

멋쟁이사자처럼 데이터분석 5기 파이널 프로젝트

# 신용카드 고객 세그먼트 분류

Team\_15

# CONTENTS

SECTION 01

데이터 소개

SECTION 02

분석 및 모델링 절차

SECTION 03

EDA

SECTION 04

모델링

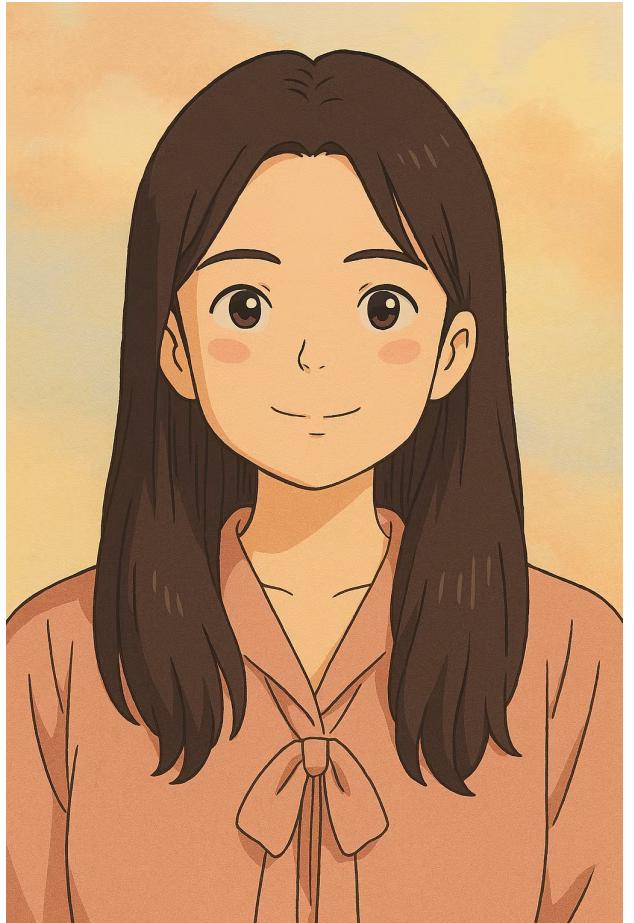
SECTION 05

예측 및 성능 평가

SECTION 06

요약 및 결론

# PROJECT TEAM MEMBERS



Sehee Moon



Minjae Kim



Eunsung Kim



Byeori Jang

Section 01

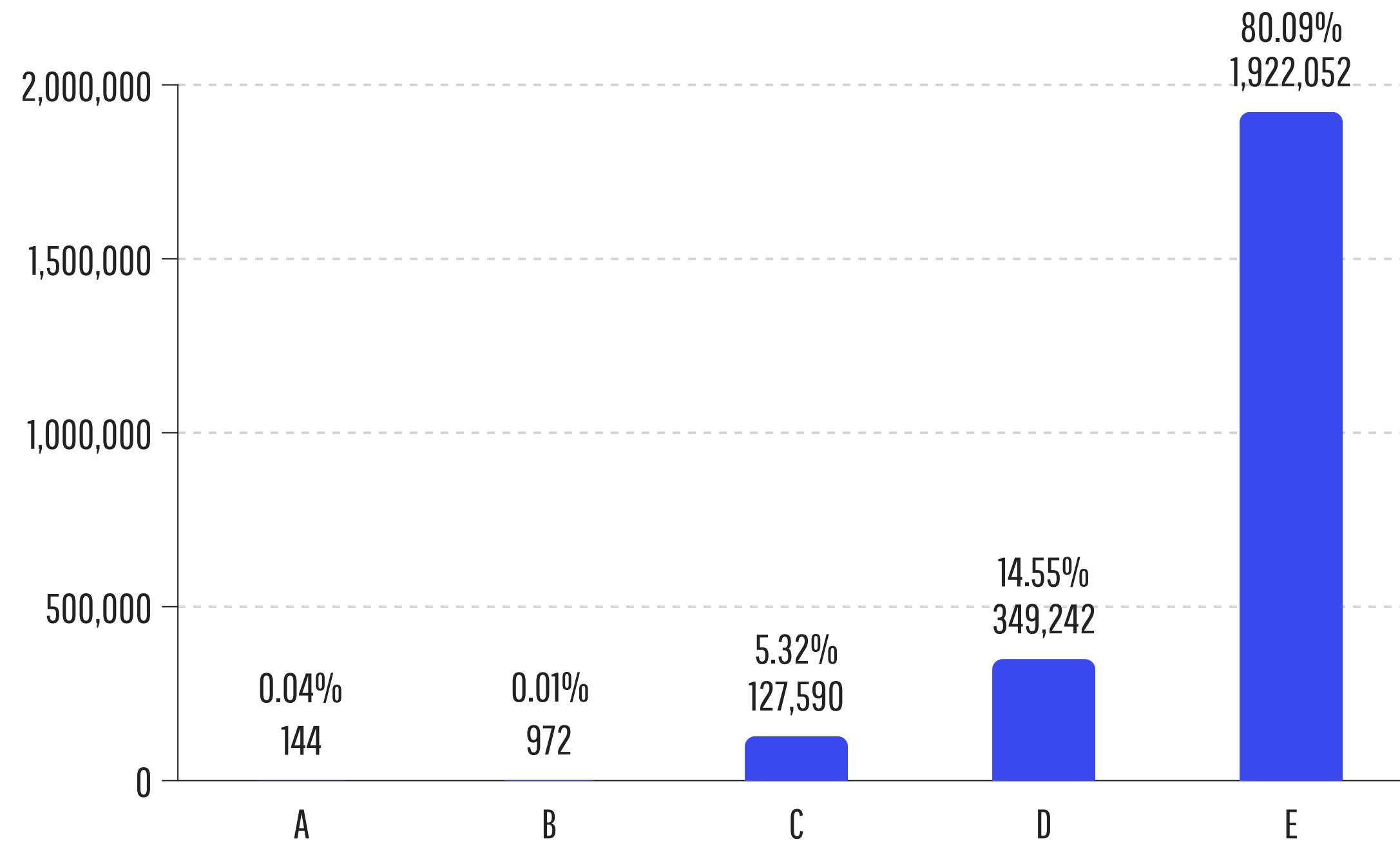
# 데이터 소개

# 1. 데이터 소개

구분	Train	Test
1. 회원정보	77 + Segment	77
2. 신용정보	42	42
3. 승인매출정보	406	406
4. 청구입금정보	46	46
5. 잔액정보	82	82
6. 채널정보	105	105
7. 마케팅정보	64	64
8. 성과정정보	49	49

- 2018.07 ~ 2018.12 고객별 금융활동 데이터
- 고객의 특성을 기반으로 A~E Segment 분류
- train data - 2,400,000 rows / 872 columns
- test data - 600,000 rows / 871 columns
- 식별자 : ID , 기준년월

# SEGMENT 분포



*Section 02*

# 부석 및 모델링 절차

# 분석 및 모델링 절차

## 1. 데이터 전처리

- 결측값 처리, 날짜 형식 변환
- 전체 병합

## 2. 피처 선택

- XGBoost, CatBoost, 상관계수 3가지 기준 사용

## 3. 클래스 분리 전략

- E/Not E → Not E 내에서 C/D/Not CD → Not CD 내에서 A/B

## 4. 모델링

- CatBoostClassifier (GPU), StratifiedKFold, 기준 지표 f1-micro

## 5. 최종예측

*Section 03*

# EDA

### 3. EDA

#### 전처리 항목

- 파일 구성별로 날짜 병합 (2018.07 ~ 2018.12)
- 피처별 분석
  - 결측값 개수, 유형 수, 최대/최솟값, 최빈값 탐색
- 결측값 대체
  - 중앙값 / 최빈값
  - 미이용 → 0으로 대체
  - -1로 대체
- 날짜 형식 변환
  - 방식 1: 년 / 월 / 이용여부로 분할
  - 방식 2: 이용여부로 이진분류 파생 변수 생성

# FEATURE SELECTION

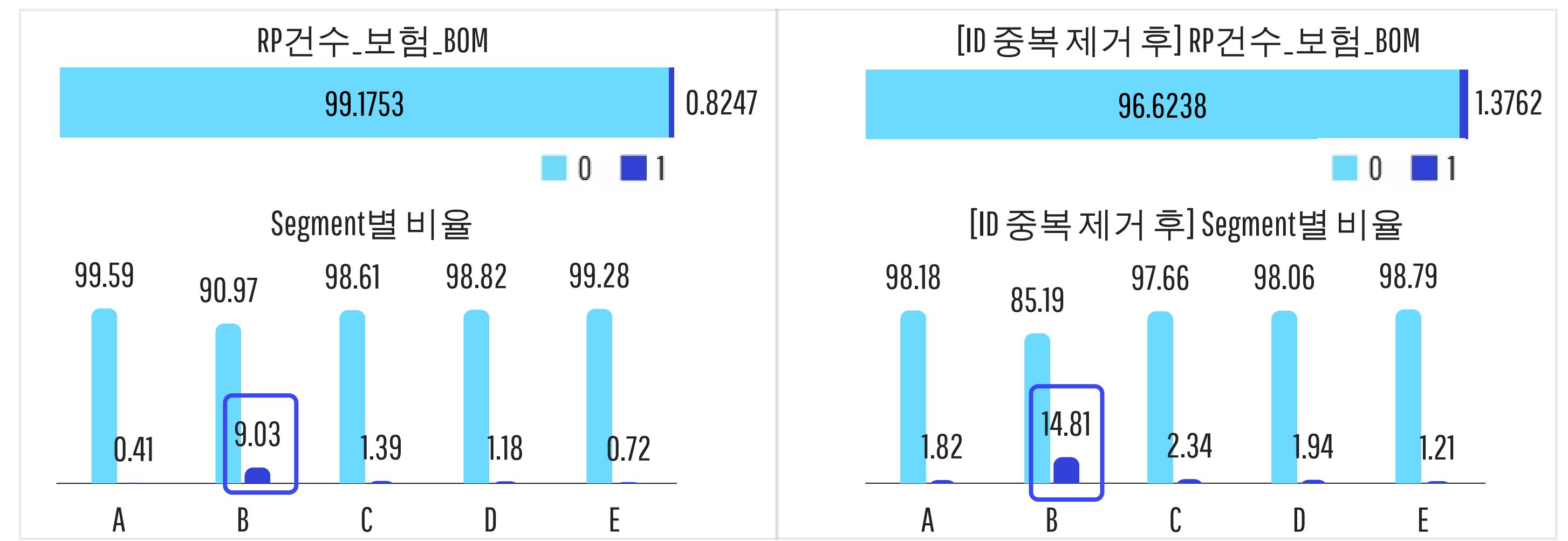
## RULE - BASED FEATURE ELIMINATION

- 값이 하나뿐인 feature 제거
- 소수 클래스의 값이 총 10개 미만인 feature 제거  
(ex. 2,400,000개 중 2,399,997 vs 3)
- 한쪽 클래스가 전체의 98% 이상을 차지하는 feature 제거
  - 단, 특정 조건을 모두 만족하는 경우 제거하지 않음
    - 특정 Segment의 소수 클래스 비율이 7% 이상
    - ID 중복 제거 후, Segment의 소수 클래스 비율이 전체 소수 클래스 비율보다 5배 이상 높은 경우

# FEATURE SELECTION

## RULE - BASED FEATURE ELIMINATION

- 특정 Segment의 소수 클래스 비율이 7% 이상
- ID 중복 제거 후, 특정 Segment의 소수 클래스 비율이 전체 소수 클래스 비율보다 5배 이상 높은 경우



# FEATURE SELECTION

- catboost 전체 → feature importance 상위 20개
- XGBoost → feature importance 상위 20개
- 상관계수 0.4 이상



- vif 계수를 확인하여 13 이하인 feature 선택
- vif 기준으로 13 안팎의 feature들로 추가/제거를 반복해 최종 feature 선정

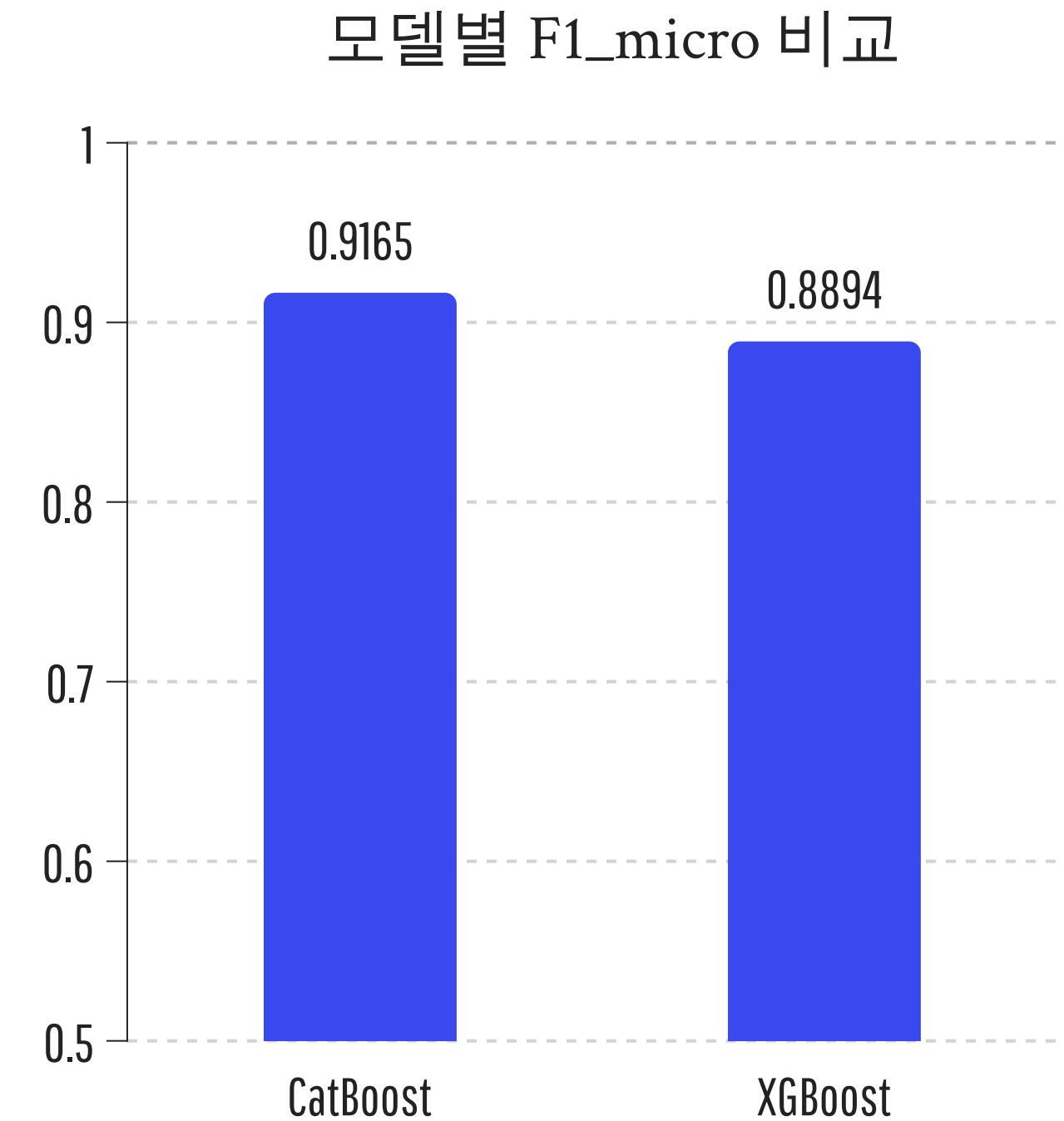
*Section 04*

모델링

# 4. 모델링

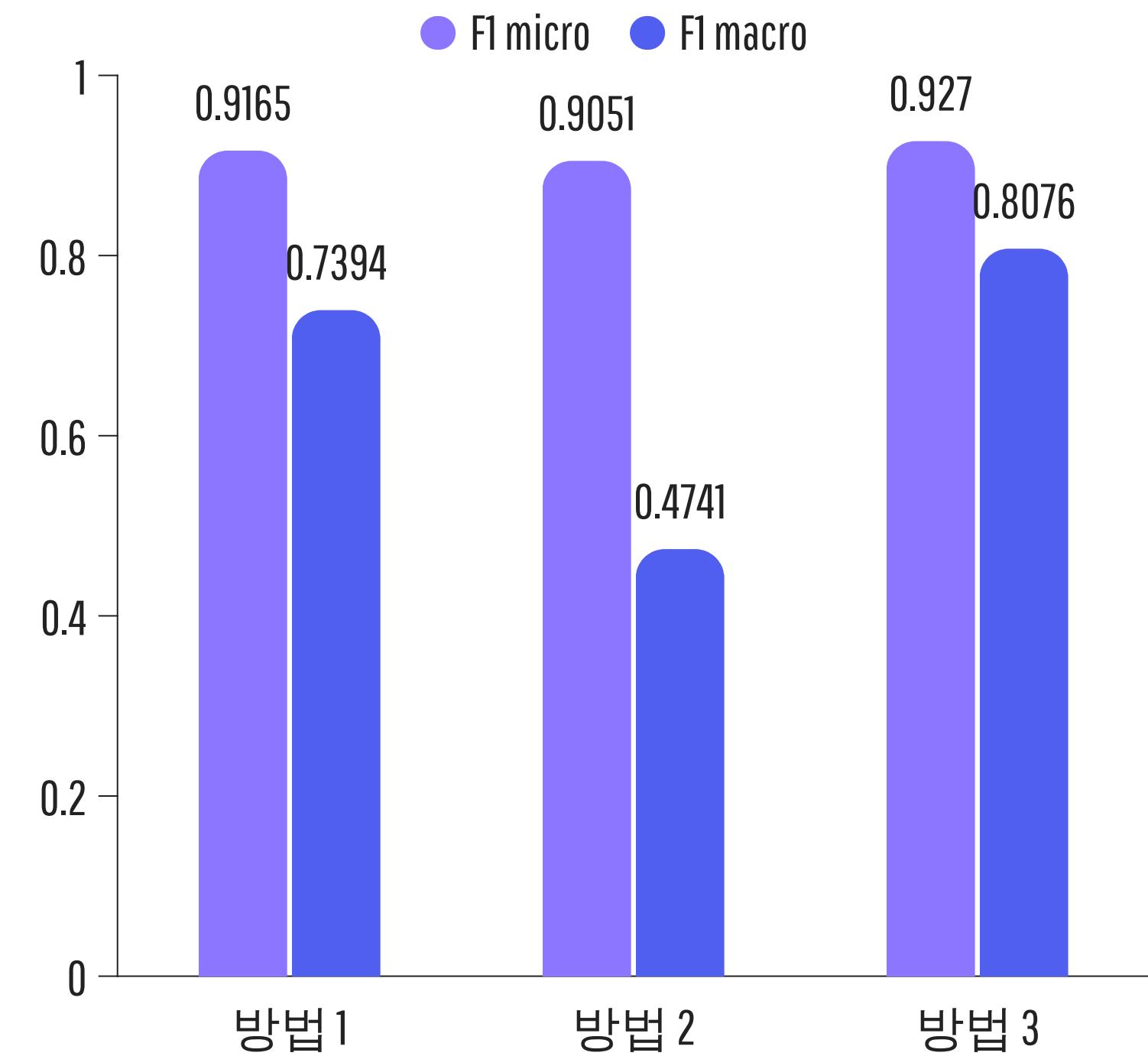
## CatBoost 선택 이유

- GPU 사용 가능 (Colab A100)  
→ 대규모 데이터 처리에 효율적
- 비교 모델 대비 좋은 성능  
→ XGBoost보다 높은 F1\_micro
- 범주형 자동 처리  
→ 전처리 간소화 (One-hot encoding 불필요)
- 하이퍼파라미터 튜닝 없이도 안정적  
→ 기본 parameter 잘 최적화됨

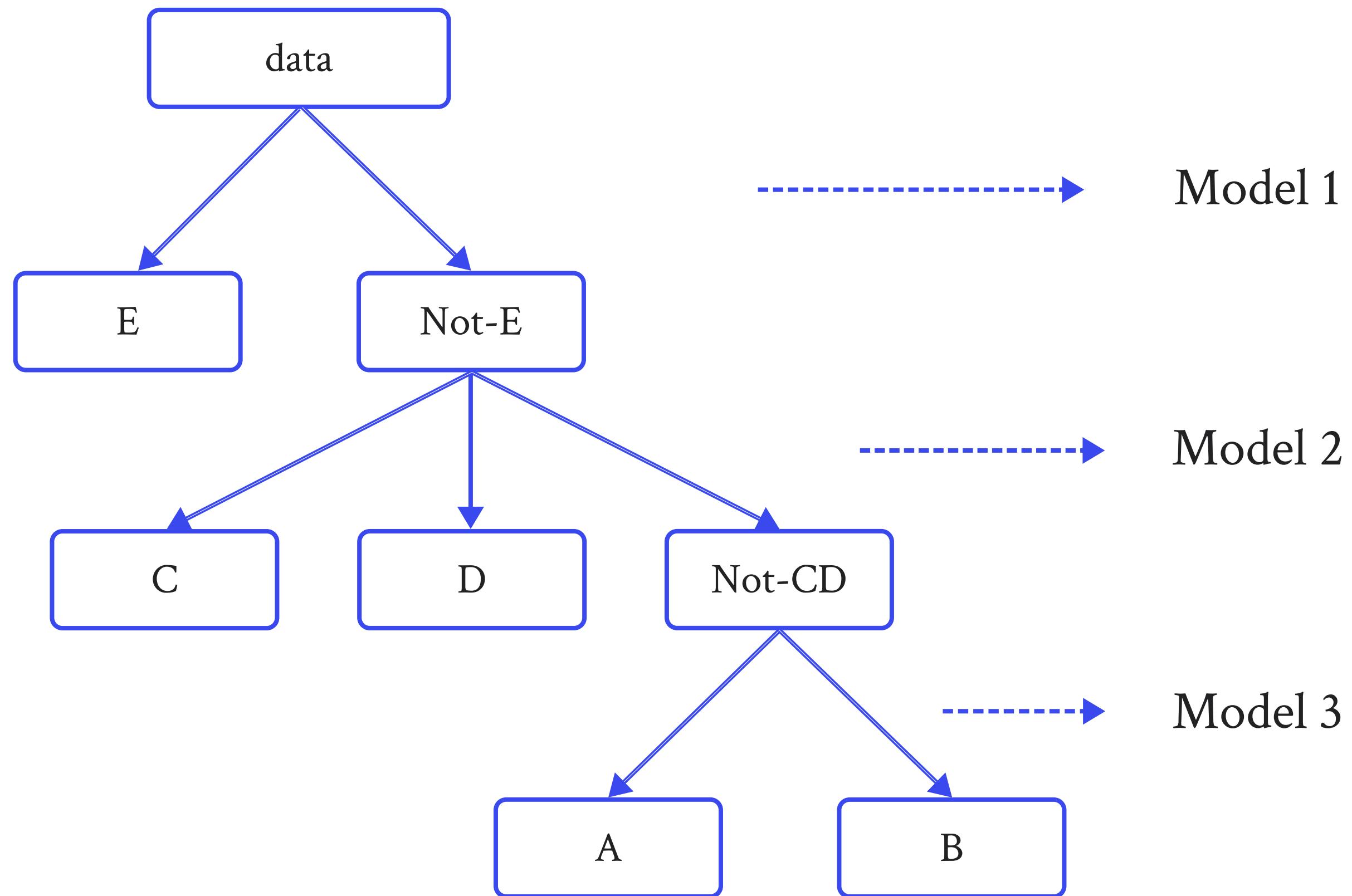


# 모델링 전략별 성능 비교

방법	구조	F1_micro	F1_macro
1	단일 모델 A-E	0.9165	0.7394
2	AB / CDE	0.9051	0.4741
3	E/Not-E → C/D/not-CD → A/ B	0.9270	0.8076



# 단계적 분류 전략



# 최종 모델 결과 요약

## 단계적 분류 (3모델 구조)

- 모델 1: E / Not-E
- 모델 2: C / D / Not-CD (from Not-E)
- 모델 3: A / B (from Not-CD)

## 주요 성능 (Validation 기준)

- F1\_micro: 0.9270
- F1\_macro: 0.8076

## 주요 Parameter

model	iterations	learning_rate	depth	early_stop	loss_function
모델 1	10,000	0.03	8	1,000	Logloss
모델 2	10,000	0.03	8	1000	MultiClass
모델 3	10,000	0.03	8	1000	Logloss

→ 단계적 설계를 통해 불균형 문제를 고려한 분류 성능을 확보

*Section 05*

# 예측 및 성능평가

# 5. 예측 및 성능 평가

방법	파이프라인	Feature 개수	F1_Micro (평균 / 3 Fold)	Kaggle
1	3	53개	0.9305	0.9563
2	1	762개	0.9092	0.9345
3	1	40개	0.9133	0.9489
4	3	43개	0.9474	0.9576

## □ 최고 점수 모델 미선정 이유

- 최고 점수 모델의 경우 한쪽 클래스의 비율이 98% 이상인 불균형 feature 삭제
  - \* 삭제한 불균형 feature의 Segment 비율과 feature의 값별 Segment 비율의 유의미한 관계 확인 제한
- 98% 기준으로 삭제한 feature 내 Segment 분류에 유의미한 feature 가능성
- 현재 데이터 기준에서만 높은 예측력을 보이는 편향적 모델의 위험성 존재

# 5. 예측 및 성능 평가

## 주 평가 지표 : F1\_micro\_Score

- 선정 이유

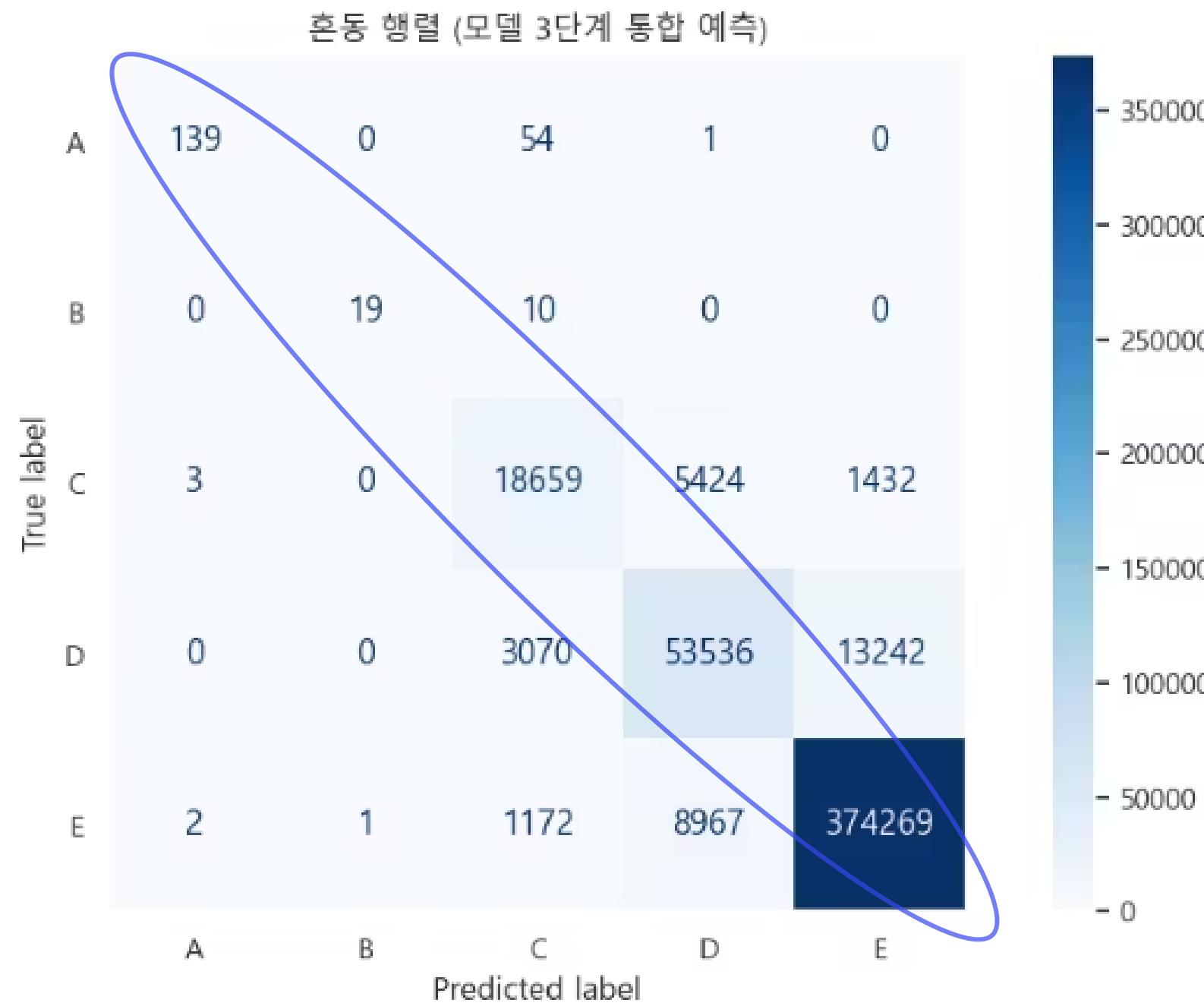
- 'E' 클래스의 고객이 실제 Segment의 약 80% 구성
  - 불균형 데이터 환경에서 **전체적인 평균 예측 오류 최소화**
  - 실제 전체 고객 예측 간 가장 현실적으로 측정 가능 판단
- \* 전체 고객 예측에서 잘못 분류된 비율 낮추는 것이 중요

\* F1\_macro는 추가 참고지표로만 활용

모델	분류기준	F1_micro	목적	해석
1	E or Not E	0.947	E 클래스 대량 편중 문제 해소	E 클래스 매우 높은 구분력
2	C or D or Not C/D	0.891	C, D 클래스 구분 + 비C,D 식별	다중 클래스에서 충분한 안정적인 분류 성능
3	A or B	0.991	소수 클래스 A/B 정밀 구분	극소 클래스 간 매우 높은 구분력

# 5. 예측 및 성능 평가

## 주 평가 지표: 혼동 행렬(Confusion Matrix)



[Cat Boost 분류모델]

- **선정이유**
  - 클래스별 분류 오류 파악
  - 다중클래스 문제의 직관적인 평가 가능
  - 정밀도, 재현율, F1-score 계산의 기초 자료
- A 예측비율 : 71.65%(139/194)
- B 예측비율 : 65.52%(19/29)
- C 예측비율 : 73.12%(18659/25518)
- D 예측비율 : 76.65(53536/69848)
- E 예측비율 : 98.03%(374269/384411)

*Section 04*

요약 및 결론

# 6. 요약 및 결론

## 최종 성과

- 데이터 전처리와 feature 엔지니어링을 통해 모델 학습에 적합한 feature 선별 및 최적화
- 단계적 분류 접근으로 CatBoost 기반 3단계 모델 설계 및 불균형 문제 효과적 대응
- 불균형 구조에서 소수 클래스(A/B)에 대한 의미있는 분류 성능 확보
- Kaggle : 0.9563

## 회고 및 개선할 점

- 잘한 점
  - 전처리와 feature 선택에 충분한 시간 투자를 통해 초기에 안정적인 feature 구성  
→ 후반부 문제 발생 시 빠르게 계획 수정 및 재학습 실행
- 한계 및 개선점
  - 초기에 근거 부족한 컬럼 제거 기준(98%)을 적용하여 반복 수정이 발생 → **설계 단계의 중요성 인식**
  - 결측치 처리에서 경험 부족으로 비효율적인 처리
  - K-Fold / Split 개념 부족으로 검증 데이터 누수 가능성 간과 → **혼동행렬 재분석을 통해 검증 설계 수정**
  - C/D 구간에서 예측 성능 저하 → **시간적 제약으로 다양한 모델 및 파라미터 실험 부족**

*Team\_15*

**THE END**



# Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

[Create a presentation \(It's free\)](#)