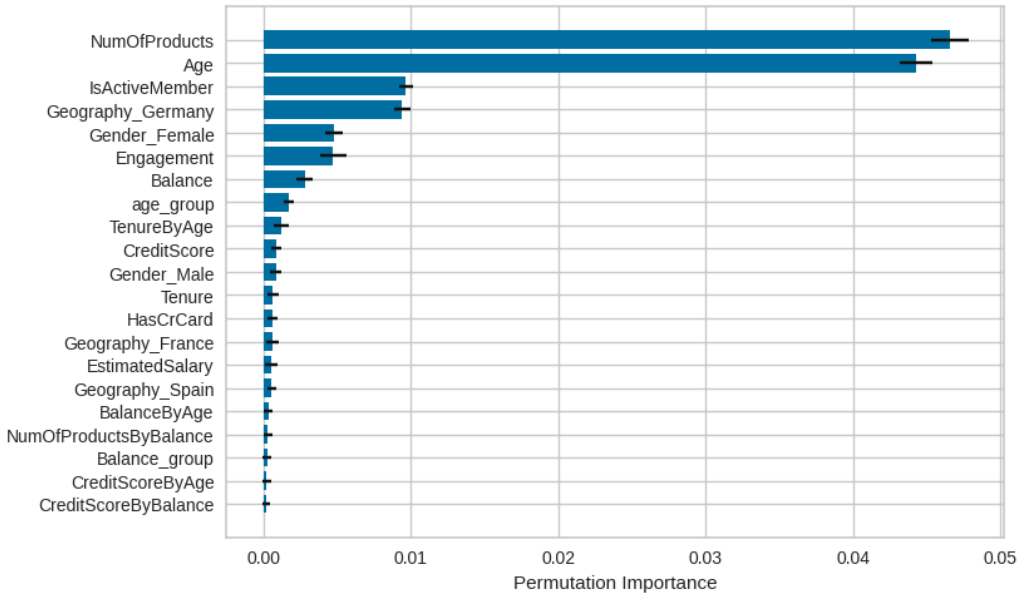
▼ Ensemble Learning - Voting

```
# 알고리즘 생성
clf_cat = model_CAT
clf_lgb = model_LGBM
clf_xgb = model_XGB
clf_ada = model_ADA
clf_gbc = model_GBC

# 보팅 구성 - 'soft'
clf_vc = VotingClassifier([('CAT', clf_cat), ('XGB', clf_xgb), ('LGB', clf_lgb), (

# 학습 및 평가
clf_vc.fit(X_train, y_train)
y_valid_preds = clf_vc.predict_proba(X_valid)[:, 1]
roc_auc = roc_auc_score(y_valid, y_valid_preds)
print(f'검증 데이터 ROC AUC : {roc_auc:.4f}')
```

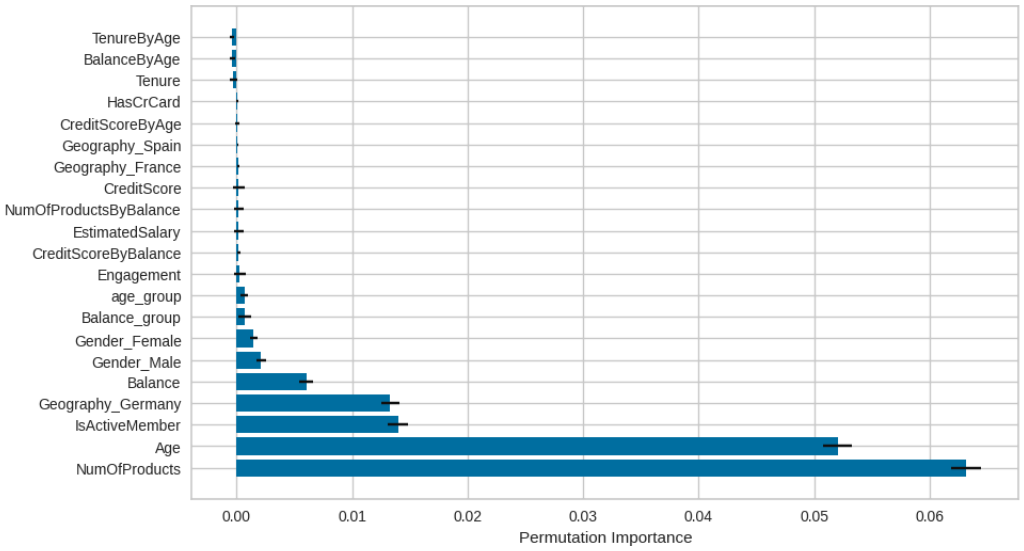
4-4. 모델 평가

▼ 변수 성능 평가

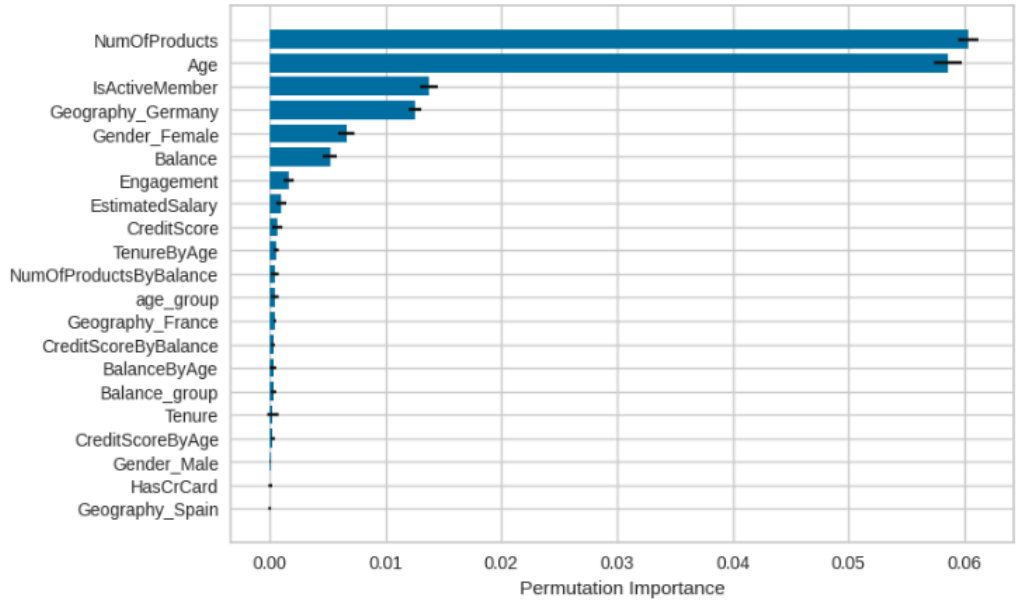
- Permutation_Importance
  - LightGBM



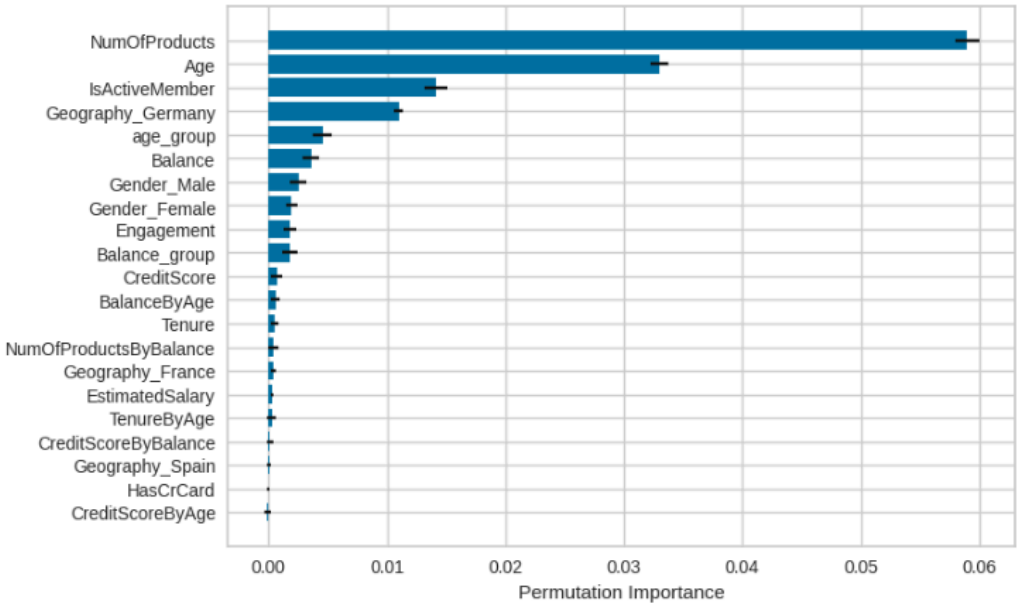
- CatBoost



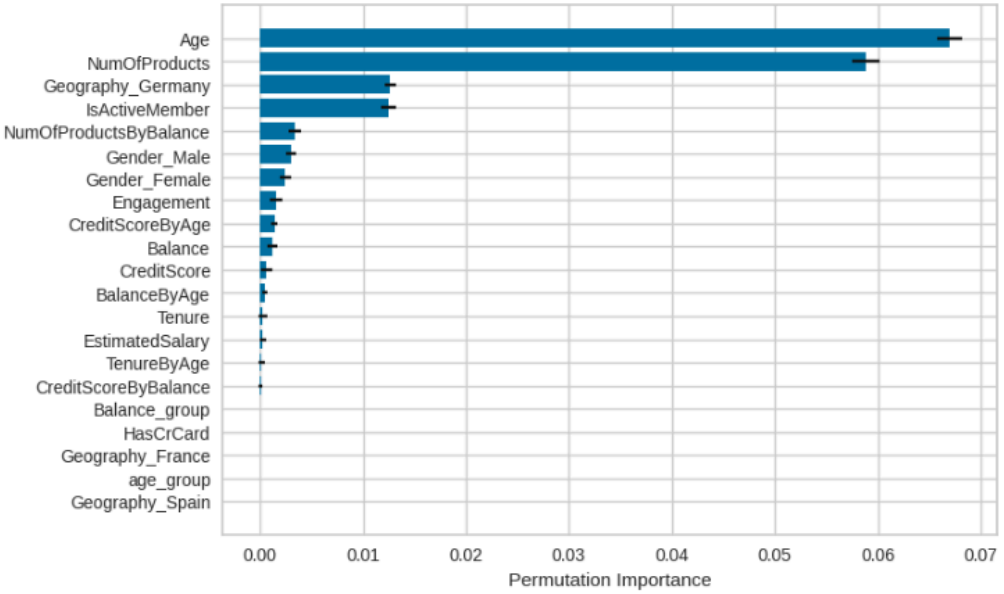
- XGBoost



- GradientBoost



- AdaBoost



▼ 모델 성능 평가

▼ 방법 1

- 파생변수 생성
  - age_group
  - balance_group
  - BalanceByAge
  - CreditScoreByAge
  - CreditScoreByBalance
  - Engagement
  - NumOfProductsByBalance
  - TenureByAge
- 모델 평가
  - LightGBM :
    - 코랩 점수 : 0.8918
  - AdaBoost
    - 코랩 점수 : 0.8834
  - CatBoost

        - 코랩 점수 : 0.8914
    - XGBoost
        - 코랩 점수 : 0.8917
    - GBC
        - 코랩 점수 : 0.8909
    - Top3 Voting(CatBoost, LightGBM, GBC)
        - 코랩 점수 : 0.8925
        - 캐글 점수 : 0.88793

▼ 방법 2

- 파생 변수 생성
    - age_group
    - balance_group
    - BalanceByAge
    - CreditScoreByBalance
    - Engagement
    - NumOfProductsByBalance
    - TenureByAge
- 모델 평가
    - LightGBM
        - 코랩 점수 : 0.8918
    - AdaBoost
        - 코랩 점수 : 0.8832
    - CatBoost
        - 코랩 점수 : 0.8909
    - XGBoost
        - 코랩 점수 : 0.8919

    - GBC
        - 코랩 점수 : 0.8905
    - Top3 Voting(LightGBM, XGBoost, GBC)
        - 코랩 점수 : 0.8920
        - 캐글 점수 : 0.88799

▼ 방법 3

- 파생 변수 생성
    - age_group
    - balance_group
    - Age_Geography
    - BalanceByAge
    - Balance_Group_Geography
    - Balacne_Group_NumOfProducts
    - CreditScoreByBalance
    - Engagement
    - NumOfProductsByBalance
    - TenureByAge
- 모델 평가
    - LightGBM
        - 코랩 점수 : 0.8919
    - AdaBoost
        - 코랩 점수 : 0.8883
    - CatBoost
        - 코랩 점수 : 0.8914
    - XGBoost
        - 코랩 점수 : 0.8916

    - GBC
        - 코랩 점수 : 0.8909
    - Top3 Voting(LightGBM, XGBoost, CatBoost)
        - 코랩 점수 : 0.8920
        - 캐글 점수 : 0.88823

▼ 방법 4

- 파생 변수 생성
    - age_group
    - Balance_group
    - Age_Geography
    - Balance_group_Geography
    - Balance_group_NumOfProducts
    - CreditScoreByBalance
    - Engagement
    - NumOfProductsByBalance
    - TenureByAge
- 모델 평가
    - LightGBM
        - 코랩 점수 : 0.8918
    - AdaBoost
        - 코랩 점수 : 0.8884
    - CatBoost
        - 코랩 점수 : 0.8914
    - XGBoost
        - 코랩 점수 : 0.8914
    - GBC
        - 코랩 점수 : 0.8911

        - Top Voting(CatBoost, LightGBM, XGBoost, GBC, AdaBoost)
            - 코랩 점수 : 0.8920
            - 캐글 점수 : 0.88776

▼ 방법 5

- 파생 변수 생성
    - age_group
    - Balance_group
    - Age_Geography
    - BalanceByAge
    - Balance_group_Geography
    - Balance_group_NumOfProducts
    - CreditScoreByAge
    - CreditScoreByBalance
    - Engagement
    - NumOfProductsByBalance
    - TenureByAge
- 모델 평가
    - LightGBM
        - 코랩 점수 : 0.8918
    - AdaBoost
        - 코랩 점수 : 0.8882
    - CatBoost
        - 코랩 점수 : 0.8914
    - XGBoost
        - 코랩 점수 : 0.8916
    - GBC
        - 코랩 점수 : 0.8916

        - Voting(XGoostB, GBC, LightGBM)
            - 코랩 점수 : 0.8921
            - 캐글 점수 : 0.88788

5. 문제 분석

5-1. 원인 분석

- 중복 값 제거 정의
    - id를 제거한 데이터에서 중복 값이 발견되어 가장 private 한 변수라고 생각되는 CustomerId와 Surname을 기준으로 중복 값을 확인함.
    - CustomerId는 고유값이 23,221개로 전체 데이터인 165,034개와 비교하여 보니 현저히 적은 것을 확인하였고 따라서 데이터 원본을 확인 후 CustomerId는 잘못된 변수라고 판단하였음.
    - id, CustomerId를 제거한 데이터는 Surname부터 나머지 값들이 모두 동일한 값은 적절치 못한 데이터 혹은 분석에 방해가 되는 요소라고 생각하여 중복 값을 제거하였음.
- 파생 변수 생성 정의
    - 은행 이탈에 대한 영향 가능성을 예측, 분류하며 의미가 있는 것으로 판단된 변수로 묶어 파생 변수를 생성하였음.

5-2. 분석 결과

- 기본 전처리 결과에서 Permutation Importance를 알아보고 중요하다고 나온 변수들끼리 한 번 더 조합을 해서 만들어낸 파생 변수(Balance_group_Geography, Balance_group_NumOfProducts, Age_Geography)를 추가했을 때 가장 높은 성능을 보임.
- 3번째 모델의 Permutation Importance 확인 결과, 고객이 가지고 있는 은행의 상품 개수와 연령이 고객 이탈 예측 모델에 가장 중요한 요소로 나타남. 이어 활성 회원 여부가 중요한 요소로 나타남.
- 중요하지 않다고 나온 변수들을 삭제하면 모델의 성능이 더 떨어짐.
    - 중요하지 않은 것들은 중요하지 않은 대로 의미가 있다고 판단함.
- 그러므로 중요하다고 나타난 변수들에 포커스를 맞추어 경쟁력을 강화하는 방법을 고려해야 함.
    - 연령 별에 맞는 은행 상품 수립 및 출시하는 전략을 통해 고객에게 주인 의식을 갖게 해야 함.

6. 스토리텔링

6-1. 시사점

- 은행 이탈 고객을 예측하는 연령과 보유한 상품의 개수가 중요함.
- 상품의 개수, 신용카드의 개수 등 고객으로 하여금 주인의식을 높이는 전략이 필요함.

6-2. 한계점

- 중복 값 처리
  - Id와 CustomerId을 제거하고 중복 값을 제거하고 성능이 올랐지만 사실 중복 값 제거의 근거로써 타당하지 않다고 여김.
  - CustomerId, Surname이 동일한 데이터 중 EstimatedSalary가 동일한 값이 매우 많았으며 그 값 중 Balance가 동일 혹은 0 인 경우가 다수 파악되었음.
    - 같은 사람이지만 여러 계좌를 보유한다는 가능성을 보여줌.
    - 같은 사람의 데이터로 보여지며 Exited의 값이 1이 다수 존재하므로 데이터의 오류가 있는 것으로 확인함.
  - 변수 중 변수를 조사한 날짜가 있었다면 중복 값 제거에 좀 더 도움이 되지 않았을까 함.
- 원본(Origin) 데이터 참고
  - Origin 데이터를 추가해서 더 많은 데이터로 모델 생성했지만 성능이 더 떨어짐.
- 코랩 점수와 캐글 점수의 차이
  - 코랩 상으로 목표한 캐글 점수에 도달할 수 있도록 SVD 처리 역량 강화가 필요하다고 판단함.

7. 프로젝트 회고 및 개선점

7-1. 피드백

- EDA 시각화 분석
  - 발표 자료 흐름 상 '피처 데이터 요약' 페이지가 존재해야 매끄러운 자료로 작성될 것임.

7-2. 회고

- 기술적 한계를 극복하고 받아들여 주어진 환경에서 최선의 방법을 선택하도록 함.

7-3. 개선점

- 데이터 자체에 대한 더욱 정밀한 분석 필요함.
- 배경지식을 이용한 데이터 이해가 필요함.
    - 해외 연금 시스템의 60대 이상 데이터가 현저히 감소하였다. → 활용한 전처리 계획 수립 등

7-4. 추후 개선 계획

- SVD 등 기술적 영역에 대해 학습하고 더 나은 모델을 만들 수 있도록 분석에 집중하며 도메인을 이해하고 접근하여 새로운 인사이트를 발견하도록 노력하겠음.

8. 부록

8-1. 참고자료

- 은행 이탈 데이터의 원본 데이터
- 은행 이탈 원본 데이터의 설명

8-2. 출처

https://www.kaggle.com/code/kmalit/bank-customer-churn-prediction

https://www.kaggle.com/datasets/adammaus/predicting-churn-for-bank-customers