

# SOCAR

# Insurance Fraud Detection

3조 김경한 장한아 서기현

2020.12.28 - 2021.01.18

# 목차

## 프로젝트 정의

- 문제 정의
- 알고리즘 선택
- Sampling 선택
- 성능 목표 설정
- Milestone

## 프로젝트 진행

- Data Set
- EDA
- PreProcessing
- Model Test
- Conclusion

# 프로젝트 결과

Model ①						Model ②					
	accuracy	precision	recall	f1	roc_acu		accuracy	precision	recall	f1	roc_acu
<b>LogisticReg</b>	0.99750	0.00000	0.00000	0.00000	0.50000	<b>LogisticReg</b>	0.75598	0.00826	0.71429	0.01634	0.73519
<b>DecisionTree</b>	0.77389	0.00939	0.85714	0.01858	0.81541	<b>DecisionTree</b>	0.60640	0.00513	0.71429	0.01019	0.66019
<b>RandomForest</b>	0.99750	0.00000	0.00000	0.00000	0.50000	<b>RandomForest</b>	0.85367	0.01105	0.57143	0.02168	0.71295
<b>SVM</b>	0.99679	0.00000	0.00000	0.00000	0.49964	<b>LightGBM</b>	0.88569	0.00717	0.28571	0.01399	0.58656
						<b>SVM</b>	0.78516	0.00938	0.71429	0.01852	0.74983
						<b>LinearSVM</b>	0.64613	0.00571	0.71429	0.01133	0.68011

# 프로젝트 정의

### 문제 정의

알고리즘 선택

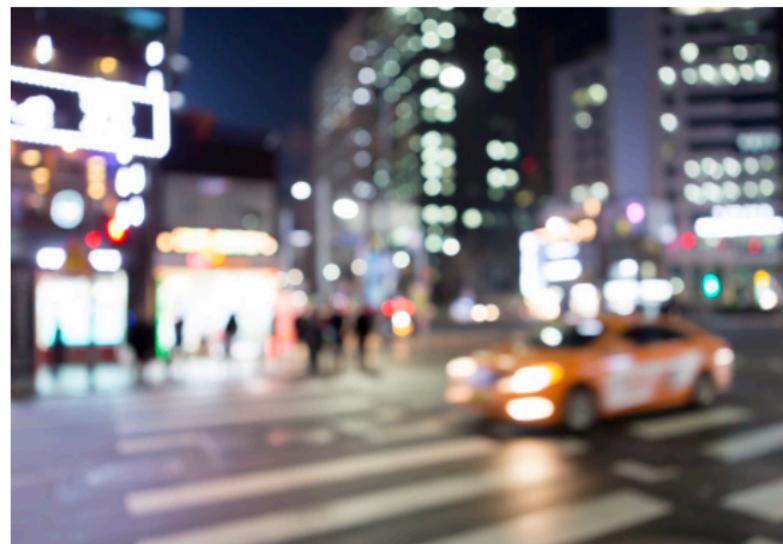
Sampling 선택

성능 목표 설정

Milestone

# 렌터카 보험 사기란?

보험료 또는 합의금을 얻을 목적으로 렌터카를 이용하여 고의 사고를 내는 행위



### 횡단보도 고의 자해

- 횡단보도 측면에 숨어 대기하던 보행자가 신호를 위반해 좌회전하는 차량에 고의로 뛰어드는 사고
- 보행자가 갑자기 차도에 뛰어들어 고의 충돌하는 경우



### 법규 위반 차량 대상 고의사고

무단 주차된 차량으로 불가피하게 중앙선을 침범한 차량을 마주 오던 차량이  
감속 없이 고의로 충돌하는 경우



### 고의 추돌 유도

- 고속도로 램프 진입 중  
고의 급제동하여 후행 차량의 추돌 유발
- 도로주행 중 갑작스럽게 차선을 변경한  
서행하여 후행 차량의 추돌 유발



### 손목 치기

- 도로를 지나가던 보행자가 주행 중인 차량에 고의로 몸을 부딪치는 경우
- 좁은 도로를 지나던 보행자가  
주행 중인 차량에 고의로 신체를 접촉하는 경우

### 문제 정의

알고리즘 선택

Sampling 선택

성능 목표 설정

Milestone

# 렌터카 보험 사기란?

보험료 또는 합의금을 얻을 목적으로 렌터카를 이용하여 고의 사고를 내는 행위



### 횡단보도 고의 자해

- 횡단보도 측면에 숨어 대기하던 보행자가 신호를 위반해 좌회전하는 차량에 고의로 뛰어드는 사고
- 보행자가 갑자기 차도에 뛰어들어 고의 충돌하는 경우



### 법규 위반 차량 대상 고의사고

- 무단 주차된 차량으로 불가피하게 중앙선을 침범한 차량을 마주 오던 차량이  
감속 없이 고의로 충돌하는 경우



### 고의 추돌 유도

- 고속도로 램프 진입 중  
고의 급제동하여 후행 차량의 추돌 유발
- 도로주행 중 갑작스럽게 차선을 변경한  
서행하여 후행 차량의 추돌 유발



### 손목 치기

- 도로를 지나가던 보행자가 주행 중인 차량에 고의로 몸을 부딪치는 경우
- 좁은 도로를 지나던 보행자가  
주행 중인 차량에 고의로 신체를 접촉하는 경우

# 프로젝트 정의

## 렌터카 보험 사기 현황

# 문제 정의

# 알고리즘 선택

## Sampling 선택

## ● 성능 목표 설정

# Milestone

# 프로젝트 정의

이 문제를 해결해야 하는 이유

## 문제 정의

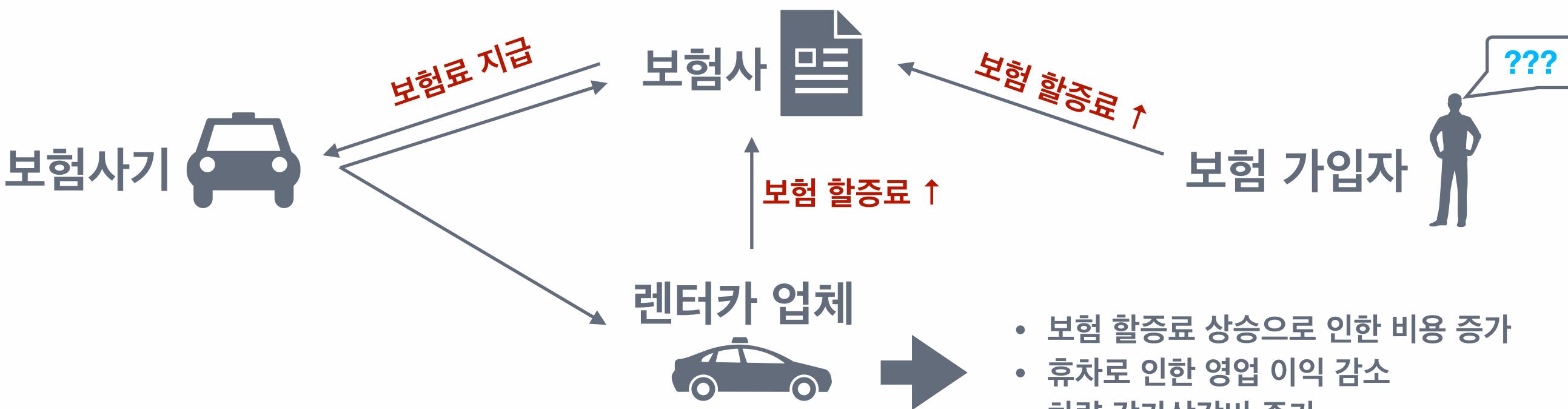
알고리즘 선택

77명 동승자 모집, 렌터카로 일부러 "쿵" …지능화된 보험사기

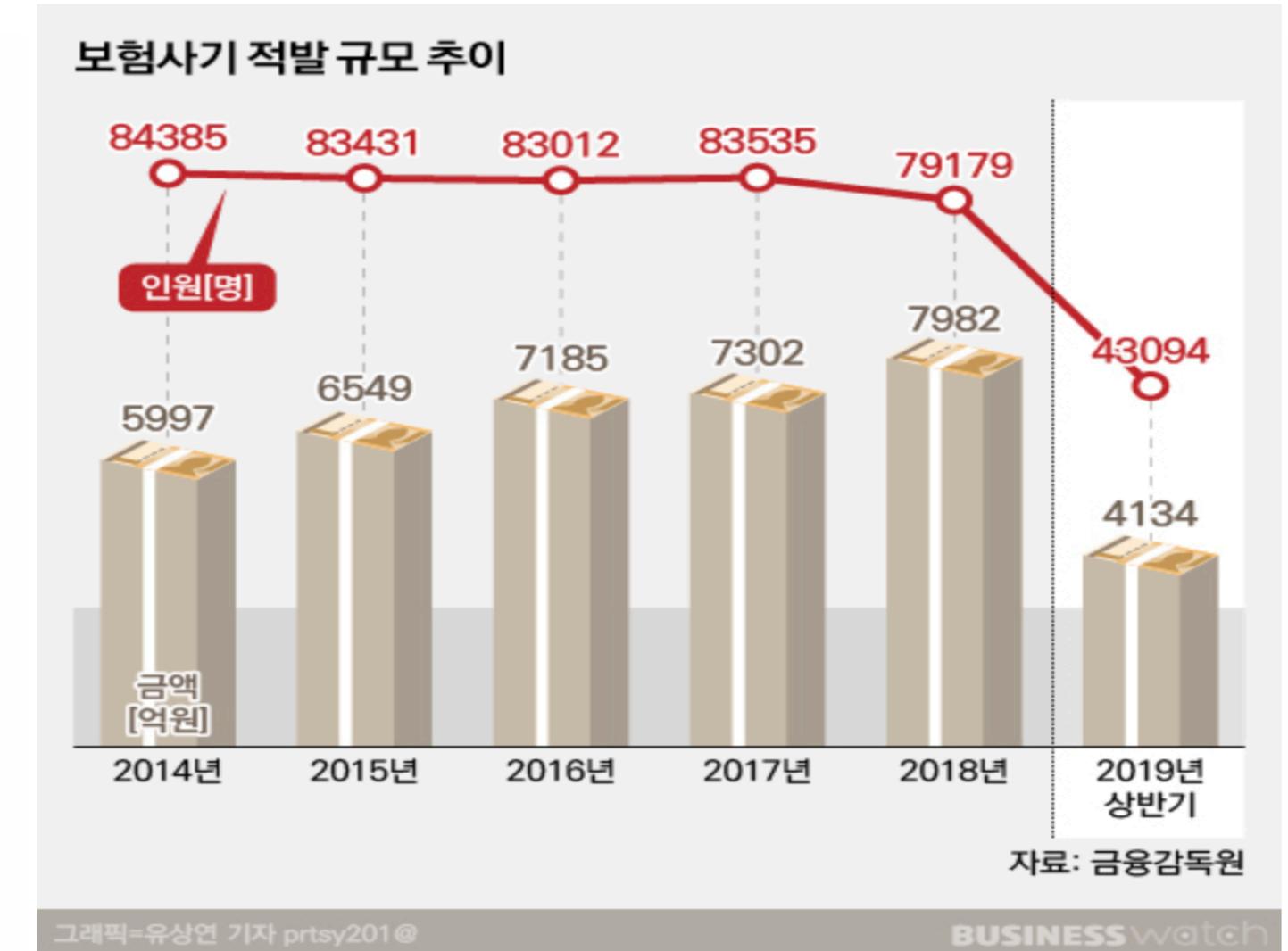
Sampling 선택

보험사기 지능화·대형화에… 지난해 역대 최고 '8000억 적발'

성능 목표 설정



Milestone



# 프로젝트 정의

## 비즈니스 목표

### 문제 정의

알고리즘 선택

Sampling 선택

성능 목표 설정

Milestone

데이터 기반 보험사기인지 아닌지 분류할 수 있는 머신러닝 모델

보험사기 객관적 의심 및 발견

예방을 위한 대응전략 구축

불필요한 손실 비용 감소

# 프로젝트 정의

## SOCAR Data Set

### 문제 정의

알고리즘 선택

Sampling 선택

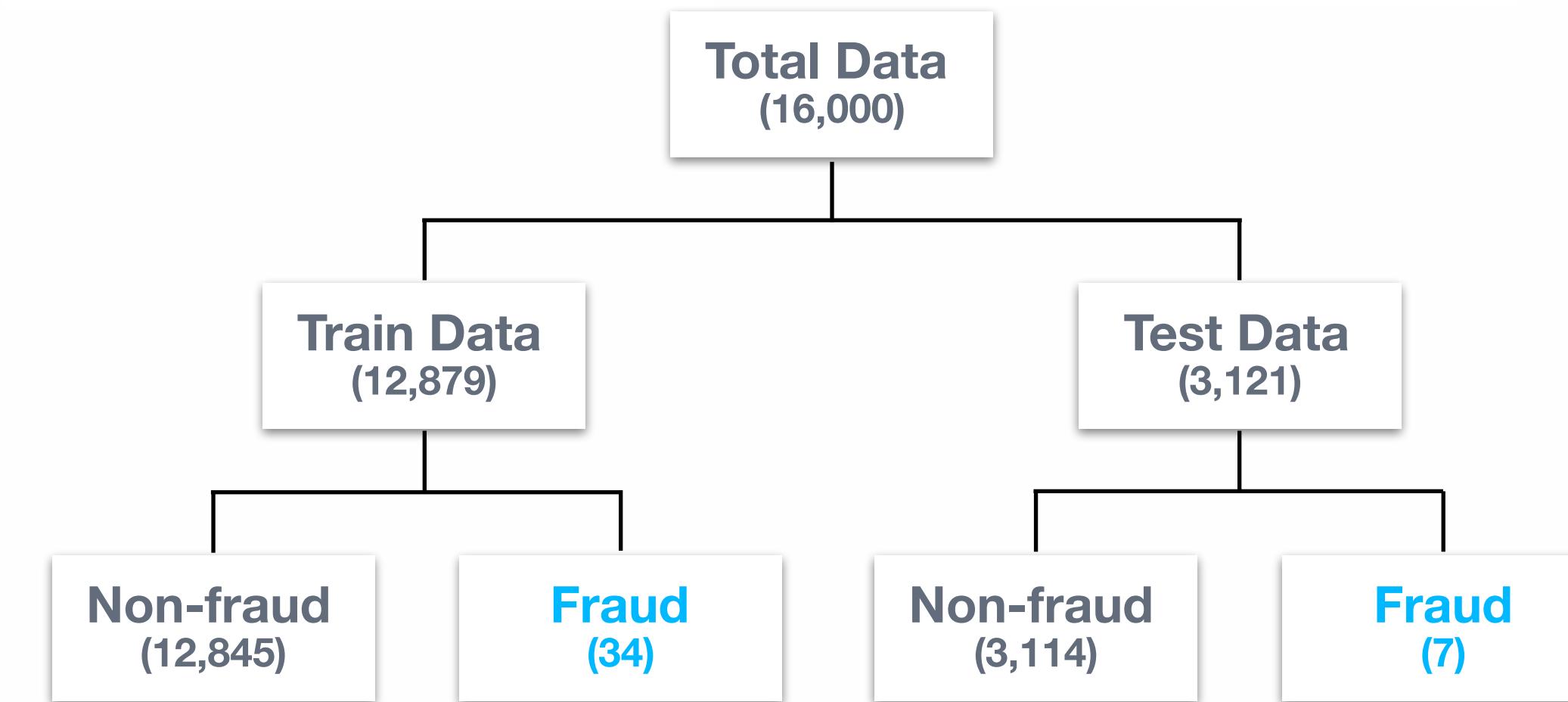
성능 목표 설정

Milestone

	fraud_YN	0	1	2	3
0	0	0	2	1	3
1	0	0	1	0	4
2	0	0	1	0	4
3	0	0	3	1	3
4	0	0	1	0	1
...	...	...	...	...	...
15995	0	0	2	0	2
15996	0	0	2	0	2
15997	0	0	2	1	2
15998	0	0	2	0	2
15999	0	0	2	0	2

16000 rows × 25 columns

	test_set	0	-1	0
0	0	-1	0	
1	1	0	1	
2	1	0	1	
3	0	-1	0	
4	0	-1	0	
...	...	...	...	
15995	0	-1	0	
15996	0	-1	0	
15997	0	-1	1	
15998	0	-1	0	
15999	0	-1	0	



# 프로젝트 정의

## SOCAR Data Set

### 문제 정의

알고리즘 선택

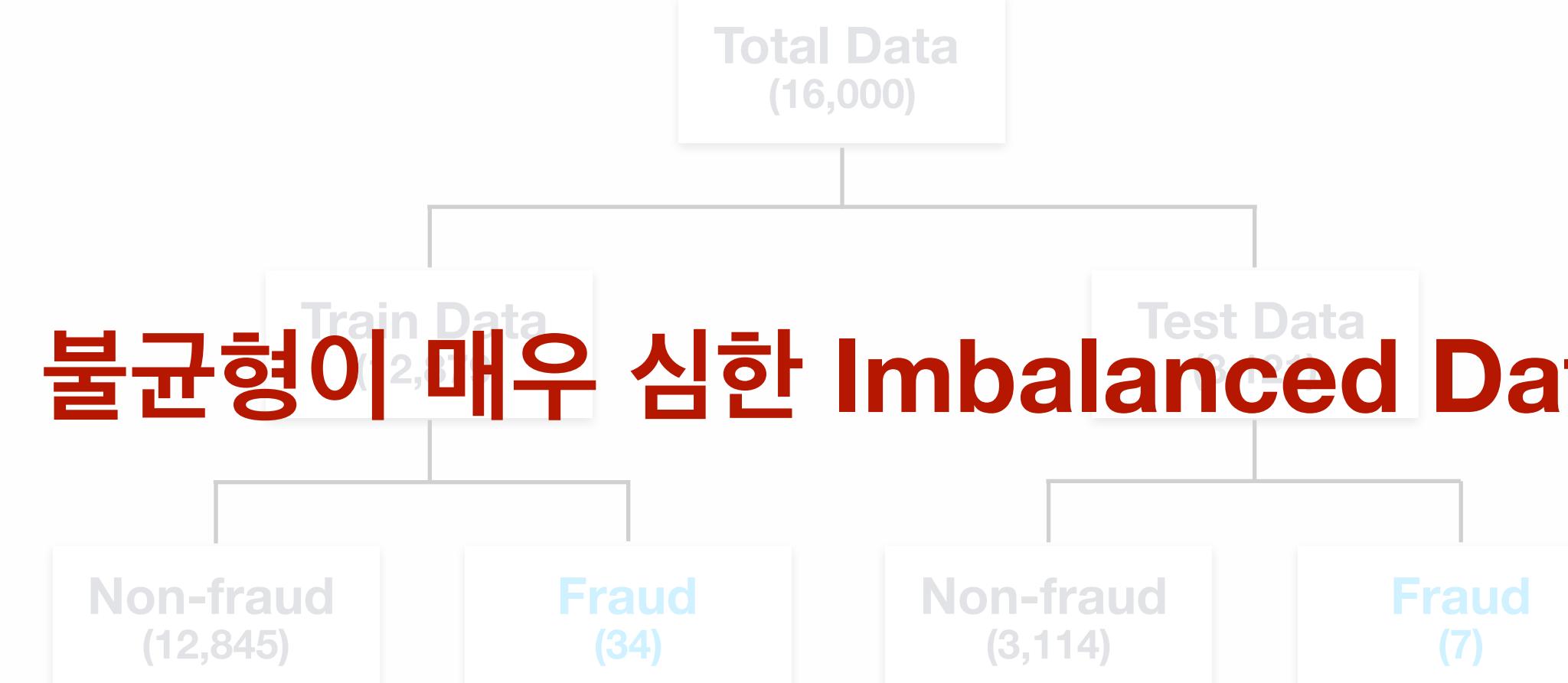
Sampling 선택

성능 목표 설정

Milestone

	fraud_YN	0	1	2	3	
0	0	0	2	1	3	
1	0	0	1	0	4	
2	0	0	1	0	4	
3	0	0	3	1	3	
4	0	0	1	0	1	
...	...	...	...	...	...	...
15995	0	0	2	0	2	
15996	0	0	2	0	2	
15997	0	0	2	1	2	
15998	0	0	2	0	2	
15999	0	0	2	0	2	

16000 rows × 25 columns



# 프로젝트 정의

## 문제 유형 정의

### 문제 정의

알고리즘 선택

Sampling 선택

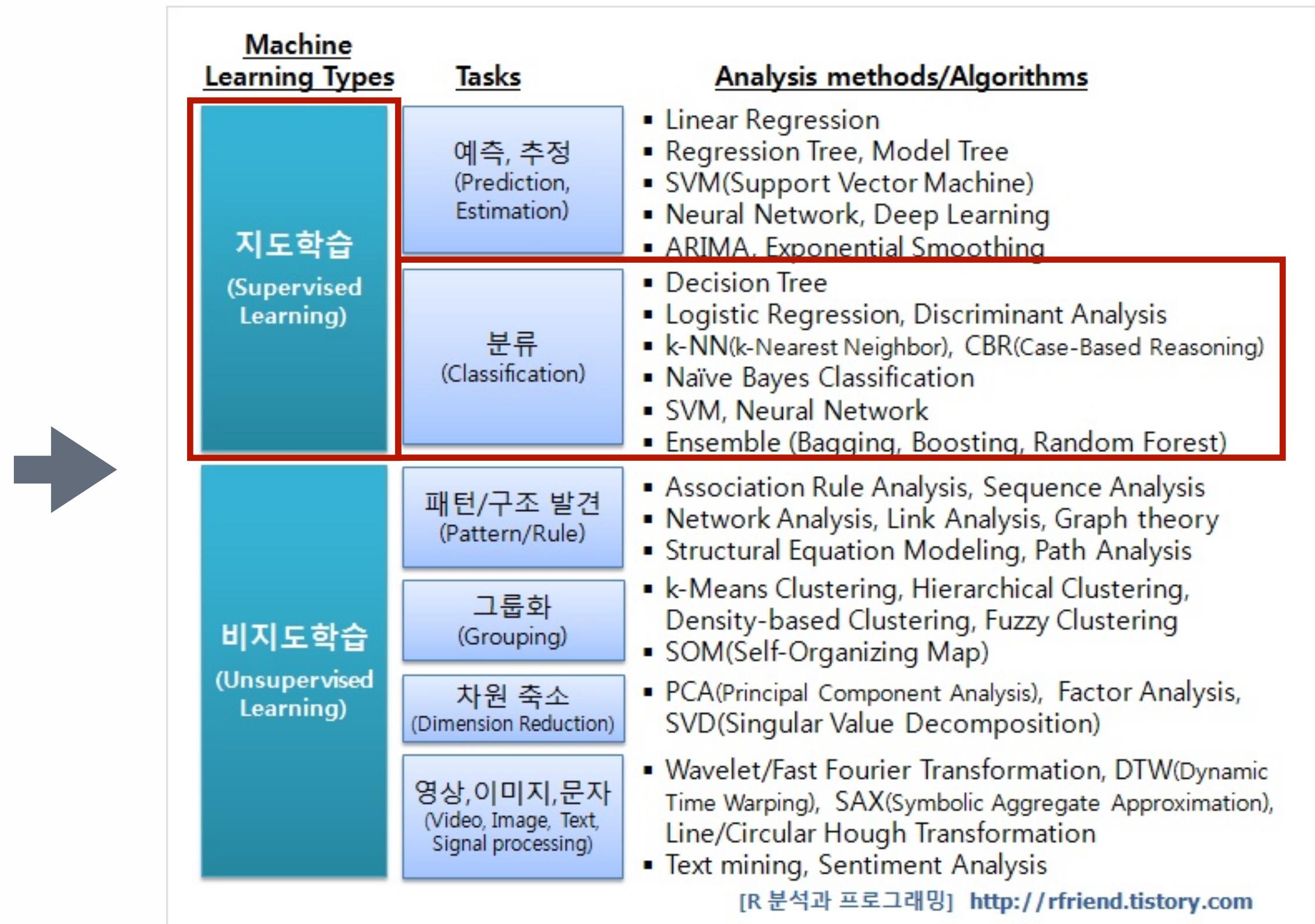
성능 목표 설정

Milestone

Labeling

+

Fraud / Non-fraud



[R 분석과 프로그래밍] <http://rfriend.tistory.com>

# 프로젝트 정의

## 성능 측정 방법 정의

### 문제 정의

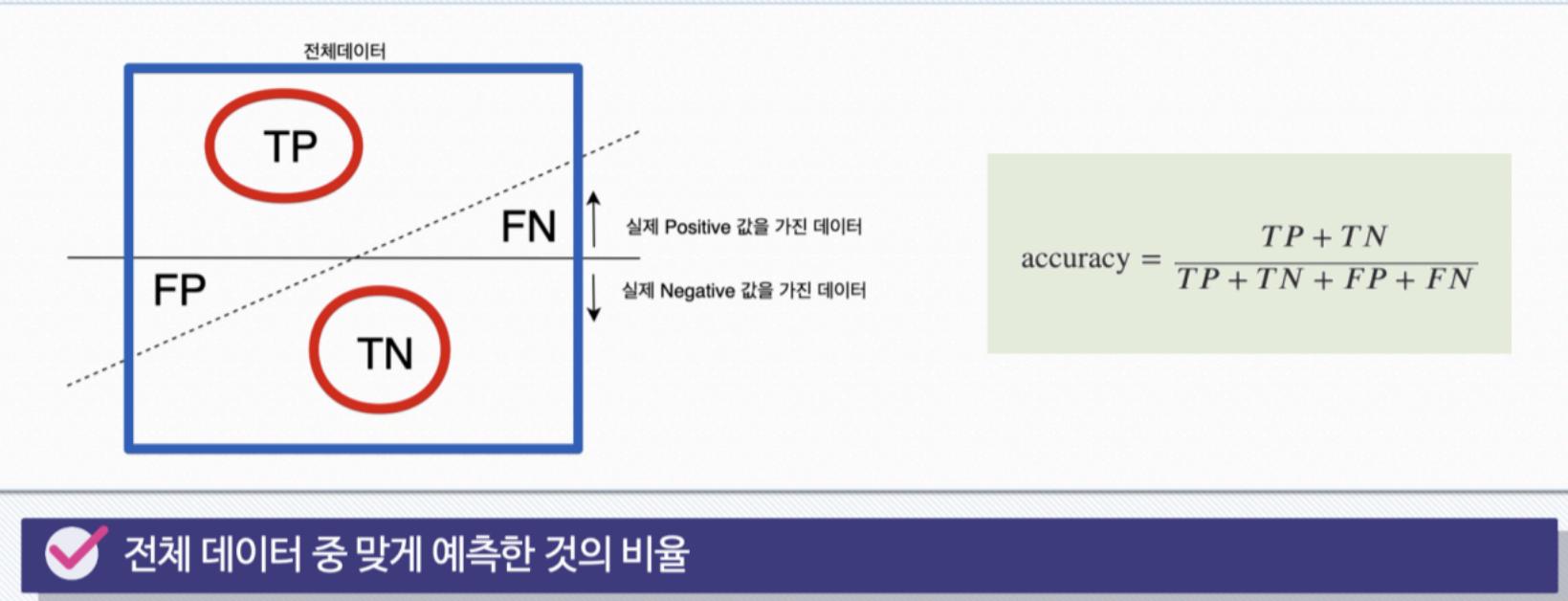
### 알고리즘 선택

### Sampling 선택

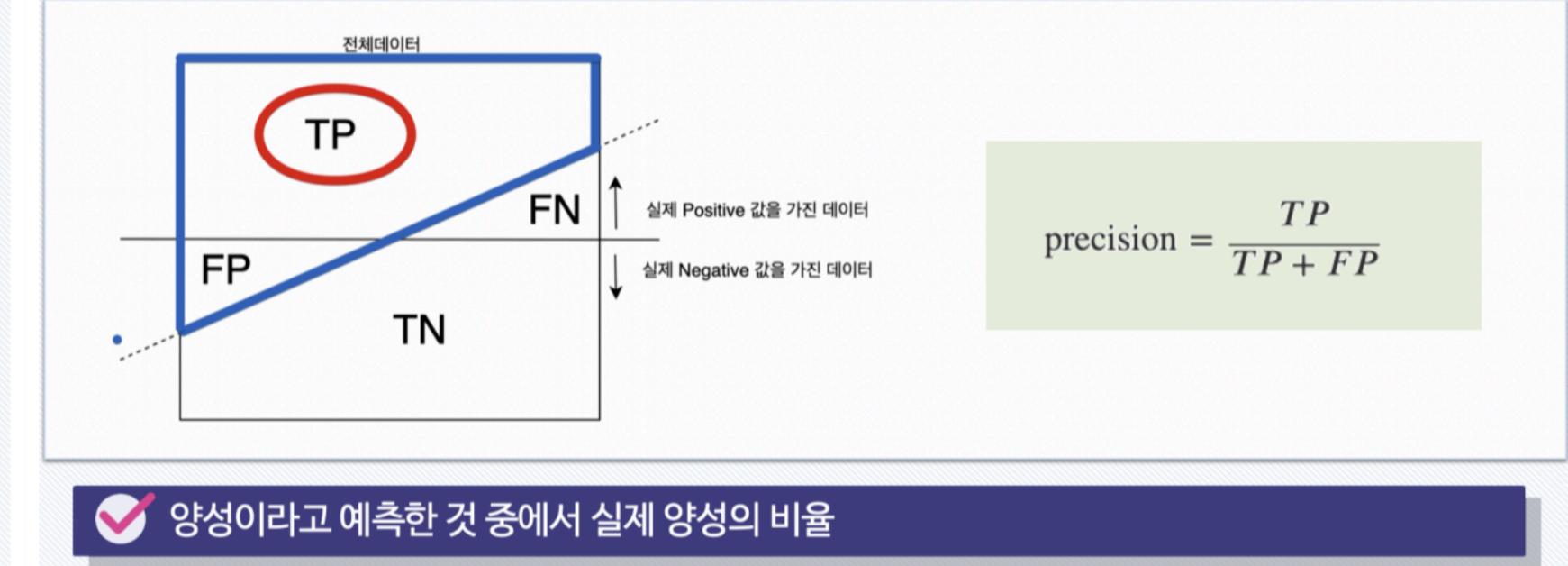
### 성능 목표 설정

### Milestone

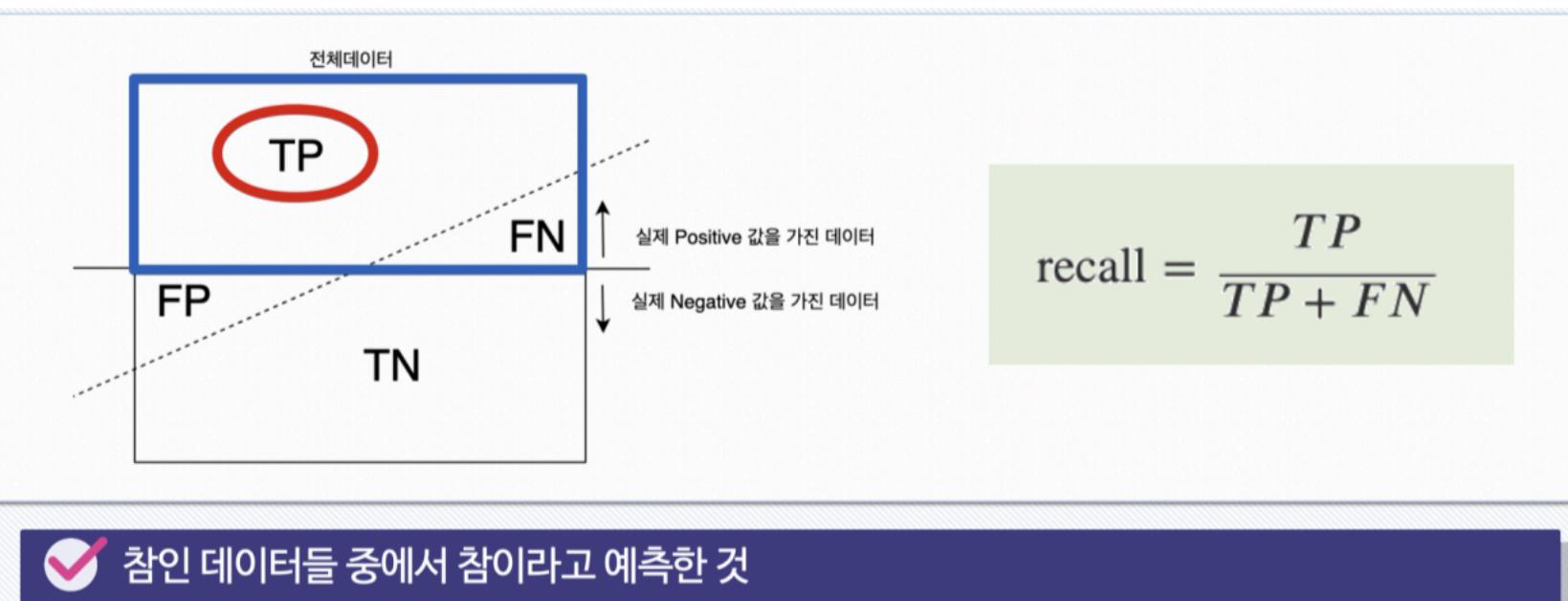
#### Accuracy



#### Precision



#### Recall



#### F1-score

$$F_\beta = (1 + \beta^2) (\text{precision} \times \text{recall}) / (\beta^2 \text{precision} + \text{recall})$$

→ F-Score에서 beta를 1로 두면 F1-score임

$$F_1 = 2 \cdot \text{precision} \cdot \text{recall} / (\text{precision} + \text{recall})$$

✓ F1-score는 Recall과 Precision을 결합한 지표

✓ Recall과 Precision이 어느 한쪽으로 치우치지 않고 둘다 높은 값을 가질 수록 높은 값을 가짐

# 프로젝트 정의

## 성능 측정 방법 정의

### 문제 정의

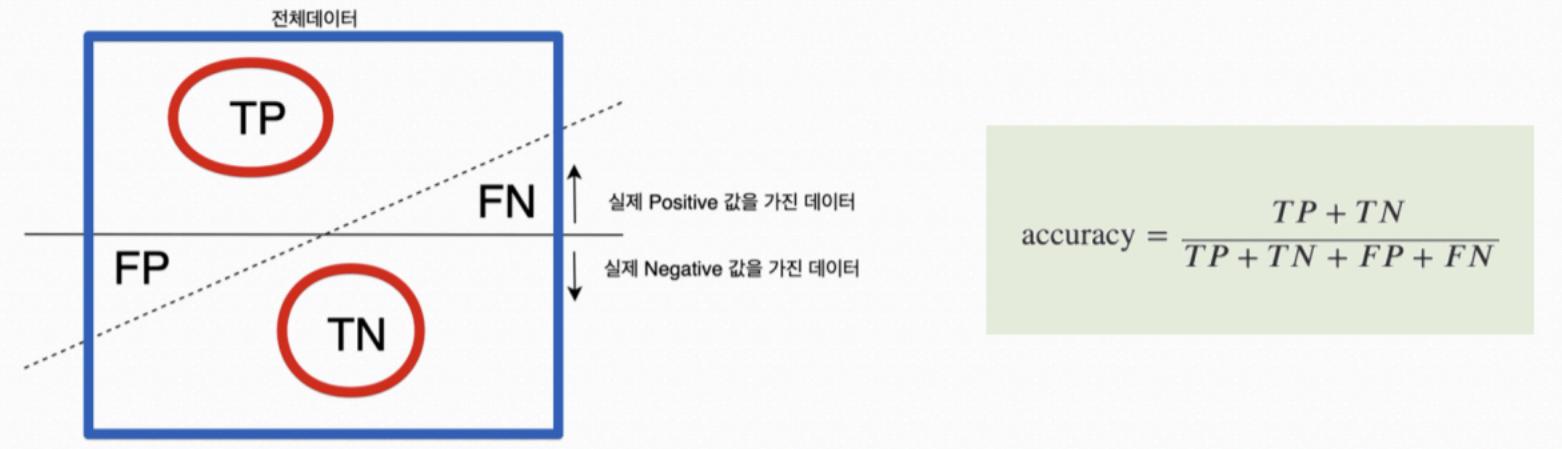
### 알고리즘 선택

### Sampling 선택

### 성능 목표 설정

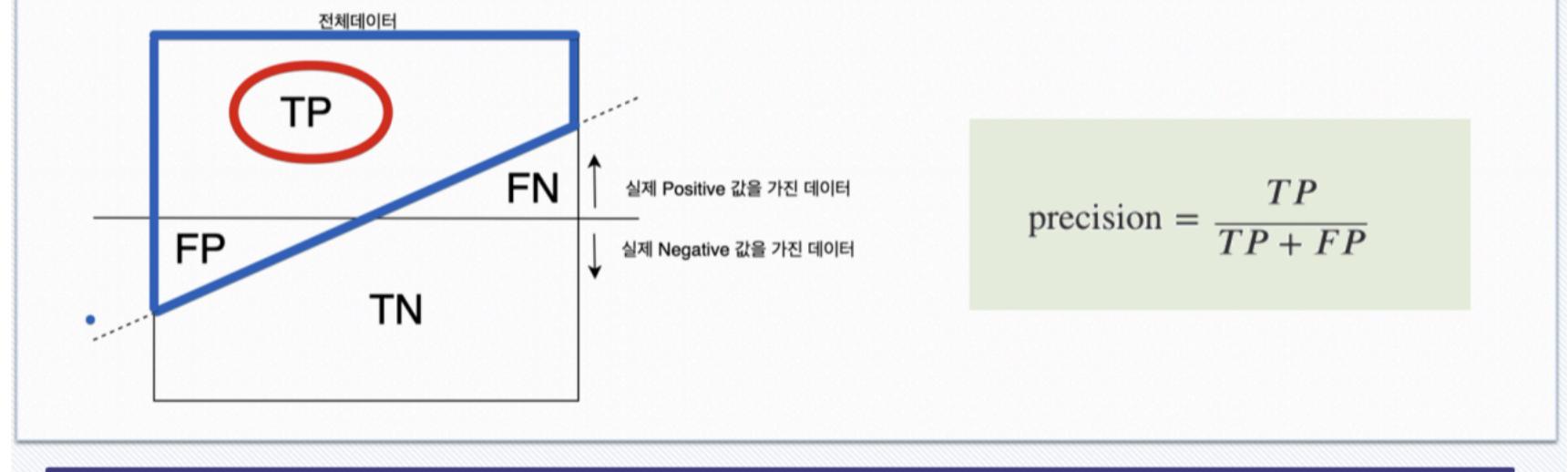
### Milestone

#### Accuracy



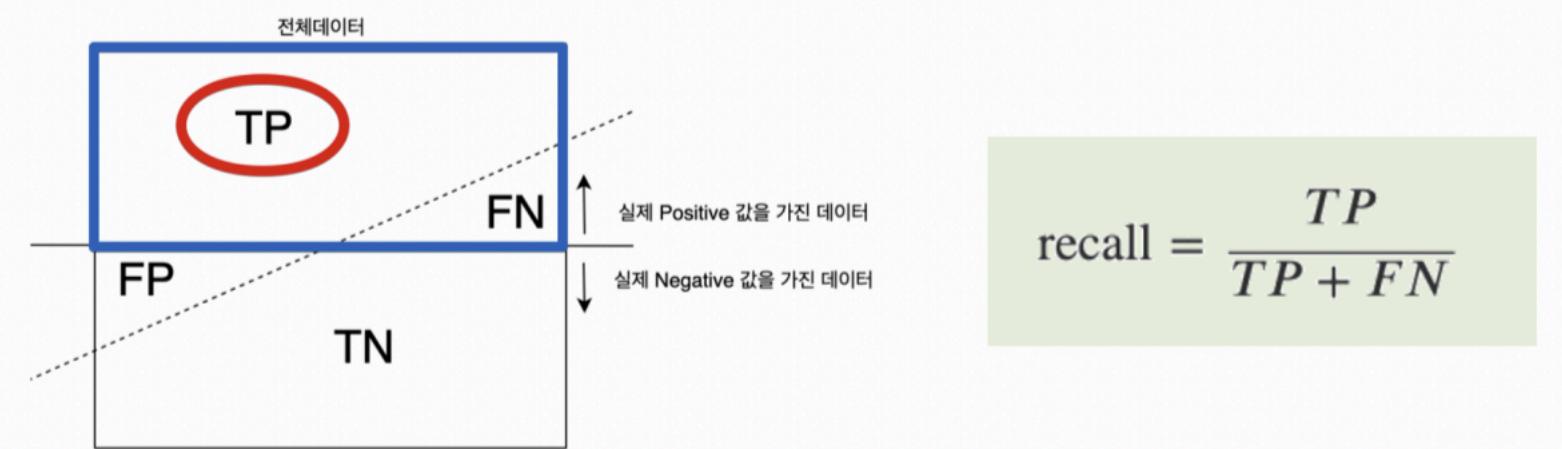
✓ 전체 데이터 중 맞게 예측한 것의 비율

#### Precision



✓ 양성이라고 예측한 것 중에서 실제 양성의 비율

#### Recall



✓ 참인 데이터들 중에서 참이라고 예측한 것

#### F1-score

$$F_\beta = (1 + \beta^2) (\text{precision} \times \text{recall}) / (\beta^2 \text{precision} + \text{recall})$$

→ F-Score에서 beta를 1로 두면 F1-score임

$$F_1 = 2 \cdot \text{precision} \cdot \text{recall} / (\text{precision} + \text{recall})$$

✓ F1-score는 Recall과 Precision을 결합한 지표

✓ Recall과 Precision이 어느 한쪽으로 치우치지 않고 둘다 높은 값을 가질 수록 높은 값을 가짐

$$\text{Recall} = \frac{\text{사기 예측 정답 수}}{\text{실제 사기 수}} = ?$$

자료 출처 : 패스트캠퍼스 데이터사이언스 취업완성 스쿨 강의 자료

# 프로젝트 정의

## 머신러닝 알고리즘

문제 정의

## 알고리즘 선택

Sampling 선택

성능 목표 설정

Milestone

### 알고리즘 선택 시 고려 사항

1. 정확성
2. 학습 시간
3. 사용 편의성
4. 매개 변수 수

### 사용할 지도학습 알고리즘 종류

1. Logistic Regression
2. Decision Tree
3. Random Forest
4. Support Vector Machine
5. Linear SVM

# 프로젝트 정의

문제 정의

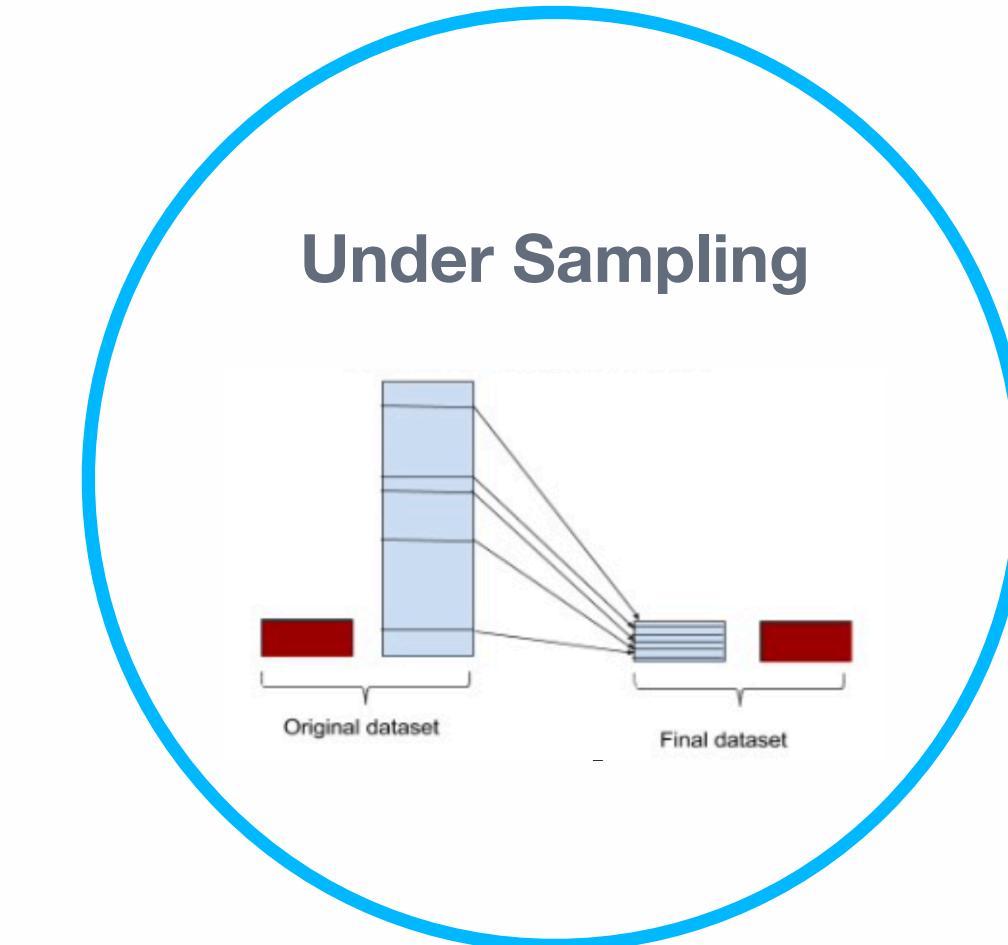
알고리즘 선택

## Sampling 선택

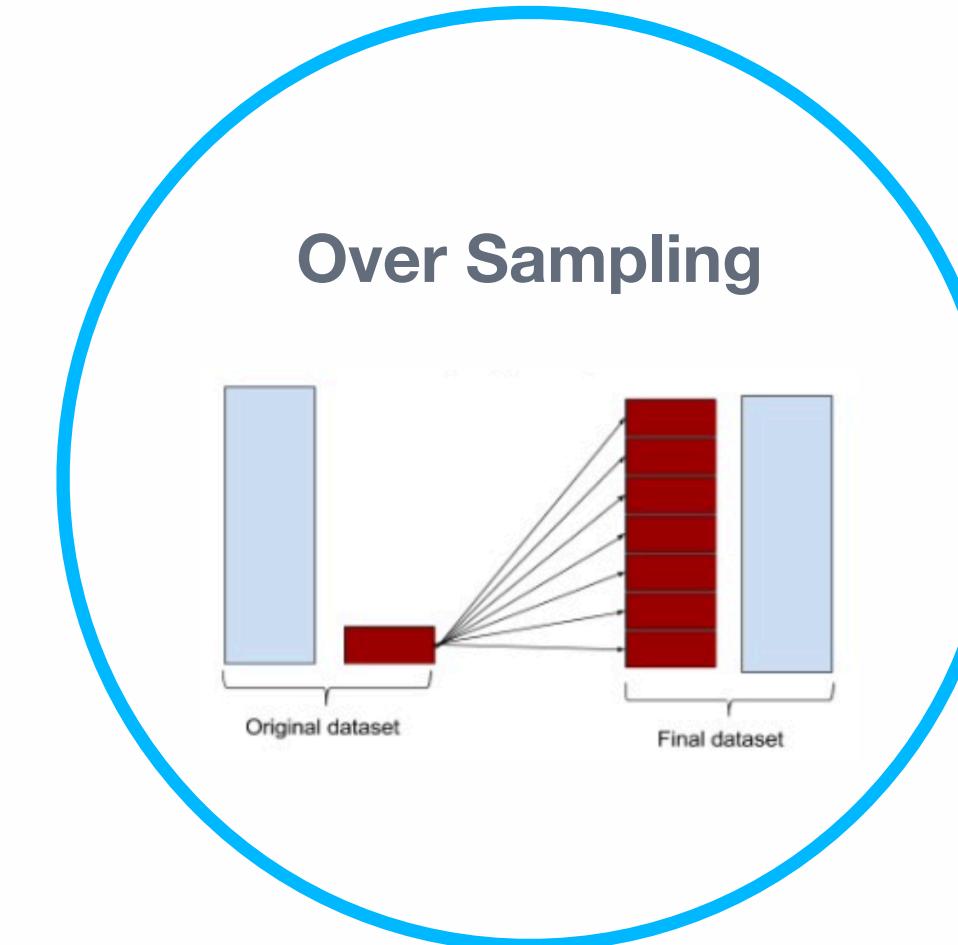
성능 목표 설정

Milestone

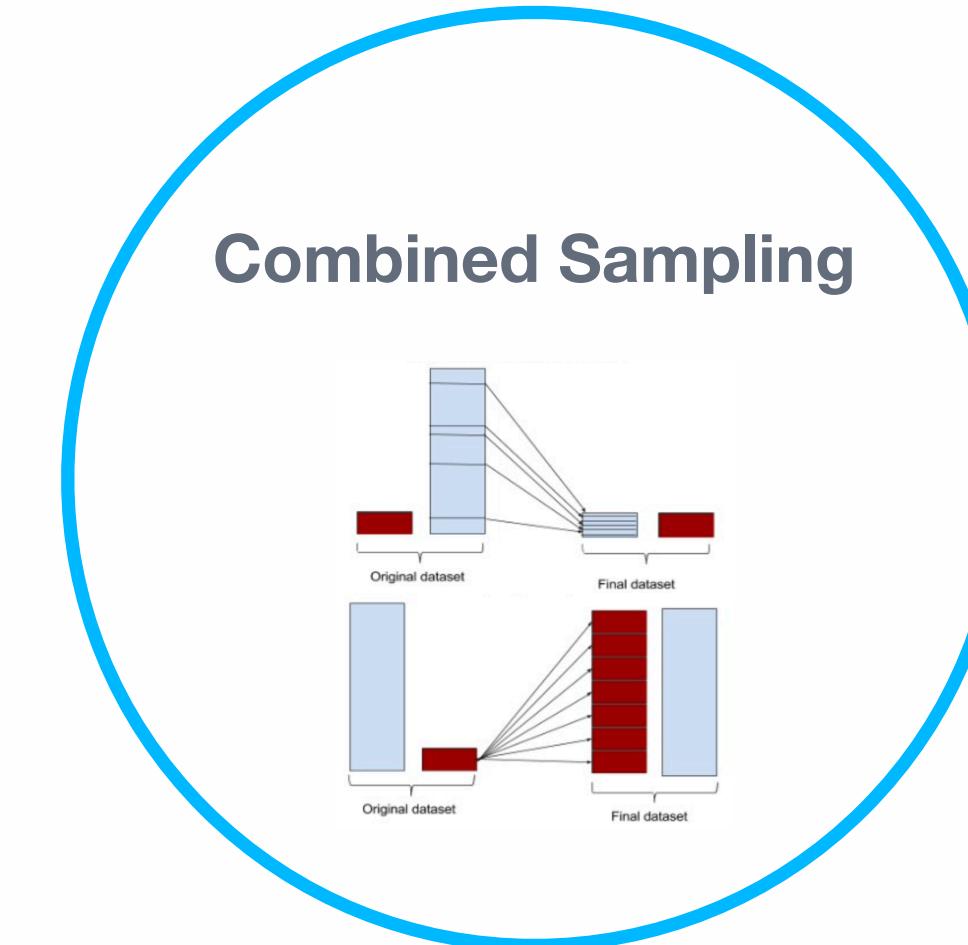
### Sampling 선택



Under Sampling



Over Sampling



Combined Sampling

- Random under sampling
- Edited Nearest Neighbors(ENN)
- NearMiss

- Random over sampling
- SMOTE
- Borderline SMOTE

- SMOTE + ENN
- SMOTE + NearMiss
- SMOTE + TomekLinks

# 프로젝트 정의

## 모델 성능 목표

### 문제 정의

#### 비교 대상

	model name	train accuracy	train precision	train recall	test accuracy	test precision	test recall
0	LogisticRegression	0.882353	0.861111	0.911765	0.546299	0.002823	0.571429
1	SupportVectorMachine	0.941176	0.916667	0.970588	0.492791	0.003153	0.714286
2	RandomForest	0.926471	0.967742	0.882353	0.530279	0.001367	0.285714

### 알고리즘 선택

### Sampling 선택

### 성능 목표 설정

#### 목표 성능

	model name	train accuracy	train precision	train recall	test accuracy	test precision	test recall
0	LogisticRegression	0.830167	0.801122	0.878396	0.700737	0.004278	0.571429

### Milestone

Train fraud data => 31 / 34      Test fraud data => 6 / 7

# 프로젝트 정의

## 프로젝트 Milestone

문제 정의

알고리즘 선택

Sampling 선택

성능 목표 설정

Milestone

1주차

(2020.12.28~2021.01.03)

2주차

(2021.01.04~2021.01.10)

3주차

(2021.01.11~2021.01.17)

D-day

(2021.01.18)



- 보험사기 관련 도메인 지식 학습
- EDA & Preprocessing
- Feature Selection

- Model Test
- Parameter 설정

- n차 EDA & Preprocessing
- Model Test
- Parameter 설정
- Model 최종 선정

- 프로젝트 최종 발표
- 피드백 반영
- README 작성
- 최종 산출물 Github push

# 프로젝트 진행

# 프로젝트 진행

## Data Set

## EDA

## PreProcessing

## Model Test

## Conclusion

SOCAR

### SOCAR Data Set 구성

1. [Redacted]	7. [Redacted]	14. [Redacted]	21. [Redacted]
2. [Redacted] JV	8. [Redacted]	15. [Redacted]	22. [Redacted]
3. [Redacted]	10. [Redacted]	16. [Redacted]	23. [Redacted]
4. [Redacted]	11. [Redacted]	17. [Redacted]	24. [Redacted]
5. [Redacted] ment	12. [Redacted]	18. [Redacted]	25. [Redacted]
6. [Redacted] nt	13. [Redacted]	19. [Redacted]	
		20. [Redacted]	
		21. [Redacted]	
		22. [Redacted]	
		23. [Redacted]	
		24. [Redacted]	
		25. [Redacted]	

# 프로젝트 진행

## 데이터 탐색

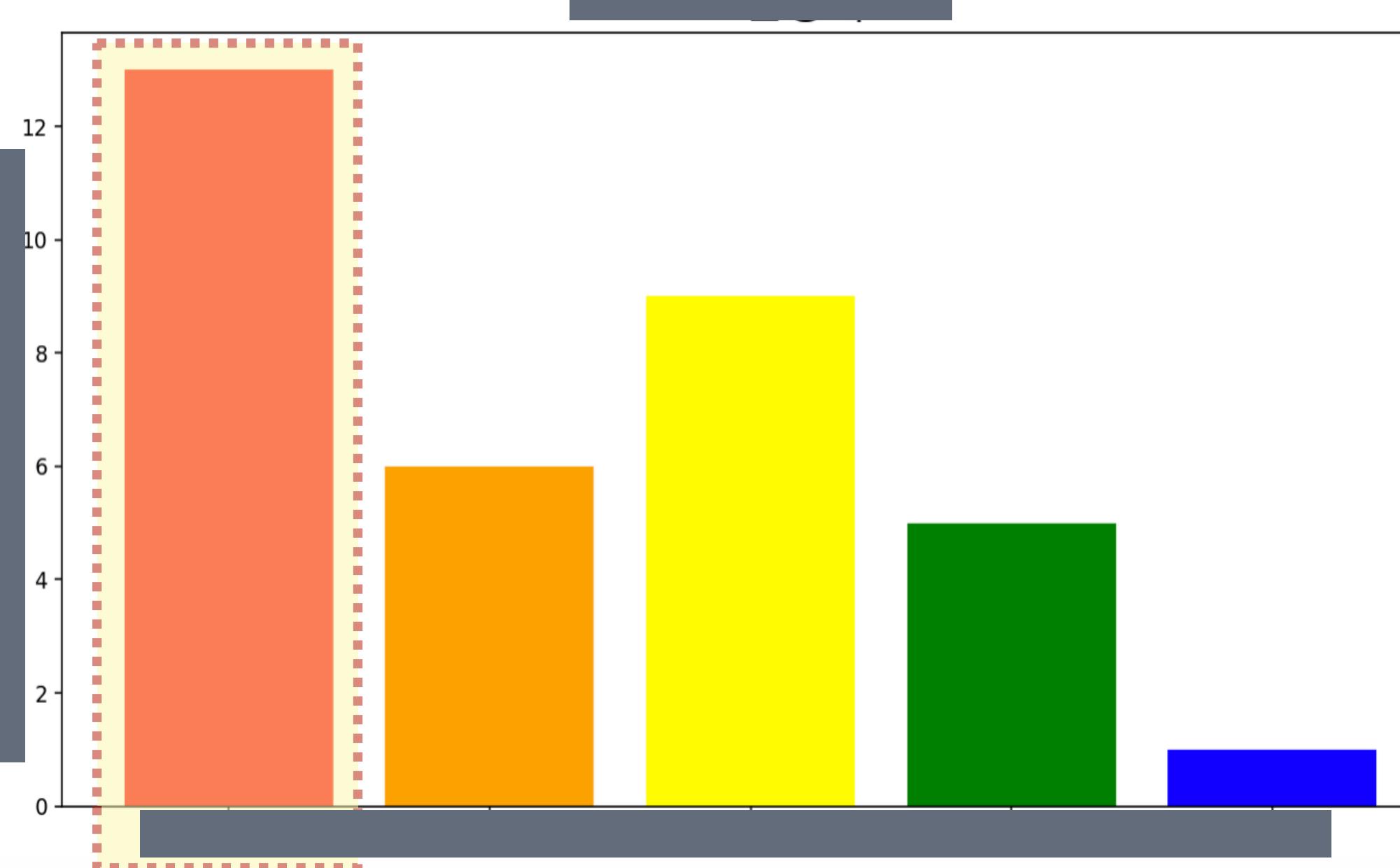
• Data Set

• EDA

• PreProcessing

• Model Test

• Conclusion



Insight

보험 사기범의 [REDACTED]



# 프로젝트 진행

## 데이터 탐색

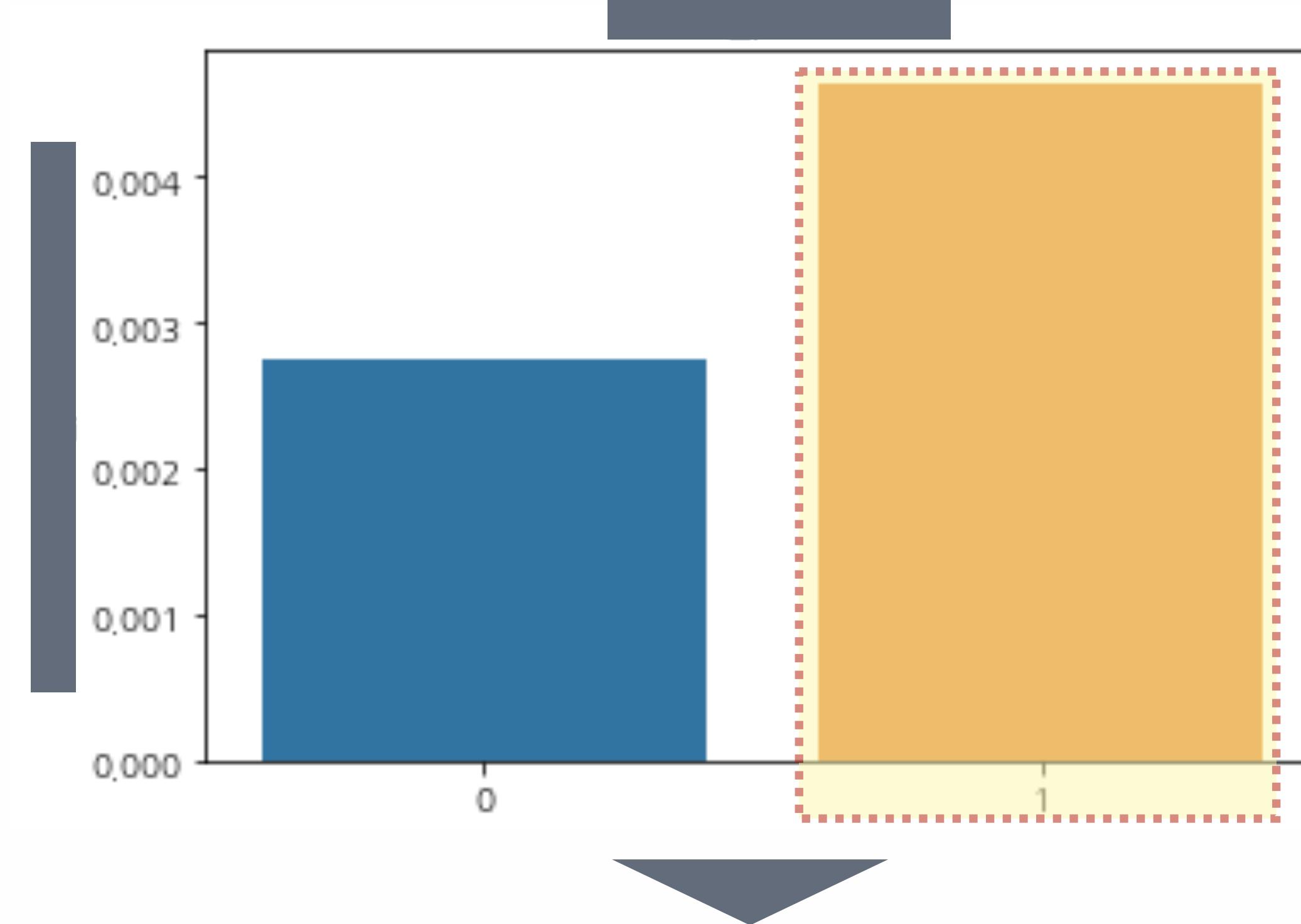
• Data Set

• EDA

• PreProcessing

• Model Test

• Conclusion



Insight

▶ 1일 경우 ‘뒤쿵’ 의심

# 프로젝트 진행

## 데이터 탐색

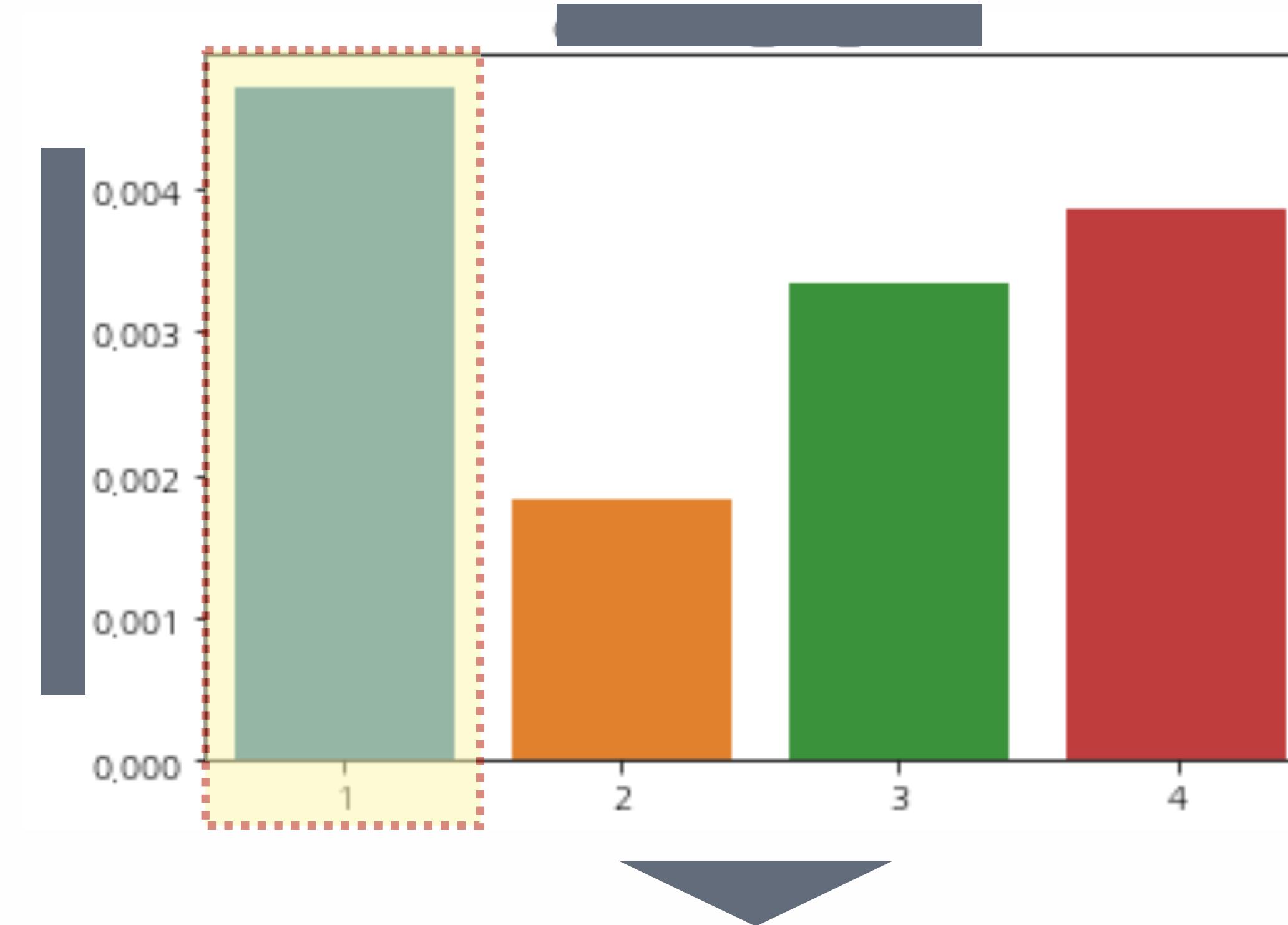
• Data Set

• EDA

• PreProcessing

• Model Test

• Conclusion



Insight

경우 보험 사기 범죄일 확률이 높음

# 프로젝트 진행

## 데이터 탐색

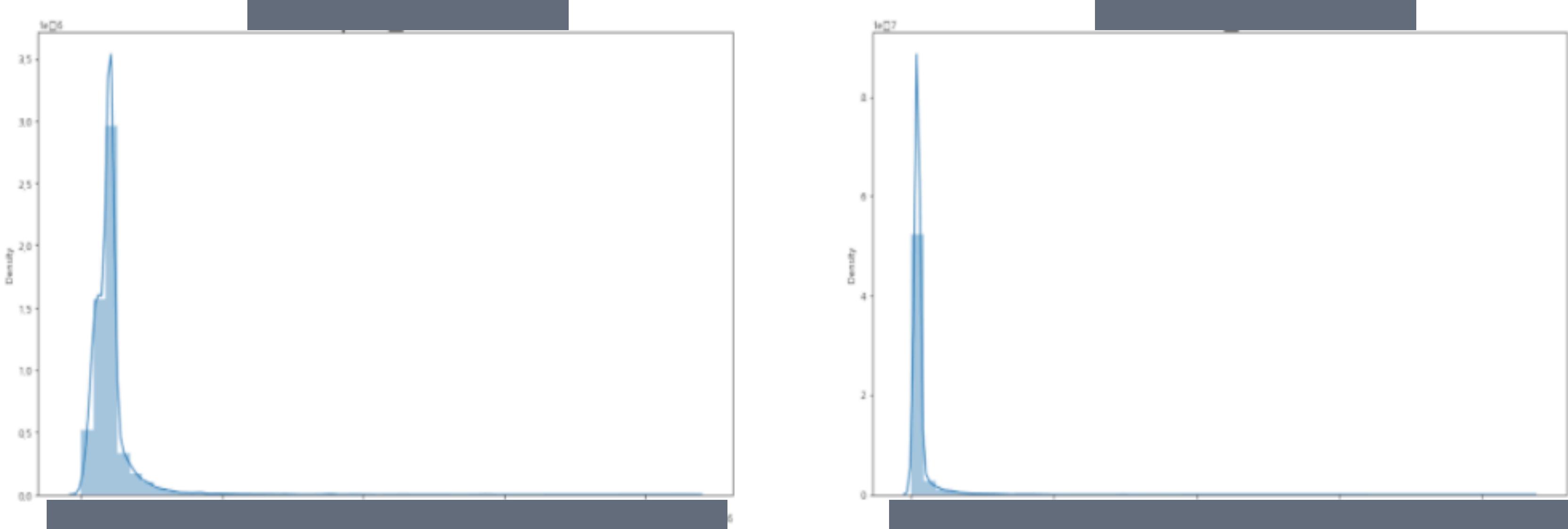
• Data Set

• EDA

• PreProcessing

• Model Test

• Conclusion



Insight

퍼져있는 범위가 매우 넓으므로 임의로 묶어서 categorical data로 변환 필요



# 프로젝트 진행

## 데이터 탐색

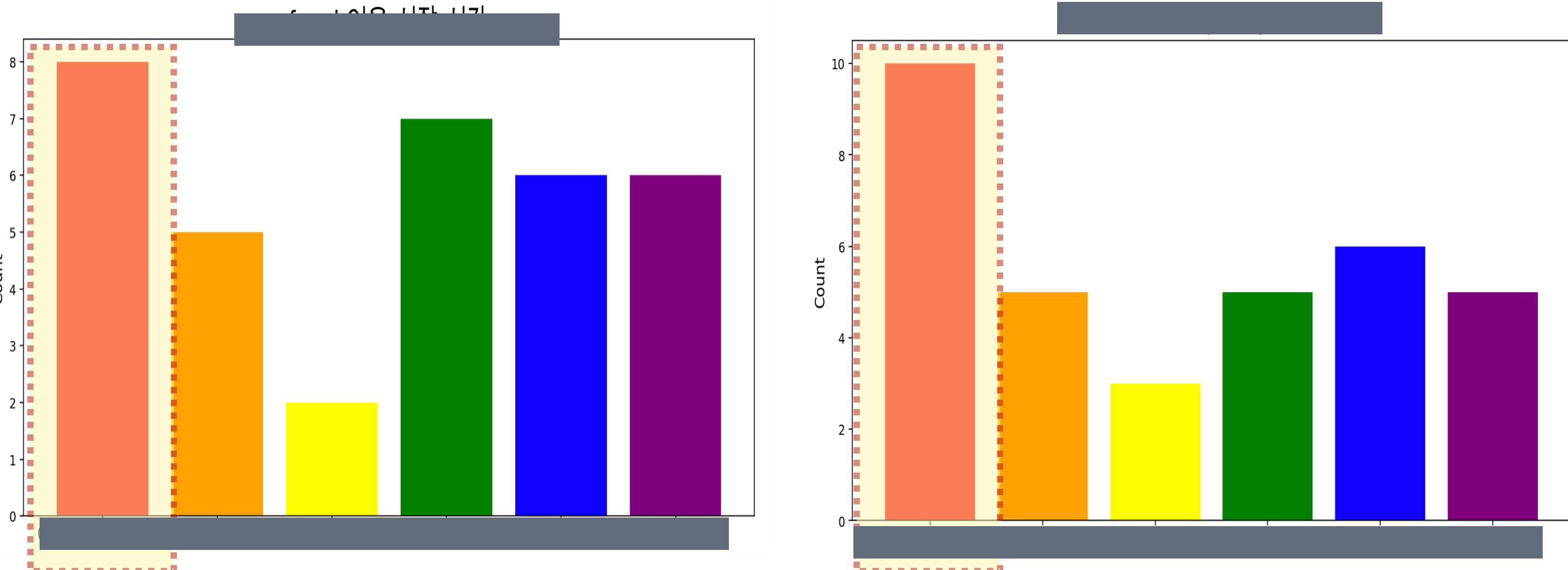
• Data Set

• EDA

• PreProcessing

• Model Test

• Conclusion

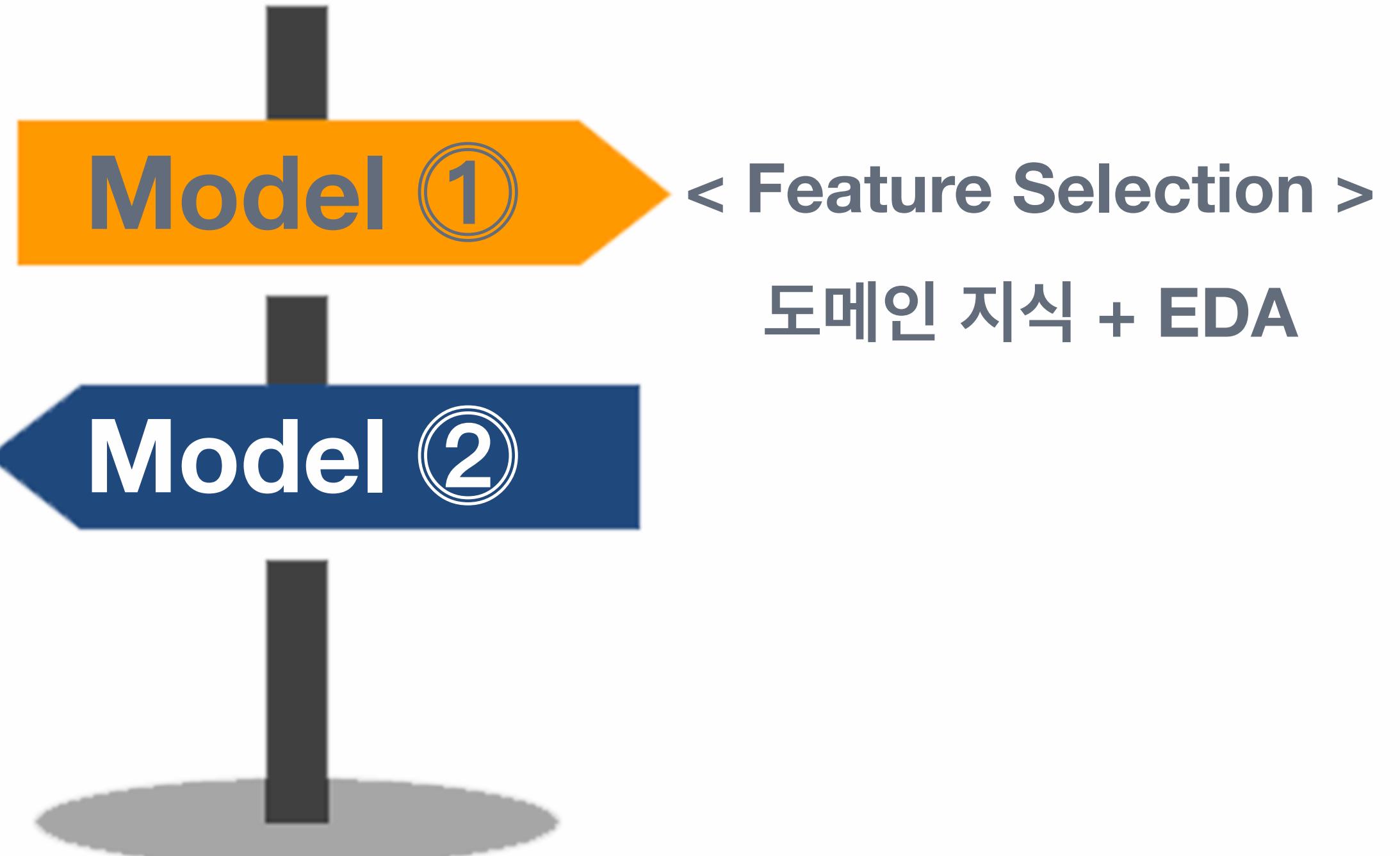


Insight

'[REDACTED]'과 '[REDACTED]'이 연속적인 category로 되어 있지 않으므로 다른 기준 적용 필요

- Data Set
- EDA
- Preprocessing
- Model Test
- Conclusion

# 서로 다른 방향으로 가다



- Data Set
- EDA
- Preprocessing
- Model Test
- Conclusion

# Model ①

### • Data Set

1. Train fraud data / Test fraud data 비율 조정

- 

### • EDA

2. 도메인 지식 및 EDA 기반 불필요 판단 컬럼 제거

- 

- 

## • Preprocessing

3. EDA 기반 noise 판단된 데이터 부분 제거

- 

- 

### • Model Test

4. 연속형 데이터 명목형으로 변환

- 

- 

### • Conclusion

5. OneHotEncoding

- 

- 

-

# 프로젝트 진행

## 데이터 전처리 - Model ① - 1. Train fraud data / Test fraud data 비율 조정

- Data Set
- EDA
- **Preprocessing**
- Model Test
- Conclusion



# 프로젝트 진행

• Data Set

• EDA

• Preprocessing

• Model Test

• Conclusion

SOCAR

## SOCAR Data Set 구성

1. [Redacted]  
[Redacted]  
[Redacted]

2. [Redacted]  
[Redacted]  
[Redacted]  
[Redacted]  
[Redacted]  
JV

3. [Redacted]  
[Redacted]  
[Redacted]

4. [Redacted]  
[Redacted]  
[Redacted]  
[Redacted]  
[Redacted]

5. [Redacted]  
[Redacted]  
[Redacted]

6. [Redacted]  
[Redacted]  
[Redacted]  
[Redacted]

7. [Redacted]  
[Redacted]  
[Redacted]

8. [Redacted]  
[Redacted]  
[Redacted]

9. [Redacted]  
[Redacted]  
[Redacted]

10. [Redacted]  
[Redacted]  
[Redacted]

11. [Redacted]  
[Redacted]  
[Redacted]

12. [Redacted]  
[Redacted]  
[Redacted]  
[Redacted]

13. [Redacted]  
[Redacted]  
[Redacted]  
[Redacted]

14. [Redacted]  
[Redacted]  
[Redacted]  
[Redacted]  
[Redacted]

15. [Redacted]  
[Redacted]  
[Redacted]

16. [Redacted]  
[Redacted]  
[Redacted]

17. [Redacted]  
[Redacted]  
[Redacted]  
[Redacted]  
[Redacted]

18. [Redacted]  
[Redacted]  
[Redacted]  
[Redacted]

19. [Redacted]  
[Redacted]  
[Redacted]

20. [Redacted]  
[Redacted]  
[Redacted]

21. [Redacted]  
[Redacted]  
[Redacted]  
[Redacted]  
[Redacted]

22. [Redacted]  
[Redacted]  
[Redacted]  
[Redacted]  
[Redacted]

23. [Redacted]  
[Redacted]  
[Redacted]  
[Redacted]  
[Redacted]

24. [Redacted]  
[Redacted]  
[Redacted]  
[Redacted]  
[Redacted]

25. [Redacted]  
[Redacted]  
[Redacted]

# 프로젝트 진행

## 데이터 전처리 - Model ① - 2. 도메인 지식 및 EDA 기반 불필요 판단 컬럼 제거

● Data Set

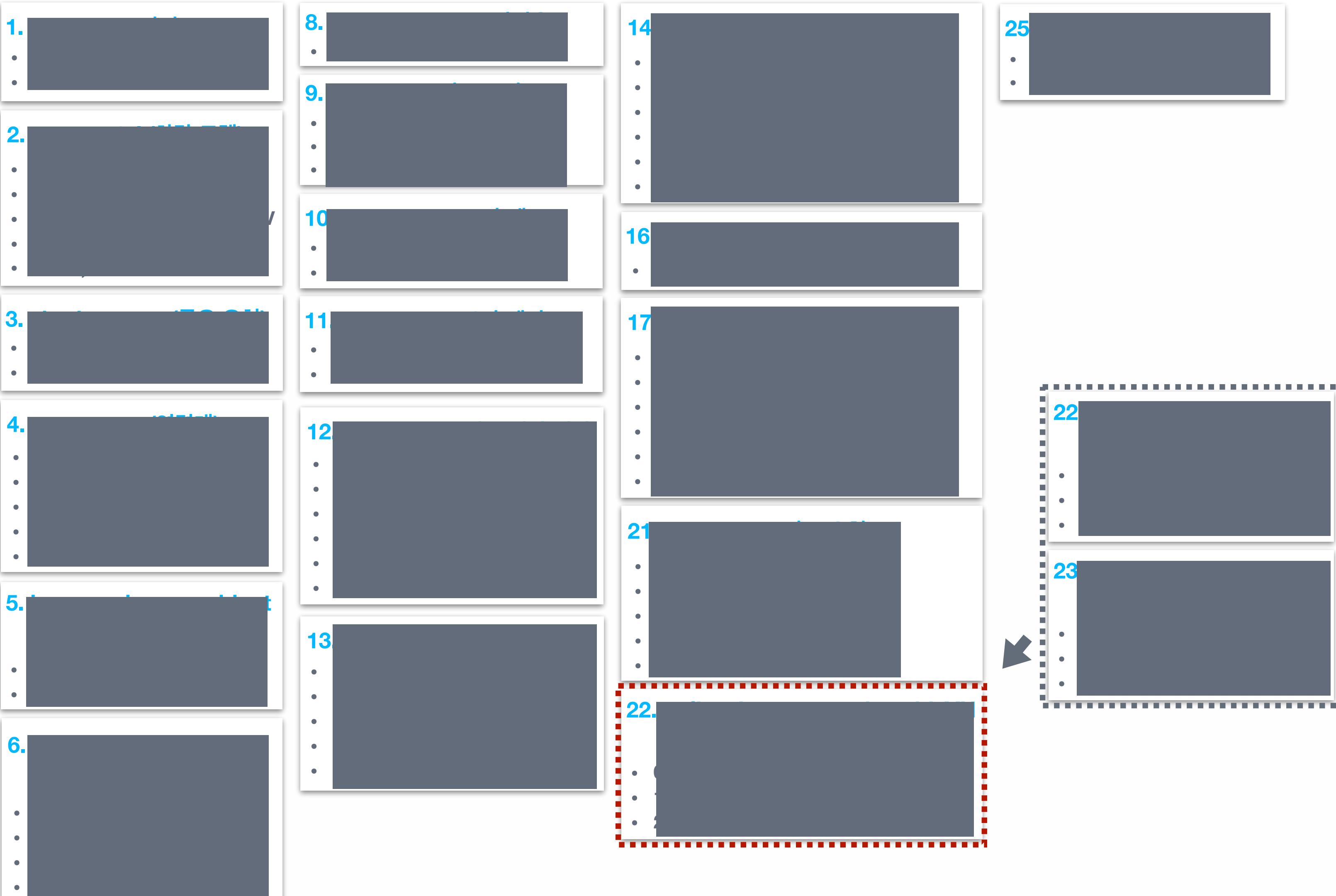
● EDA

● Preprocessing

● Model Test

● Conclusion

SOCAR



# 프로젝트 진행

## 데이터 전처리 - Model ① - 2. 도메인 지식 및 EDA 기반 불필요 판단 컬럼 제거

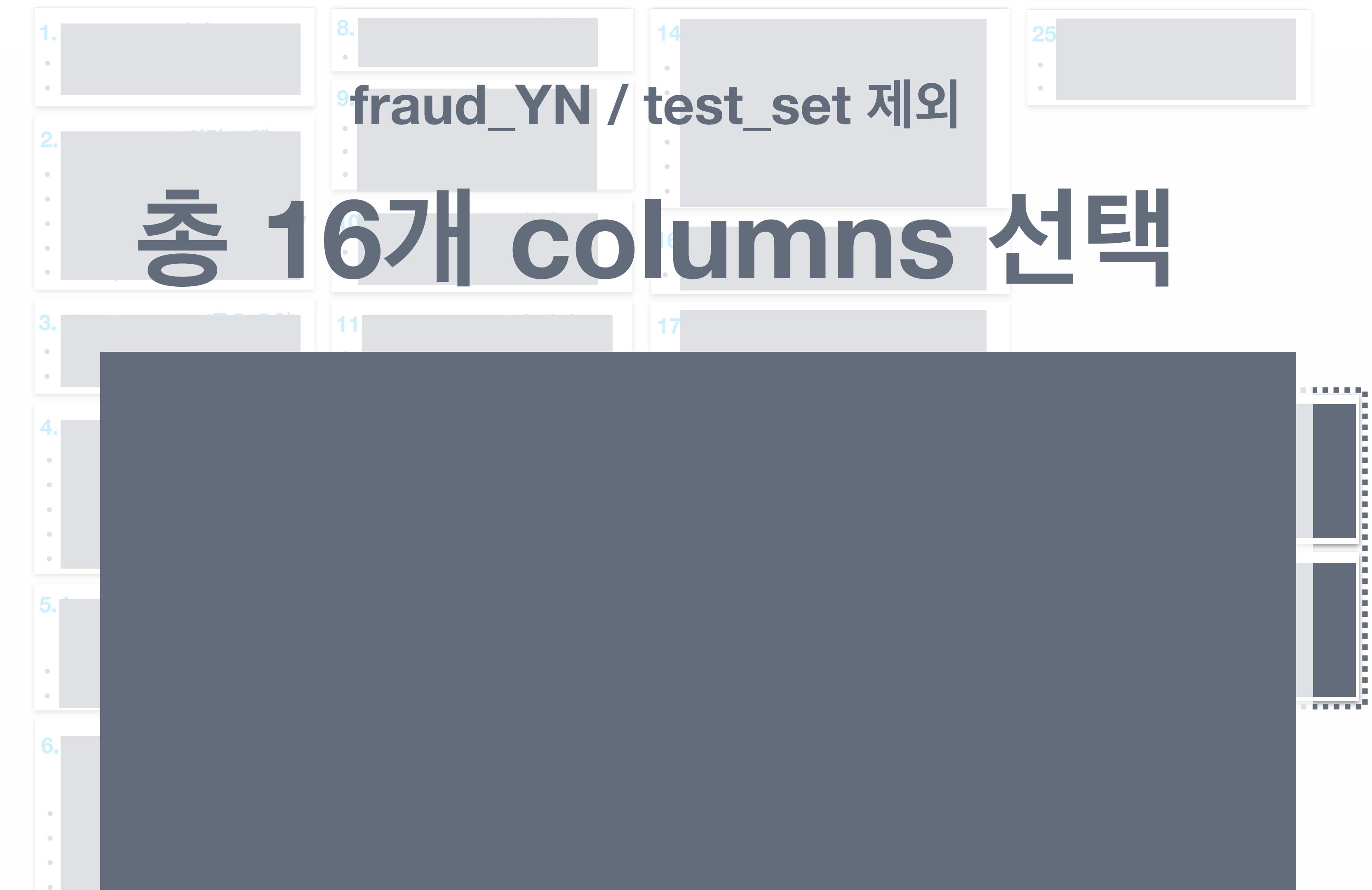
• Data Set

• EDA

• Preprocessing

• Model Test

• Conclusion



# 프로젝트 진행

## 데이터 전처리 - Model ① - 3. EDA 기반 noise 판단된 데이터 부분 제거

### • Data Set

- non-fraud train data의 [REDACTED] '-1' 데이터 개수 : 26개
- fraud train data의 [REDACTED] '-1' 데이터 개수 : 0개

### • EDA

```
socar_dataset = socar_dataset[socar_dataset['[REDACTED]'] != -1] # -1 : 알 수 없음 제거
```

### • Preprocessing

- non-fraud 전체 data의 [REDACTED] 데이터 개수 : 491개
- fraud 전체 data의 [REDACTED] 데이터 개수 : 0개

### • Model Test

- non-fraud 전체 data의 [REDACTED] 데이터 개수 : 39개
- non-fraud 전체 data의 [REDACTED] 데이터 개수 : 0개
- fraud 전체 data의 [REDACTED] 데이터 개수 : 0개
- fraud 전체 data의 [REDACTED] 데이터 개수 : 0개

### • Conclusion

```
socar_dataset = socar_dataset[socar_dataset['[REDACTED]']]
```

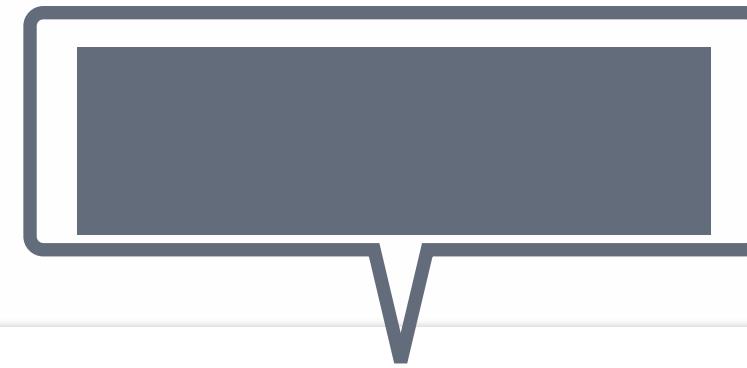
# 프로젝트 진행

## 데이터 전처리 - Model ① - 3. EDA 기반 noise 판단된 데이터 부분 제거

### • Data Set

- non-fraud 전체 data의 데이터 개수 : 584개
- non-fraud 전체 data의 데이터 개수 : 0개
- non-fraud 전체 data의 데이터 개수 : 249개
- fraud 전체 data의 데이터 개수 : 0개
- fraud 전체 data의 데이터 개수 : 0개
- fraud 전체 data의 데이터 개수 : 0개

### • EDA



### • Preprocessing

```
socar_dataset = socar_dataset[socar_dataset]
```

### • Model Test

- non-fraud 전체 data의 데이터 개수 : 102개
- non-fraud 전체 data의 데이터 개수 : 269개
- fraud 전체 data의 데이터 개수 : 0개
- fraud 전체 data의 데이터 개수 : 0개

### • Conclusion

```
socar_dataset = socar_dataset[socar_dataset]
```

# 프로젝트 진행

## 데이터 전처리 - Model ① - 4. 연속형 데이터 명목형으로 변환

• Data Set

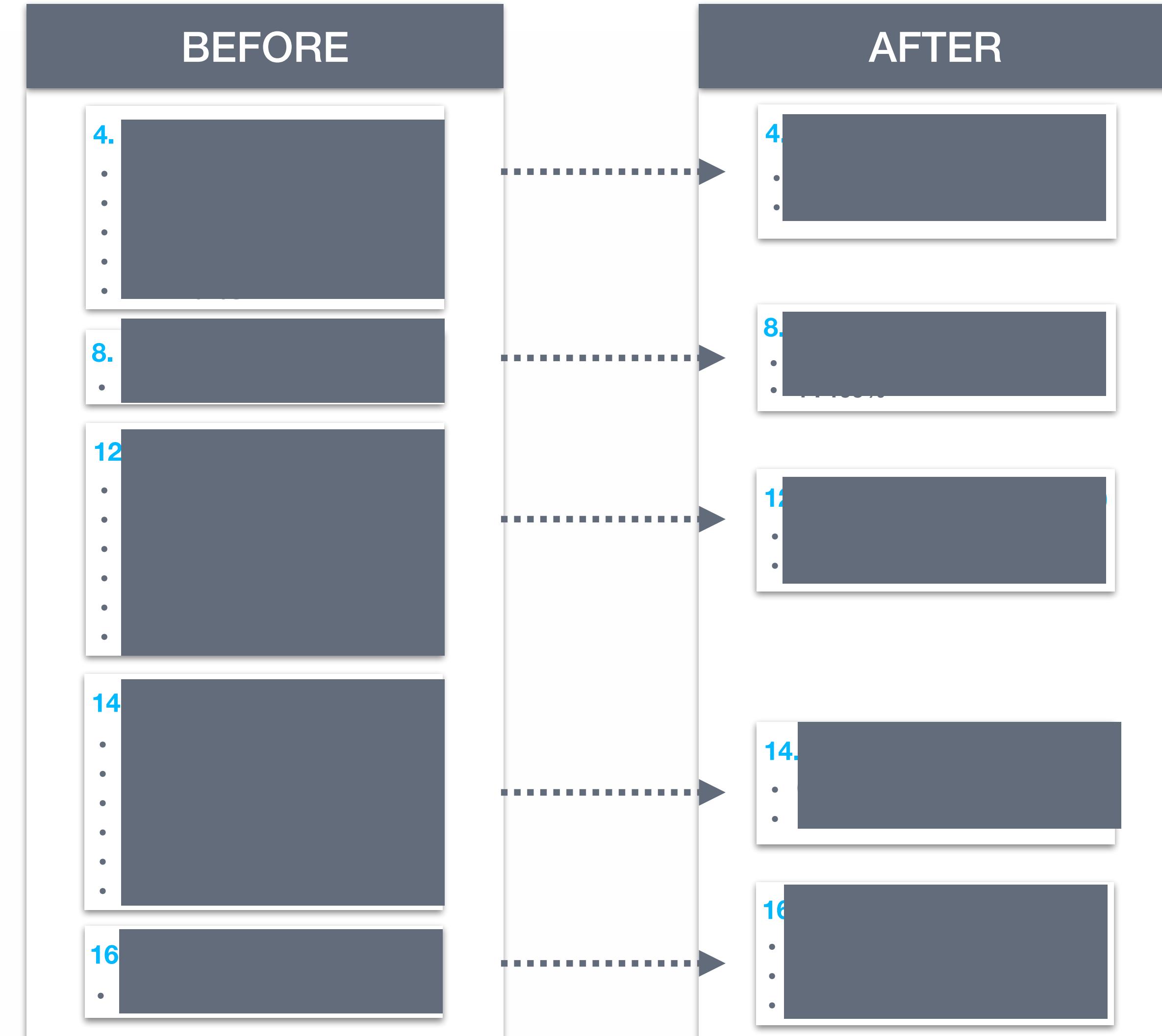
• EDA

• Preprocessing

• Model Test

• Conclusion

SOCAR



# 프로젝트 진행

## 데이터 전처리 - Model ① - 5. OneHotEncoding

• Data Set



• EDA

• Preprocessing

• Model Test

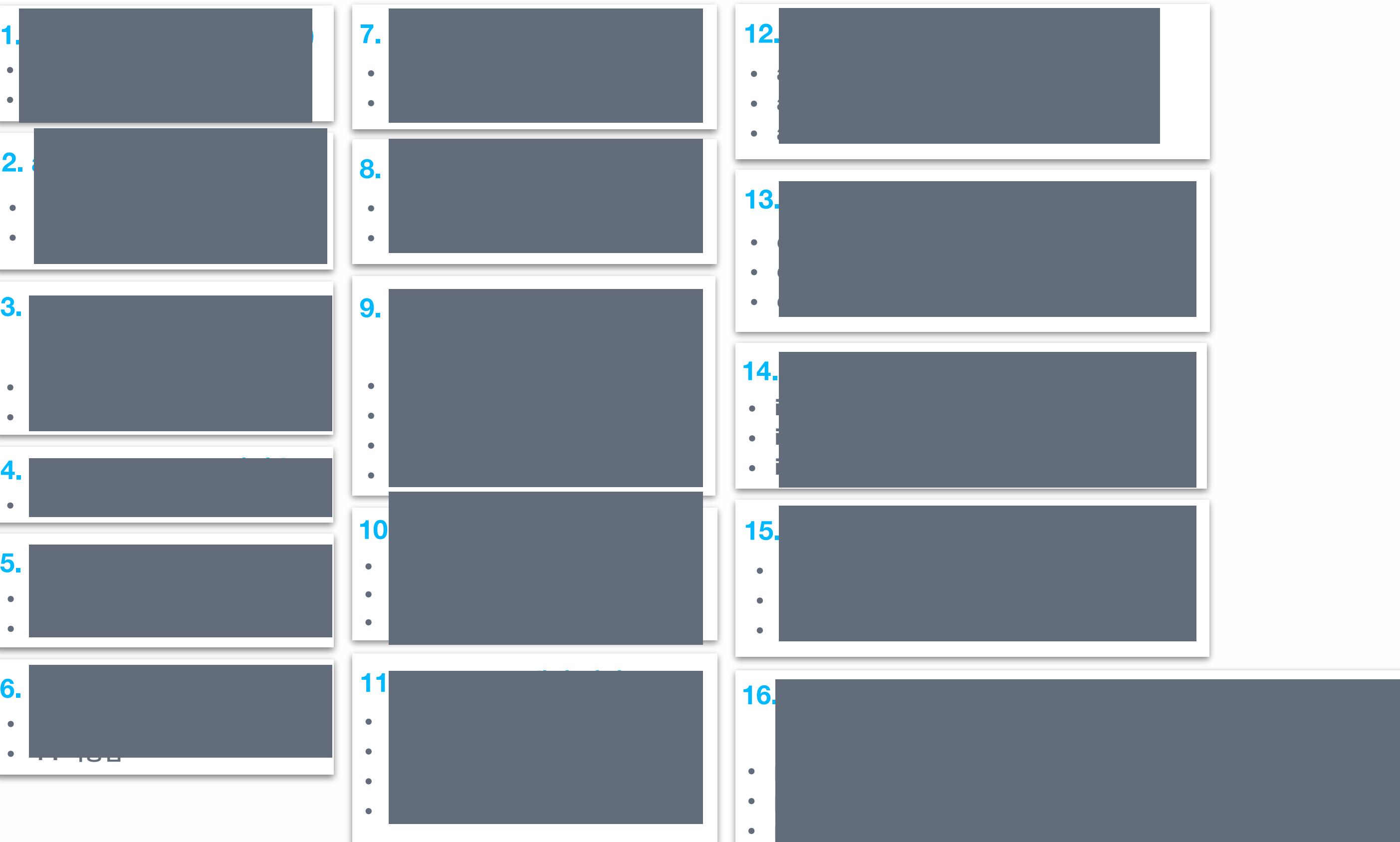
• Conclusion

1	0	0	0	0	0	1	0	0	0	1	0	0
1	0	0	0	0	0	0	1	0	0	0	1	0
0	0	1	0	0	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	0	0	0	1	0	0
1	0	0	0	0	0	1	0	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
1	0	0	0	0	0	1	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	1	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0

# 프로젝트 진행

## Feature Selection - Model ①

• Data Set



• EDA

• Preprocessing

• Model Test

• Conclusion

# 프로젝트 진행

## Feature Selection - Model ①

• Data Set



• EDA

• Preprocessing

• Model Test

• Conclusion

- Data Set
- EDA
- Preprocessing
- Model Test
- Conclusion

# Model ②

# 프로젝트 진행

## 데이터 전처리 - Model ② - EDA 기반 noise 판단된 데이터 부분 제거

### • Data Set

- non-fraud 전체 data의 [redacted] 데이터 개수 : 491개
- fraud 전체 data의 [redacted] 데이터 개수 : 0개

### • EDA

```
socar_all = socar_all[socar_all[
```

### • Preprocessing

- non-fraud 전체 data의 [redacted] 데이터 개수 : 584개
- non-fraud 전체 data의 [redacted] 데이터 개수 : 249개
- fraud 전체 data의 [redacted] 데이터 개수 : 0개
- fraud 전체 data의 [redacted] 데이터 개수 : 0개

### • Model Test

```
socar_all = socar_all[socar_all[  
socar_all = socar_all[socar_all[
```

### • Conclusion

- non-fraud 전체 data의 [redacted] 데이터 개수 : 827개
- fraud 전체 data의 [redacted] 데이터 개수 : 1개

```
socar_all = socar_all[socar_all[
```

# 프로젝트 진행

## 데이터 전처리 - Model ② - EDA 기반 noise 판단된 데이터 부분 제거

### • Data Set

- non-fraud 전체 data의 데이터 개수 : 73개
- non-fraud 전체 data의 데이터 개수 : 21개
- fraud 전체 data의 데이터 개수 : 0개
- fraud 전체 data의 데이터 개수 : 0개

```
socar_all = socar_all[socar_all[
```

### • EDA

## • Preprocessing

- 0 : '0' 이외의 값의 평균값으로 변환

```
socar_all = socar_all[socar_all[
```

### • Model Test

- 0 : '0' 이외의 값의 중간값으로 변환

```
socar_all = socar_all[socar_all[
```

### • Conclusion

# 프로젝트 진행

## 데이터 전처리 - Model ② - 연속형 데이터 명목형으로 변환

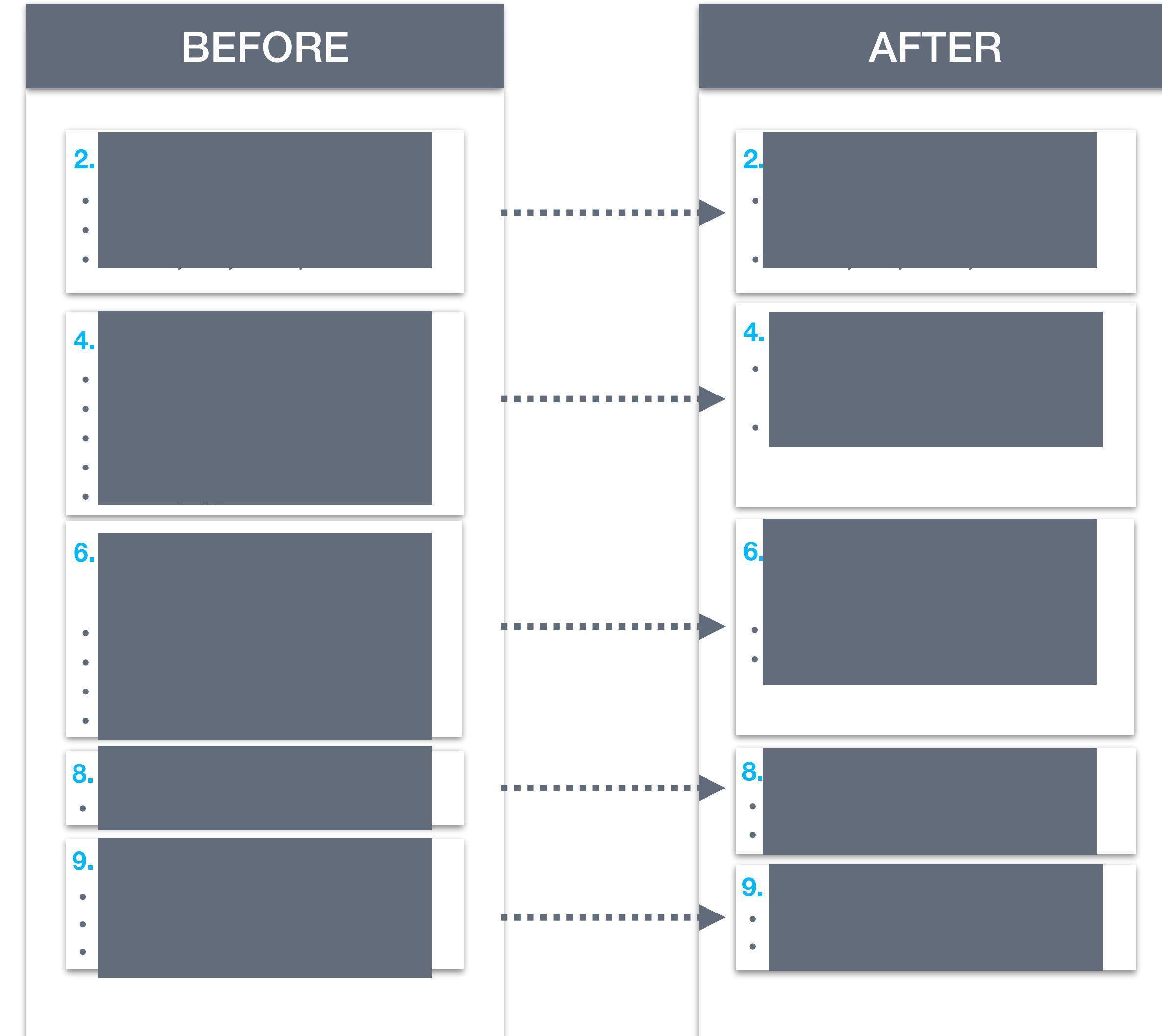
• Data Set

• EDA

• Preprocessing

• Model Test

• Conclusion



# 프로젝트 진행

## 데이터 전처리 - Model ② - 연속형 데이터 명목형으로 변환

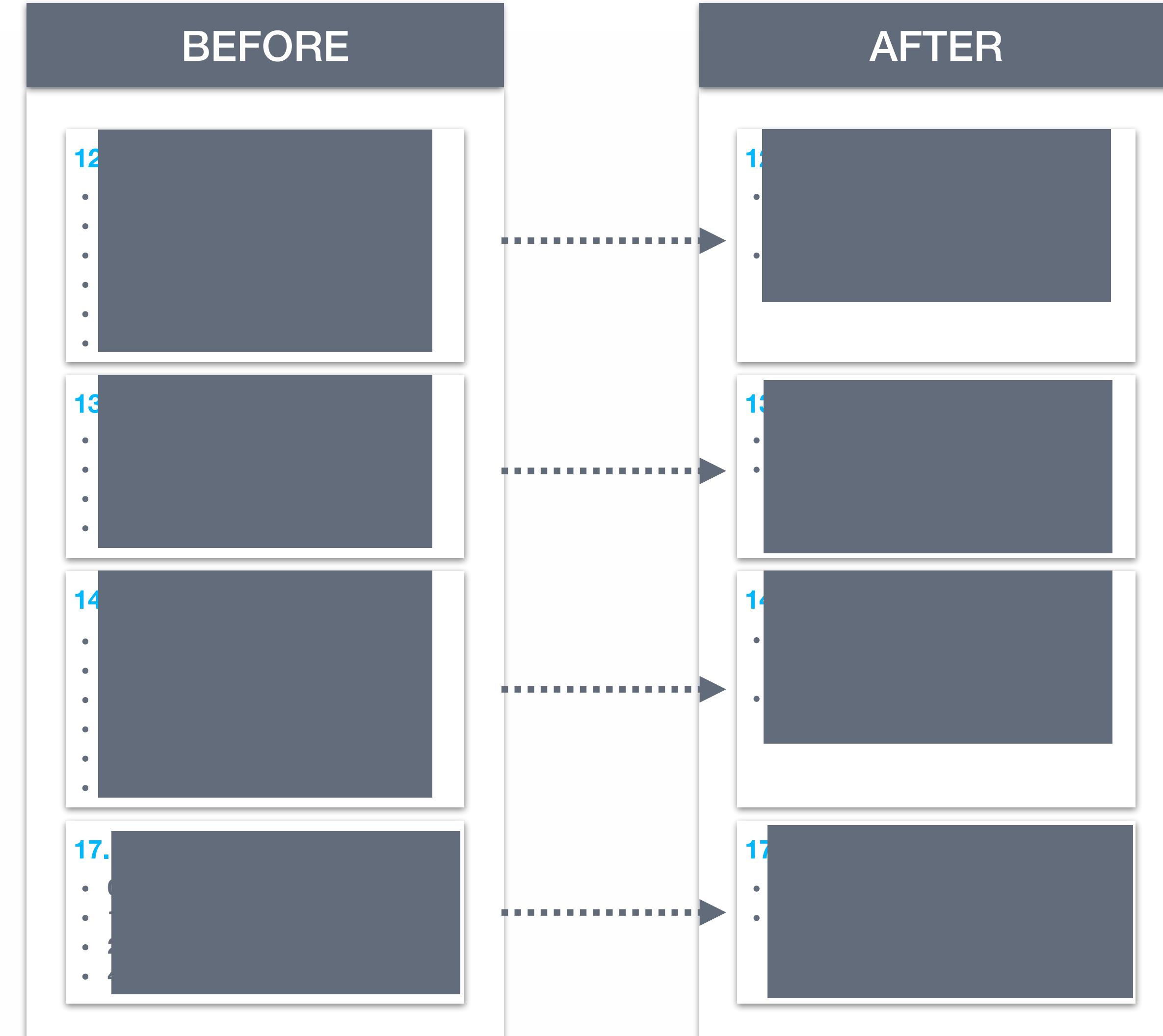
• Data Set

• EDA

• Preprocessing

• Model Test

• Conclusion



# 프로젝트 진행

## 데이터 전처리 - Model ② - 연속형 데이터 명목형으로 변환

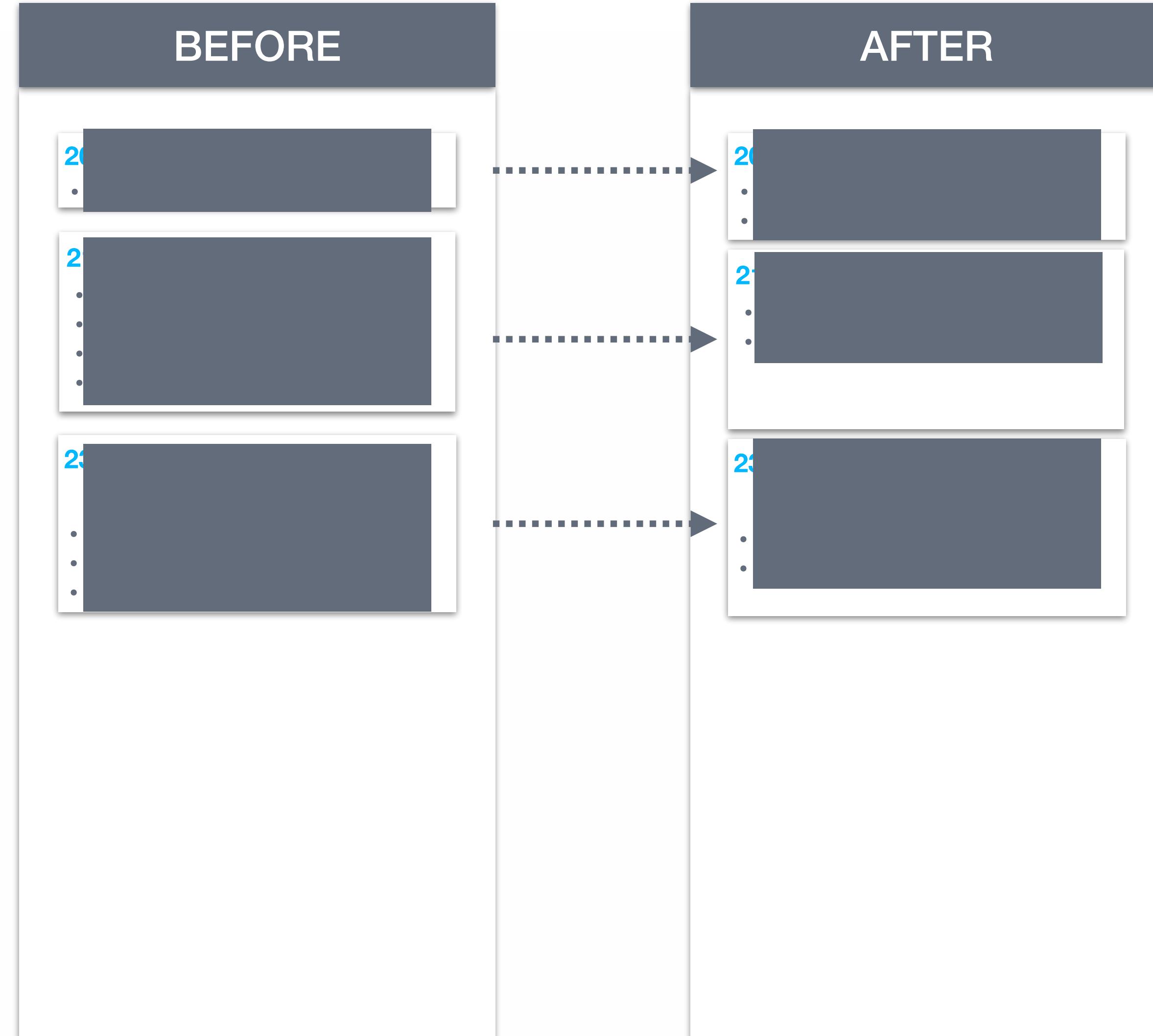
• Data Set

• EDA

• Preprocessing

• Model Test

• Conclusion



# 프로젝트 진행

## 데이터 전처리 - Model ② - Scaling 적용

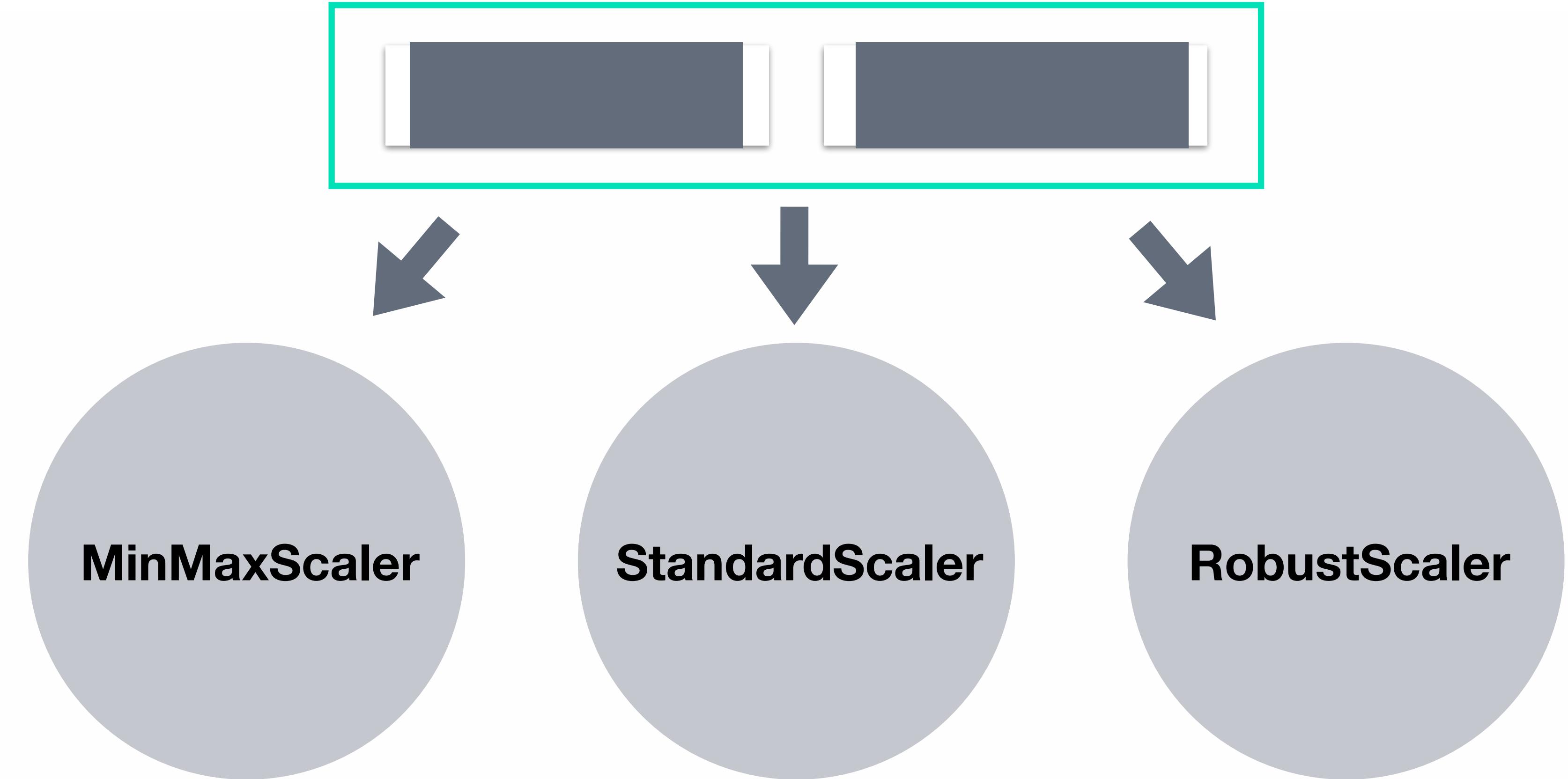
• Data Set

• EDA

• Preprocessing

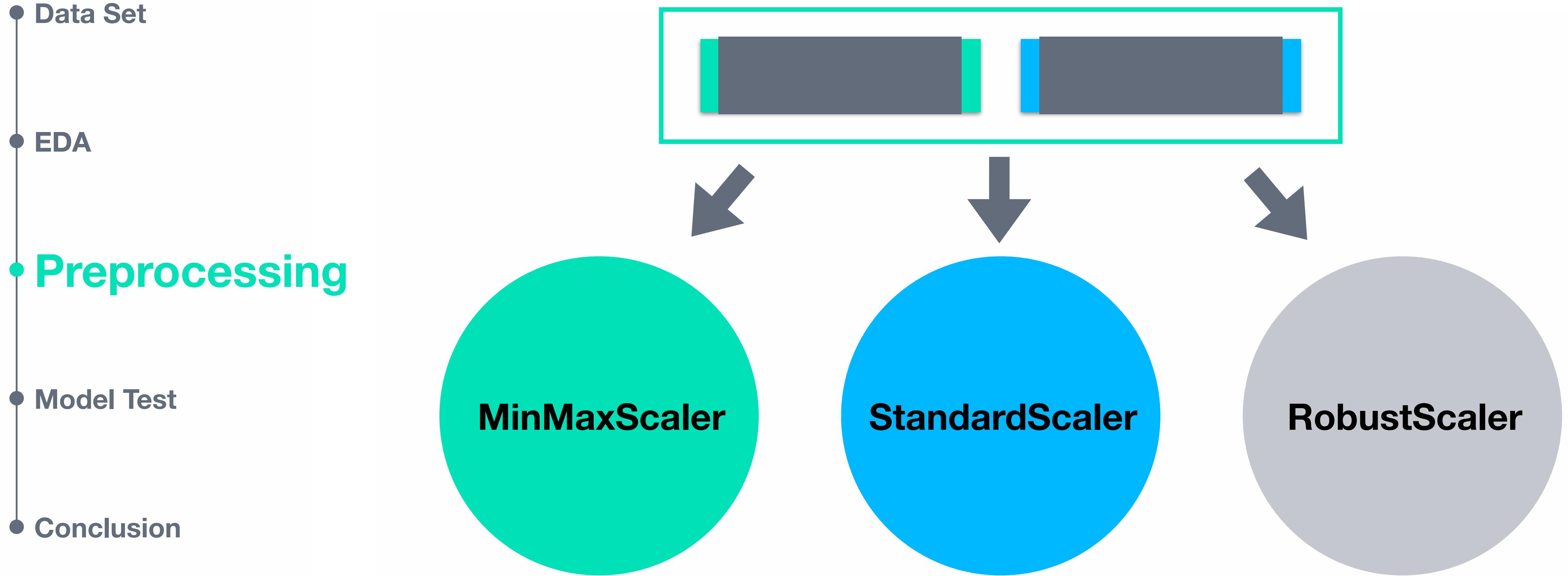
• Model Test

• Conclusion



# 프로젝트 진행

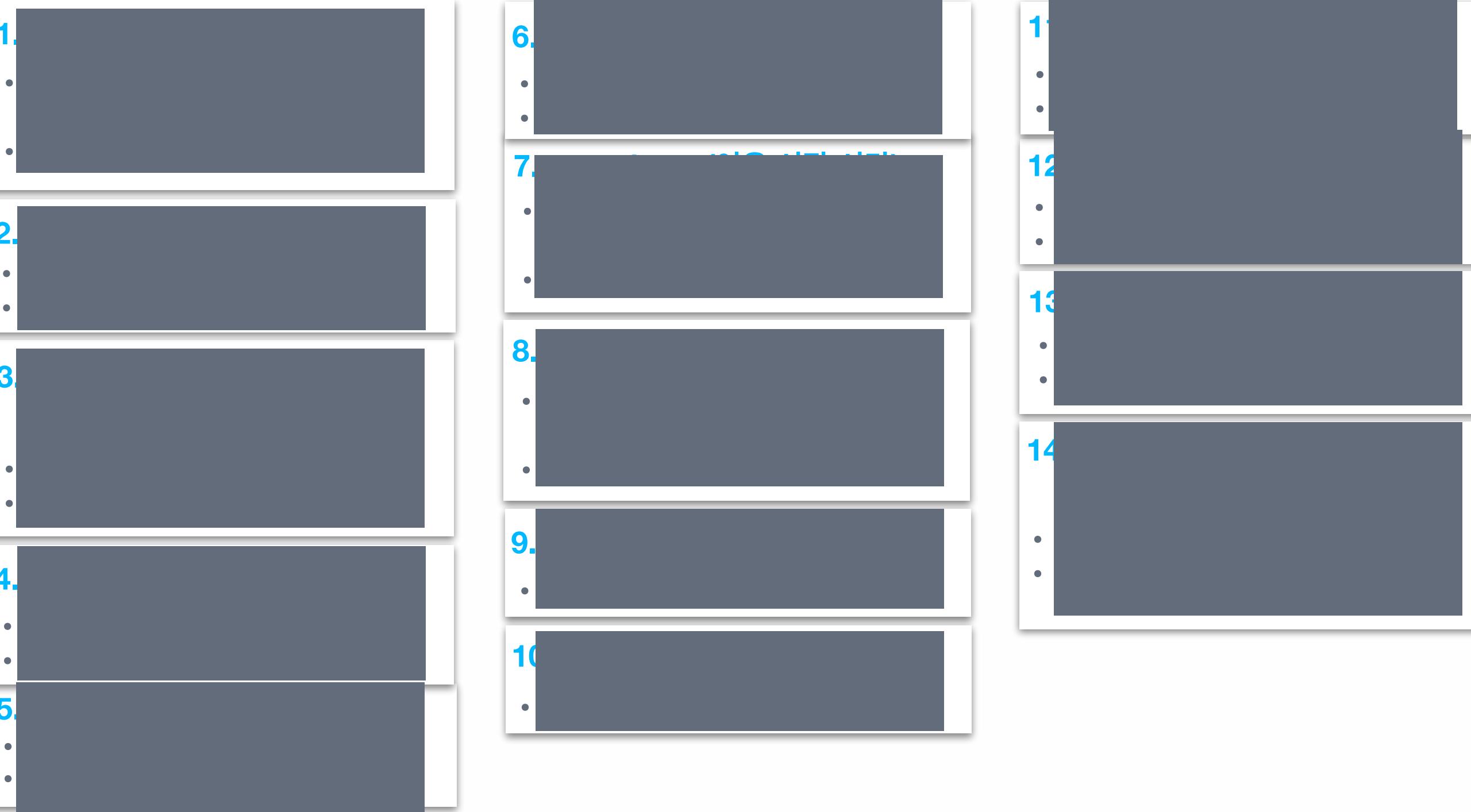
## 데이터 전처리 - Model ② - Scaling 적용



# 프로젝트 진행

## Feature Selection - Model ②

• Data Set



• EDA

• Preprocessing

• Model Test

• Conclusion

# 프로젝트 진행

## Feature Selection - Model ②

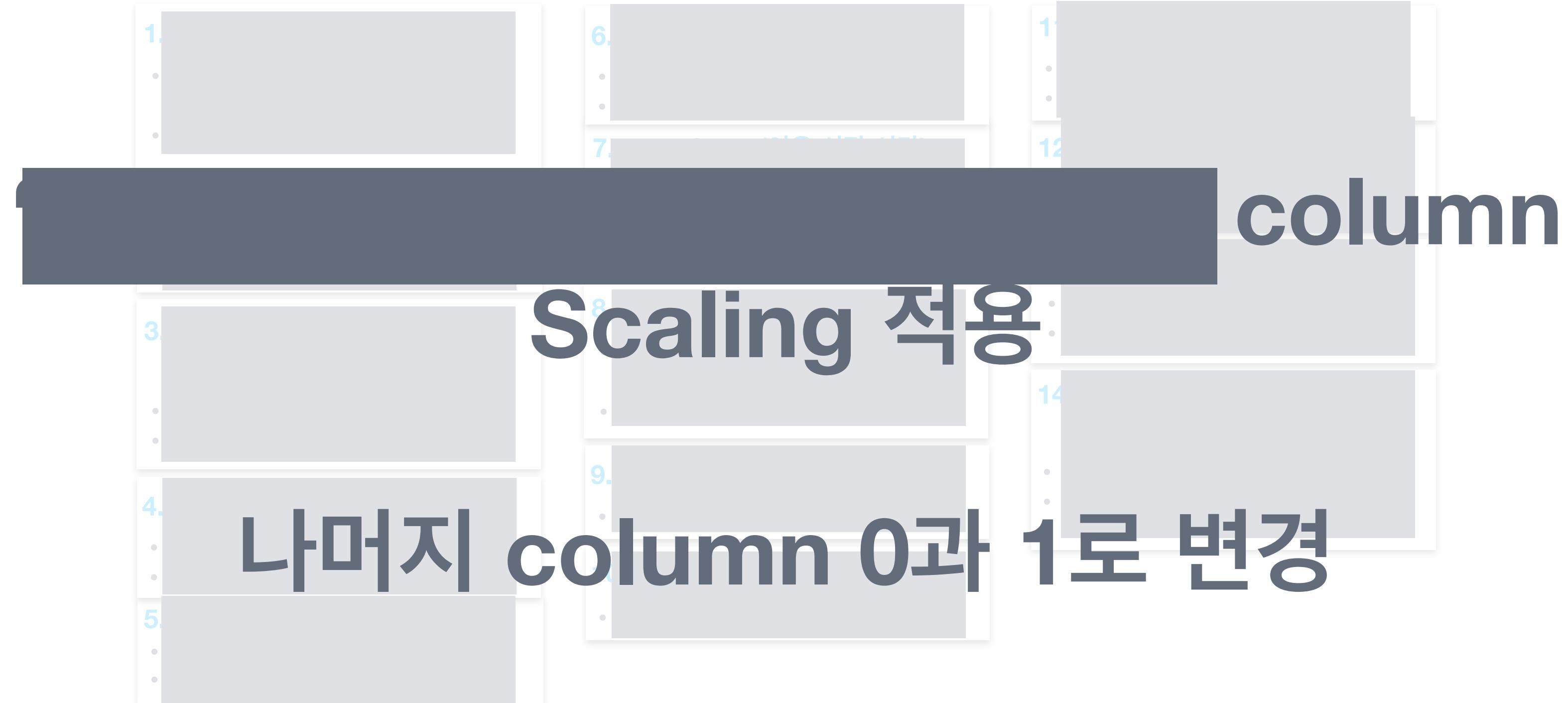
• Data Set

• EDA

• Preprocessing

• Model Test

• Conclusion



- Data Set
- EDA
- Preprocessing
- **Model Test**
- Conclusion

# Model ①

## Model Test - Model ①

### • Data Set

### • EDA

### • Preprocessing

### • Model Test

### • Conclusion



# 프로젝트 진행

## Model Test - Model ① - Under Sampling & Over Sampling

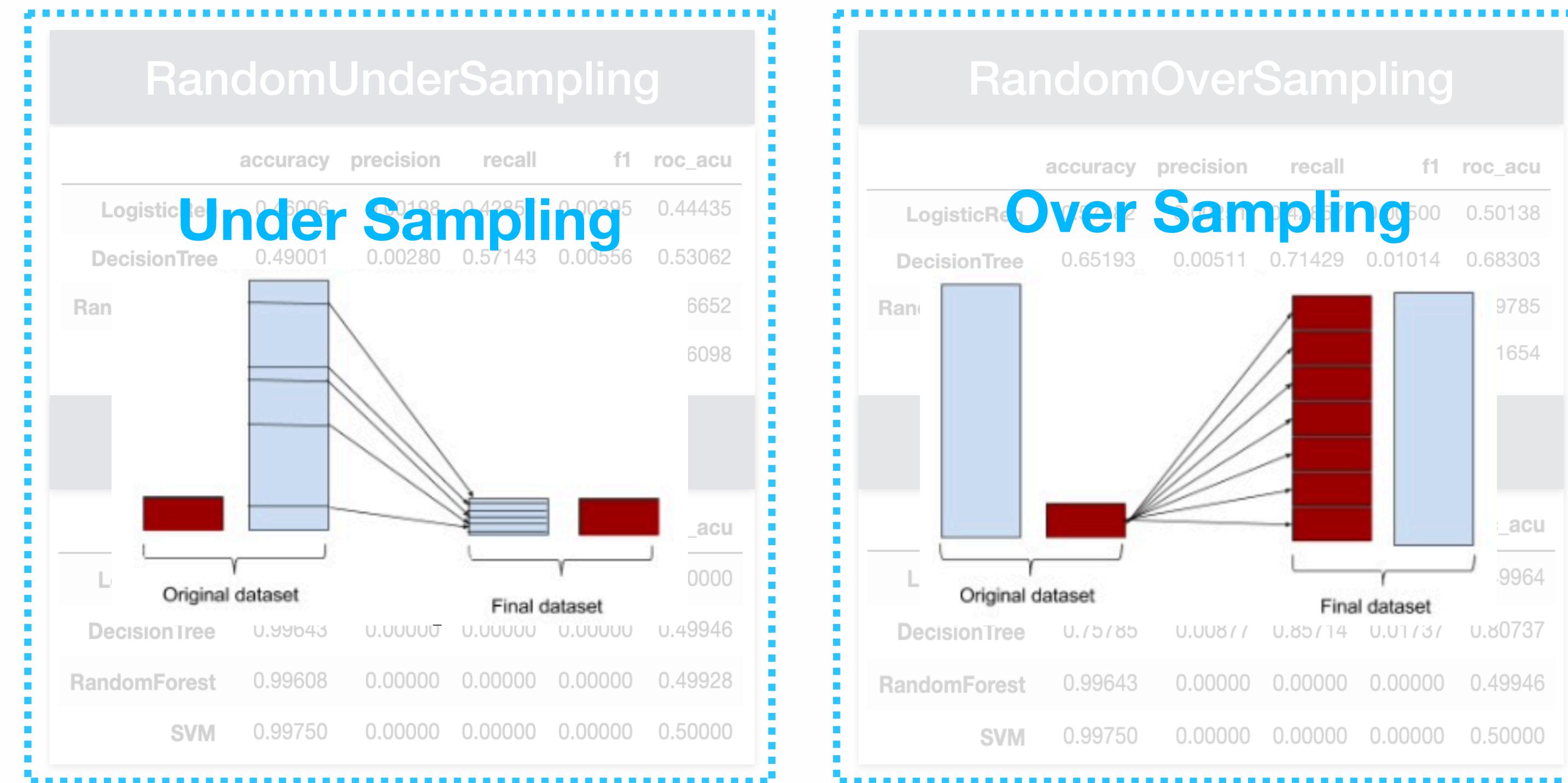
- Data Set

- EDA

- Preprocessing

- Model Test

- Conclusion



# 프로젝트 진행

## Model Test - Model ① - Under Sampling & Over Sampling

### • Data Set

### • EDA

### • Preprocessing

### • Model Test

### • Conclusion

RandomUnderSampling					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.46006	0.00198	0.42857	0.00395	0.44435
DecisionTree	0.49001	0.00280	0.57143	0.00556	0.53062
RandomForest	0.50428	0.00216	0.42857	0.00430	0.46652
SVM	0.49322	0.00211	0.42857	0.00420	0.46098

RandomOverSampling					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.57382	0.00251	0.42857	0.00500	0.50138
DecisionTree	0.65193	0.00511	0.71429	0.01014	0.68303
RandomForest	0.99322	0.00000	0.00000	0.00000	0.49785
SVM	0.54672	0.00158	0.28571	0.00314	0.41654

ENN					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.99750	0.00000	0.00000	0.00000	0.50000
DecisionTree	0.99643	0.00000	0.00000	0.00000	0.49946
RandomForest	0.99608	0.00000	0.00000	0.00000	0.49928
SVM	0.99750	0.00000	0.00000	0.00000	0.50000

SMOTE ✓					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.99679	0.00000	0.00000	0.00000	0.49964
DecisionTree	0.75785	0.00877	0.85714	0.01737	0.80737
RandomForest	0.99643	0.00000	0.00000	0.00000	0.49946
SVM	0.99750	0.00000	0.00000	0.00000	0.50000

### Parameter

- Logistic Regression : random\_state=13, solver='liblinear'
- Decision Tree : random\_state=13, max\_depth=6
- Random Forest : random\_state=13, n\_estimators=100
- Support Vector Machine : kernel='linear', gamma='auto', C=9

# 프로젝트 진행

## Model Test - Model ① - Combined Sampling

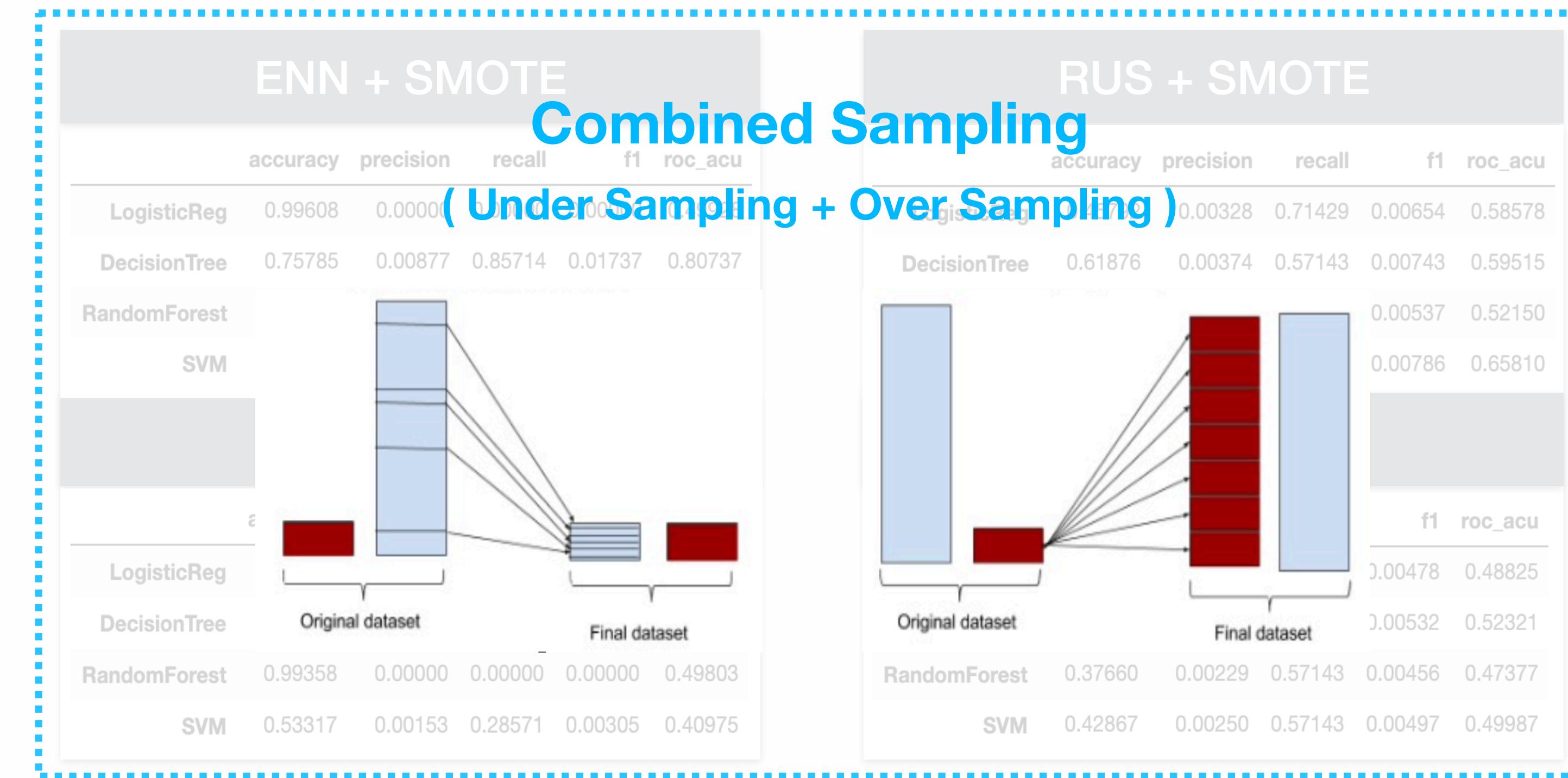
- Data Set

- EDA

- Preprocessing

- Model Test

- Conclusion



# 프로젝트 진행

## Model Test - Model ① - Combined Sampling

### • Data Set

### • EDA

### • Preprocessing

### • Model Test

### • Conclusion

ENN + SMOTE ✓					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.99608	0.00000	0.00000	0.00000	0.49928
DecisionTree	0.75785	0.00877	0.85714	0.01737	0.80737
RandomForest	0.99536	0.00000	0.00000	0.00000	0.49893
SVM	0.99750	0.00000	0.00000	0.00000	0.50000

ENN + ROS					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.57204	0.00250	0.42857	0.00498	0.50049
DecisionTree	0.73573	0.00271	0.28571	0.00537	0.51129
RandomForest	0.99358	0.00000	0.00000	0.00000	0.49803
SVM	0.53317	0.00153	0.28571	0.00305	0.40975

RUS + SMOTE					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.45792	0.00328	0.71429	0.00654	0.58578
DecisionTree	0.61876	0.00374	0.57143	0.00743	0.59515
RandomForest	0.47183	0.00270	0.57143	0.00537	0.52150
SVM	0.46006	0.00395	0.85714	0.00786	0.65810

RUS + ROS					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.40549	0.00240	0.57143	0.00478	0.48825
DecisionTree	0.33310	0.00267	0.71429	0.00532	0.52321
RandomForest	0.37660	0.00229	0.57143	0.00456	0.47377
SVM	0.42867	0.00250	0.57143	0.00497	0.49987

### Parameter

- Logistic Regression : random\_state=13, solver='liblinear'
- Decision Tree : random\_state=13, max\_depth=6
- Random Forest : random\_state=13, n\_estimators=100
- Support Vector Machine : kernel='linear', gamma='auto', C=9

# 프로젝트 진행

## Model Test - Model ① - Combined Sampling

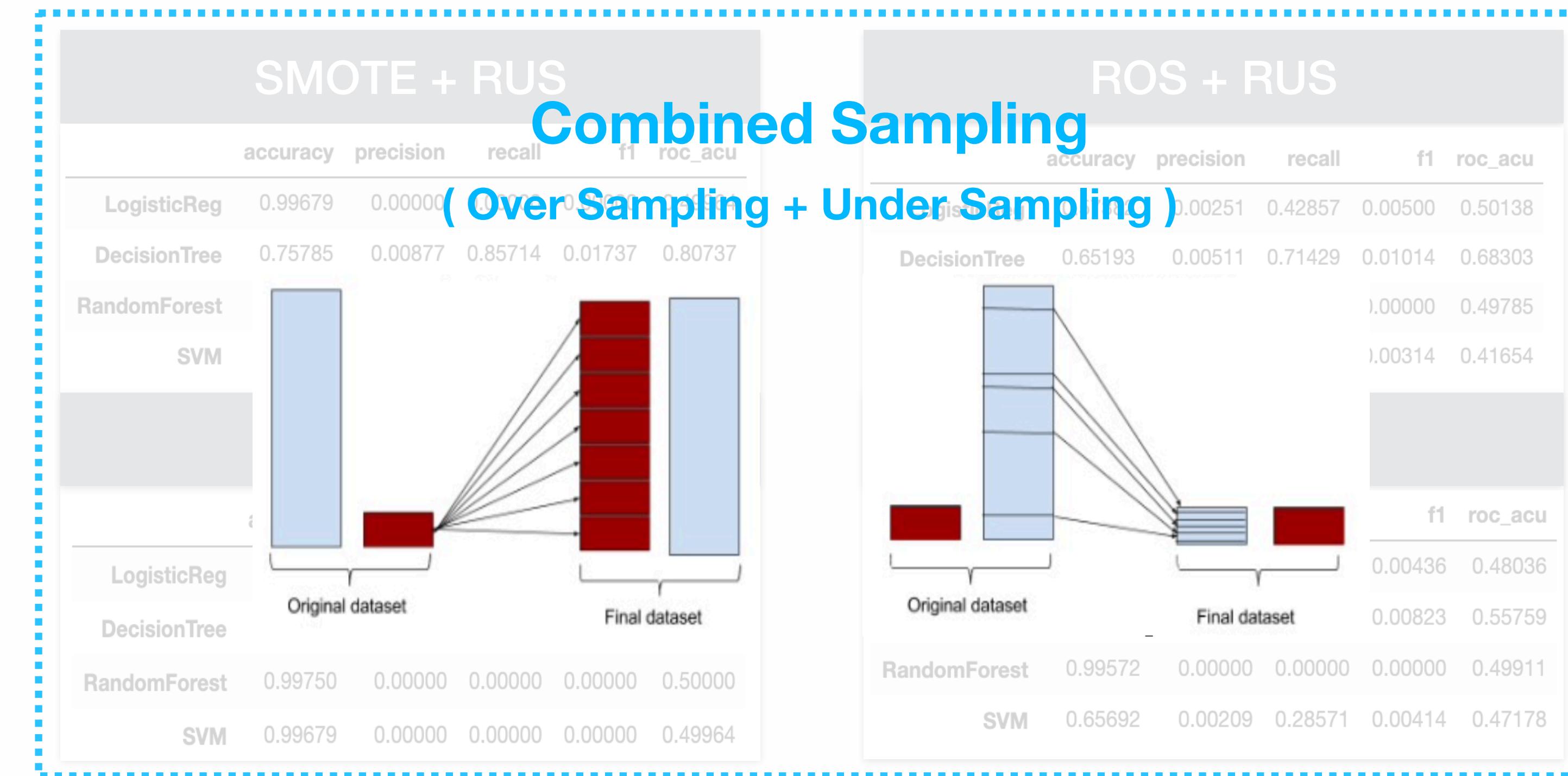
- Data Set

- EDA

- Preprocessing

- Model Test

- Conclusion



# 프로젝트 진행

## Model Test - Model ① - Combined Sampling

### • Data Set

### • EDA

### • Preprocessing

### • Model Test

### • Conclusion

SMOTE + RUS					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.99679	0.00000	0.00000	0.00000	0.49964
DecisionTree	0.75785	0.00877	0.85714	0.01737	0.80737
RandomForest	0.99715	0.00000	0.00000	0.00000	0.49982
SVM	0.99750	0.00000	0.00000	0.00000	0.50000

ROS + RUS					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.57382	0.00251	0.42857	0.00500	0.50138
DecisionTree	0.65193	0.00511	0.71429	0.01014	0.68303
RandomForest	0.99322	0.00000	0.00000	0.00000	0.49785
SVM	0.54672	0.00158	0.28571	0.00314	0.41654

SMOTE + ENN ✓					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.99750	0.00000	0.00000	0.00000	0.50000
DecisionTree	0.77389	0.00939	0.85714	0.01858	0.81541
RandomForest	0.99750	0.00000	0.00000	0.00000	0.50000
SVM	0.99679	0.00000	0.00000	0.00000	0.49964

### ROS + ENN

ROS + ENN					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.67404	0.00220	0.28571	0.00436	0.48036
DecisionTree	0.82810	0.00418	0.28571	0.00823	0.55759
RandomForest	0.99572	0.00000	0.00000	0.00000	0.49911
SVM	0.65692	0.00209	0.28571	0.00414	0.47178

### Parameter

- Logistic Regression : random\_state=13, solver='liblinear'
- Decision Tree : random\_state=13, max\_depth=6
- Random Forest : random\_state=13, n\_estimators=100
- Support Vector Machine : kernel='linear', gamma='auto', C=9

# 프로젝트 진행

## Model Test - Model ① - Combined Sampling

### • Data Set

### • EDA

### • Preprocessing

### • Model Test

### • Conclusion

SMOTE + ENN ✓					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.99750	0.00000	0.00000	0.00000	0.50000
DecisionTree	0.77389	0.00939	0.85714	0.01858	0.81541
RandomForest	0.99750	0.00000	0.00000	0.00000	0.50000
SVM	0.99679	0.00000	0.00000	0.00000	0.49964

SMOTE + RUS					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.99679	0.00000	0.00000	0.00000	0.49964
DecisionTree	0.75785	0.00877	0.85714	0.01737	0.80737
RandomForest	0.99715	0.00000	0.00000	0.00000	0.49982
SVM	0.99750	0.00000	0.00000	0.00000	0.50000

ROS + RUS					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.57382	0.00251	0.42857	0.00500	0.50138
DecisionTree	0.65193	0.00511	0.71429	0.01014	0.68303
RandomForest	0.99322	0.00000	0.00000	0.00000	0.49785
SVM	0.54672	0.00158	0.28571	0.00314	0.41654

ROS + ENN					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.67404	0.00220	0.28571	0.00436	0.48036
DecisionTree	0.82810	0.00418	0.28571	0.00823	0.55759
RandomForest	0.99572	0.00000	0.00000	0.00000	0.49911
SVM	0.65692	0.00209	0.28571	0.00414	0.47178

## Parameter

- Logistic Regression : random\_state=13, solver='liblinear'
- Decision Tree : random\_state=13, max\_depth=6
- Random Forest : random\_state=13, n\_estimators=100
- Support Vector Machine : kernel='linear', gamma='auto', C=9

# 프로젝트 진행

## Model Test - Model ① - 목표 달성 평가

- Data Set

목표 성능

model name	train accuracy	train precision	train recall	test accuracy	test precision	test recall
0 ?	0.9	0.9	0.9	0.8	0.8	0.8

- EDA

- Preprocessing

- Model Test

- Conclusion

Train fraud data => 31 / 34      Test fraud data => 6 / 7

성능 결과

model name	train accuracy	train precision	train recall	test accuracy	test precision	test recall
0 Decision Tree	0.825259	0.008611	0.515151	0.77389	0.00939	0.81541

Train fraud data => 17 / 33

Test fraud data => 6 / 7

- Data Set
- EDA
- Preprocessing
- **Model Test**
- Conclusion

# Model ②

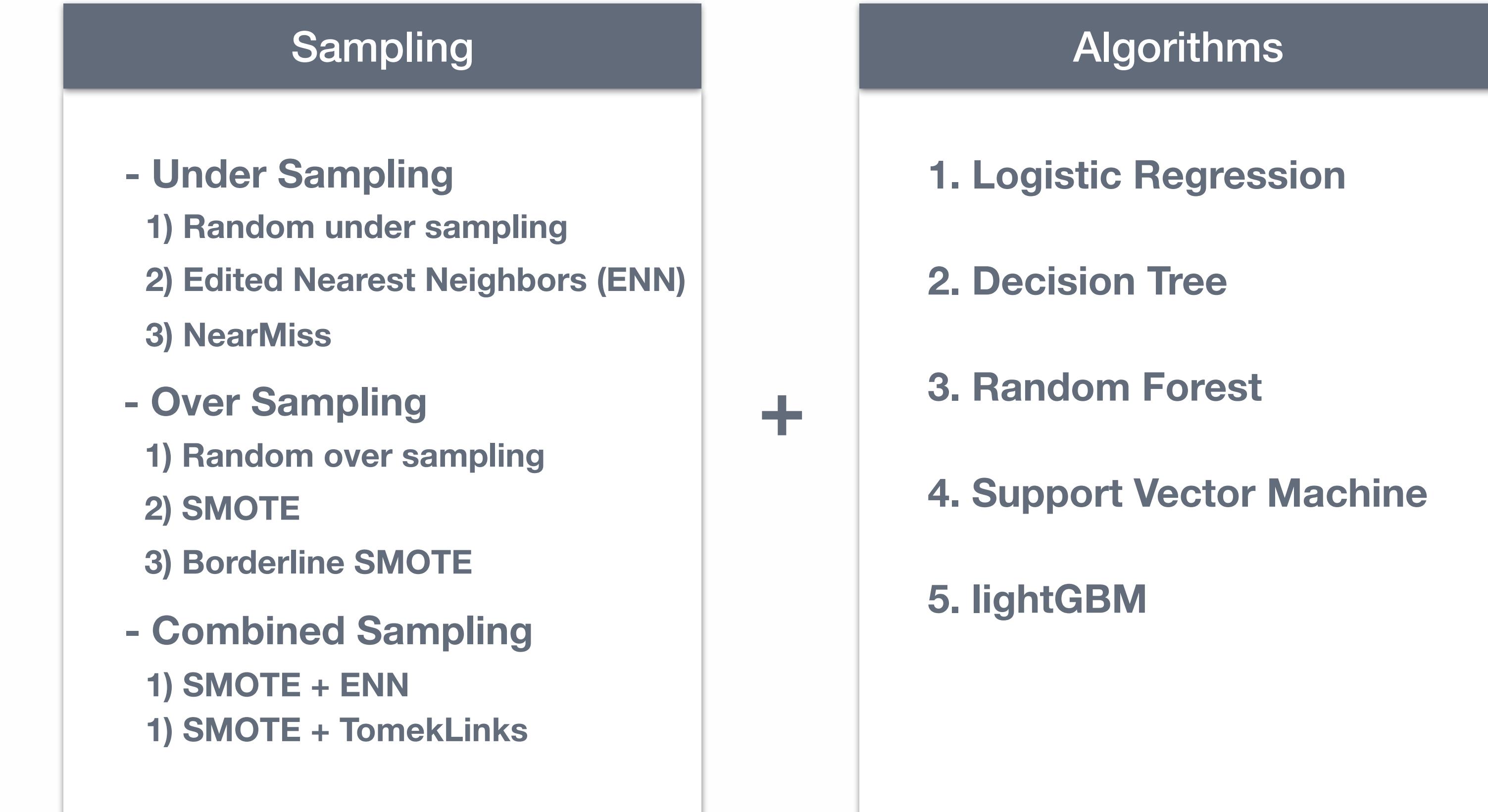
- Data Set

- EDA

- Preprocessing

- **Model Test**

- Conclusion



# 프로젝트 진행

## Model Test - Model ② - Under Sampling & Over Sampling

### • Data Set

RandomUnderSampling					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.41548	0.00277	0.57143	0.00552	0.49323
DecisionTree	0.48034	0.00234	0.42857	0.00466	0.45453
RandomForest	0.41103	0.00275	0.57143	0.00548	0.49100
LightGBM	0.56019	0.00550	0.85714	0.0109	
SVM	0.47872	0.00311	0.57143	0.00618	0.52494
LinearSVM	0.49574	0.00241	0.42857	0.00480	0.46225

ADASYN					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.80422	0.01029	0.71429	0.02028	0.75938
DecisionTree	0.63924	0.00560	0.71429	0.01111	0.67666
RandomForest	0.84759	0.00800	0.42857	0.01571	0.63868
LightGBM	0.90109	0.00418	0.14286	0.00813	0.52305
SVM	0.81597	0.01094	0.71429	0.02155	0.76527
LinearSVM	0.75517	0.00824	0.71429	0.01629	0.73479

ENN					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.76490	0.00858	0.71429	0.01695	0.73966
DecisionTree	0.80030	0.00569	0.71429	0.01019	0.66019
RandomForest	0.85488	0.01114	0.57143	0.02186	0.71356
LightGBM	0.81558	0.01369	0.28571	0.01307	0.58249
SVM	0.77098	0.00880	0.71429	0.01739	0.74271
LinearSVM	0.62505	0.00539	0.71429	0.01070	0.66954

SMOTE					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.76490	0.00858	0.71429	0.01695	0.73966
DecisionTree	0.80030	0.00569	0.71429	0.01019	0.66019
RandomForest	0.85488	0.01114	0.57143	0.02186	0.71356
LightGBM	0.81558	0.01369	0.28571	0.01307	0.58249
SVM	0.77098	0.00880	0.71429	0.01739	0.74271
LinearSVM	0.62505	0.00539	0.71429	0.01070	0.66954

NearMiss					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.10093	0.00315	1.00000	0.00627	0.54919
DecisionTree	0.11674	0.00320	1.00000	0.00638	0.55711
RandomForest	0.06567	0.00303	1.00000	0.00604	0.53150
LightGBM	0.26469	0.00384	1.00000	0.00766	0.63130
SVM	0.13417	0.00327	1.00000	0.00651	0.56585
LinearSVM	0.11755	0.00321	1.00000	0.00639	0.55752

Borderline SMOTE					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.85894	0.00292	0.14286	0.00571	0.50192
DecisionTree	0.38022	0.00391	0.85714	0.00779	0.61800
RandomForest	0.88772	0.00368	0.14286	0.00717	0.51635
LightGBM	0.91771	0.00505	0.14286	0.00976	0.53139
SVM	0.86380	0.00302	0.14286	0.00592	0.50436
LinearSVM	0.82529	0.00235	0.14286	0.00462	0.48505

Parameter

### • Preprocessing

- Logistic Regression : random\_state=13, solver='liblinear'
- Decision Tree : random\_state=13, max\_depth=3
- Random Forest : random\_state=13, n\_estimators=200, max\_depth=10
- lightGBM : num\_leaves=10, n\_estimators=100, boost\_from\_average=False
- SVC : kernel='linear'
- LinearSVC : C=200, random\_state=13

### • Conclusion

# 프로젝트 진행

## Model Test - Model ② - Under Sampling & Over Sampling

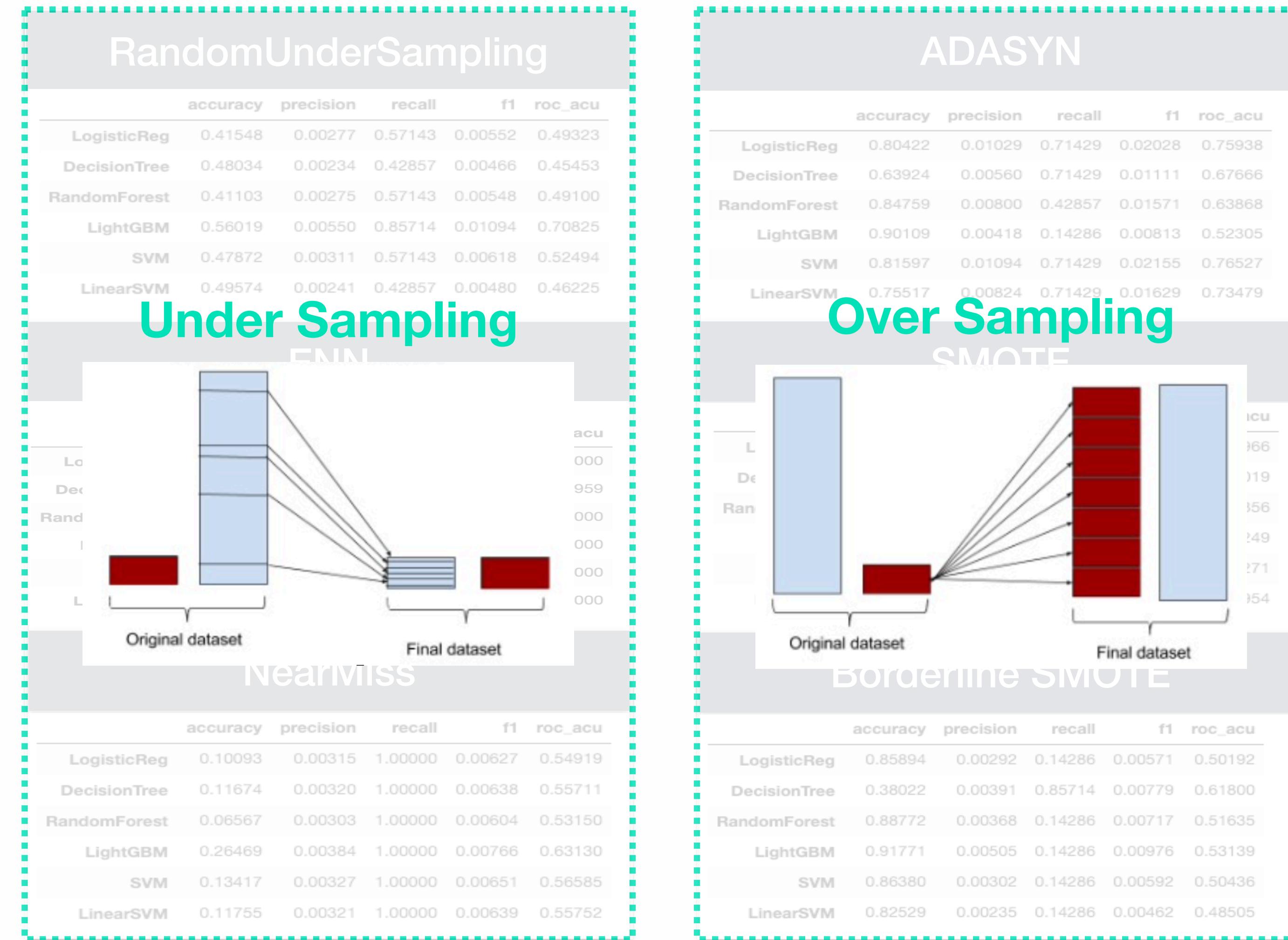
- Data Set

- EDA

- Preprocessing

- Model Test

- Conclusion



# 프로젝트 진행

## Model Test - Model ② - Under Sampling & Over Sampling

### • Data Set

#### RandomUnderSampling

	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.41548	0.00277	0.57143	0.00552	0.49323
DecisionTree	0.48034	0.00234	0.42857	0.00466	0.45453
RandomForest	0.41103	0.00275	0.57143	0.00548	0.49100
LightGBM	0.56019	0.00550	0.85714	0.01094	0.70825
SVM	0.47872	0.00311	0.57143	0.00618	0.52494
LinearSVM	0.49574	0.00241	0.42857	0.00480	0.46225

#### ADASYN ✓

	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.80422	0.01029	0.71429	0.02028	0.75938
DecisionTree	0.63924	0.00560	0.71429	0.01111	0.67666
RandomForest	0.84759	0.00800	0.42857	0.01571	0.63868
LightGBM	0.90109	0.00418	0.14286	0.00813	0.52305
SVM	0.81597	0.01094	0.71429	0.02155	0.76527
LinearSVM	0.75517	0.00824	0.71429	0.01629	0.73479

#### ENN

	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.99716	0.00000	0.00000	0.00000	0.50000
DecisionTree	0.99635	0.00000	0.00000	0.00000	0.49959
RandomForest	0.99716	0.00000	0.00000	0.00000	0.50000
LightGBM	0.99716	0.00000	0.00000	0.00000	0.50000
SVM	0.99716	0.00000	0.00000	0.00000	0.50000
LinearSVM	0.99716	0.00000	0.00000	0.00000	0.50000

#### NearMiss

	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.10093	0.00315	1.00000	0.00627	0.54919
DecisionTree	0.11674	0.00320	1.00000	0.00638	0.55711
RandomForest	0.06567	0.00303	1.00000	0.00604	0.53150
LightGBM	0.26469	0.00384	1.00000	0.00766	0.63130
SVM	0.13417	0.00327	1.00000	0.00651	0.56585
LinearSVM	0.11755	0.00321	1.00000	0.00639	0.55752

#### SMOTE

	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.76490	0.00858	0.71429	0.01695	0.73966
DecisionTree	0.60640	0.00513	0.71429	0.01019	0.66019
RandomForest	0.85488	0.01114	0.57143	0.02186	0.71356
LightGBM	0.87758	0.00669	0.28571	0.01307	0.58249
SVM	0.77098	0.00880	0.71429	0.01739	0.74271
LinearSVM	0.62505	0.00539	0.71429	0.01070	0.66954

#### Borderline SMOTE

	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.85894	0.00292	0.14286	0.00571	0.50192
DecisionTree	0.38022	0.00391	0.85714	0.00779	0.61800
RandomForest	0.88772	0.00368	0.14286	0.00717	0.51635
LightGBM	0.91771	0.00505	0.14286	0.00976	0.53139
SVM	0.86380	0.00302	0.14286	0.00592	0.50436
LinearSVM	0.82529	0.00235	0.14286	0.00462	0.48505

### • Model Test

### Conclusion

# 프로젝트 진행

## Model Test - Model ② - Combined Sampling

- Data Set



- EDA

- Preprocessing

- Model Test

- Conclusion

# 프로젝트 진행

## Model Test - Model ② - Combined Sampling

### • Data Set

ENN + SMOTE					
	accuracy	precision	recall	f1	roc_acu
<b>LogisticReg</b>	0.74990	0.00806	0.71429	0.01595	0.73214
<b>DecisionTree</b>	0.60640	0.00513	0.71429	0.01019	0.66019
<b>RandomForest</b>	0.84718	0.01058	0.57143	0.02078	0.70970
<b>LightGBM</b>	0.87880	0.00676	0.28571	0.01320	0.58310
<b>SVM</b>	0.74017	0.00776	0.71429	0.01536	0.72726
<b>LinearSVM</b>	0.85245	0.01096	0.57143	0.02151	0.71234

SMOTE + ENN ✓					
	accuracy	precision	recall	f1	roc_acu
<b>LogisticReg</b>	0.83056	0.01188	0.71429	0.02336	0.77259
<b>DecisionTree</b>	0.60397	0.00510	0.71429	0.01013	0.65897
<b>RandomForest</b>	0.91893	0.00000	0.00000	0.00000	0.46077
<b>LightGBM</b>	0.92987	0.00595	0.14286	0.01143	0.53749
<b>SVM</b>	0.84151	0.01269	0.71429	0.02494	0.77808
<b>LinearSVM</b>	0.86826	0.01524	0.71429	0.02985	0.79149

### • EDA

### • Preprocessing

### • Model Test

### • Conclusion

#### TomekLinks + SMOTE

	accuracy	precision	recall	f1	roc_acu
<b>LogisticReg</b>	0.75598	0.00826	0.71429	0.01634	0.73519
<b>DecisionTree</b>	0.60640	0.00513	0.71429	0.01019	0.66019
<b>RandomForest</b>	0.85367	0.01105	0.57143	0.02168	0.71295
<b>LightGBM</b>	0.88569	0.00717	0.28571	0.01399	0.58656
<b>SVM</b>	0.78516	0.00938	0.71429	0.01852	0.74983
<b>LinearSVM</b>	0.64613	0.00571	0.71429	0.01133	0.68011

#### SMOTE + TomekLinks

	accuracy	precision	recall	f1	roc_acu
<b>LogisticReg</b>	0.78192	0.00924	0.71429	0.01825	0.74820
<b>DecisionTree</b>	0.60640	0.00513	0.71429	0.01019	0.66019
<b>RandomForest</b>	0.85813	0.00860	0.42857	0.01685	0.64396
<b>LightGBM</b>	0.87272	0.00958	0.42857	0.01875	0.65128
<b>SVM</b>	0.78354	0.00931	0.71429	0.01838	0.74901
<b>LinearSVM</b>	0.68464	0.00640	0.71429	0.01269	0.69942

# 프로젝트 진행

## Model Test - Model ② - Combined Sampling

### • Data Set

ENN + SMOTE					
	accuracy	precision	recall	f1	roc_acu
<b>LogisticReg</b>	0.74990	0.00806	0.71429	0.01595	0.73214
<b>DecisionTree</b>	0.60640	0.00513	0.71429	0.01019	0.66019
<b>RandomForest</b>	0.84718	0.01058	0.57143	0.02078	0.70970
<b>LightGBM</b>	0.87880	0.00676	0.28571	0.01320	0.58310
<b>SVM</b>	0.74017	0.00776	0.71429	0.01536	0.72726
<b>LinearSVM</b>	0.85245	0.01096	0.57143	0.02151	0.71234

SMOTE + ENN					
	accuracy	precision	recall	f1	roc_acu
<b>LogisticReg</b>	0.83056	0.01188	0.71429	0.02336	0.77259
<b>DecisionTree</b>	0.60397	0.00510	0.71429	0.01013	0.65897
<b>RandomForest</b>	0.91893	0.00000	0.00000	0.00000	0.46077
<b>LightGBM</b>	0.92987	0.00595	0.14286	0.01143	0.53749
<b>SVM</b>	0.84151	0.01269	0.71429	0.02494	0.77808
<b>LinearSVM</b>	0.86826	0.01524	0.71429	0.02985	0.79149

### • EDA

### • Preprocessing

### • Model Test

### • Conclusion

TomekLinks + SMOTE ✓					
	accuracy	precision	recall	f1	roc_acu
<b>LogisticReg</b>	0.75598	0.00826	0.71429	0.01634	0.73519
<b>DecisionTree</b>	0.60640	0.00513	0.71429	0.01019	0.66019
<b>RandomForest</b>	0.85367	0.01105	0.57143	0.02168	0.71295
<b>LightGBM</b>	0.88569	0.00717	0.28571	0.01399	0.58656
<b>SVM</b>	0.78516	0.00938	0.71429	0.01852	0.74983
<b>LinearSVM</b>	0.64613	0.00571	0.71429	0.01133	0.68011

SMOTE + TomekLinks					
	accuracy	precision	recall	f1	roc_acu
<b>LogisticReg</b>	0.78192	0.00924	0.71429	0.01825	0.74820
<b>DecisionTree</b>	0.60640	0.00513	0.71429	0.01019	0.66019
<b>RandomForest</b>	0.85813	0.00860	0.42857	0.01685	0.64396
<b>LightGBM</b>	0.87272	0.00958	0.42857	0.01875	0.65128
<b>SVM</b>	0.78354	0.00931	0.71429	0.01838	0.74901
<b>LinearSVM</b>	0.68464	0.00640	0.71429	0.01269	0.69942

# 프로젝트 진행

## Model Test - Model ② - Combined Sampling

### • Data Set

ENN + SMOTE					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.74990	0.00806	0.71429	0.01595	0.73214
DecisionTree	0.60640	0.00513	0.71429	0.01019	0.66019
RandomForest	0.84718	0.01058	0.57143	0.02078	0.70970
LightGBM	0.87880	0.00676	0.28571	0.01320	0.58310
SVM	0.74017	0.00776	0.71429	0.01536	0.72726
LinearSVM	0.85245	0.01096	0.57143	0.02151	0.71234

SMOTE + ENN					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.83056	0.01188	0.71429	0.02336	0.77259
DecisionTree	0.60397	0.00510	0.71429	0.01013	0.65897
RandomForest	0.91893	0.00000	0.00000	0.00000	0.46077
LightGBM	0.92987	0.00595	0.14286	0.01143	0.53749
SVM	0.84151	0.01269	0.71429	0.02494	0.77808
LinearSVM	0.86826	0.01524	0.71429	0.02985	0.79149

TomekLinks + SMOTE ✓					
	accuracy	precision	recall	f1	roc_acu
LogisticReg	0.75598	0.00826	0.71429	0.01634	0.73519
DecisionTree	0.60640	0.00513	0.71429	0.01019	0.66019
RandomForest	0.85367	0.01105	0.57143	0.02168	0.71295
LightGBM	0.88569	0.00717	0.28571	0.01399	0.58656
SVM	0.78516	0.00938	0.71429	0.01852	0.74983
LinearSVM	0.64613	0.00571	0.71429	0.01133	0.68011

Best Model!

### • Model Test

### • Conclusion

# 프로젝트 진행

## Model Test - Model ② - 목표 달성 평가

- Data Set

목표 성능

model name	train accuracy	train precision	train recall	test accuracy	test precision	test recall
0 ?	0.9	0.9	0.9	0.8	0.8	0.8

- EDA

- Preprocessing

- Model Test

- Conclusion

Train fraud data => 31 / 34

Test fraud data => 6 / 7

성능 결과

model name	train accuracy	train precision	train recall	test accuracy	test precision	test recall
0 RandomForest	0.886515	0.022341	0.757575	0.85367	0.01105	0.57143

↓ ↓ ↓ ↑ ↓ ↓

Train fraud data => 25 / 33

Test fraud data => 4 / 7

# 프로젝트 진행

## 요약 - Model ①

### • Data Set

### • EDA

### • Preprocessing

### • Model Test

### • Conclusion

## Model ①

	model name	train accuracy	train precision	train recall	test accuracy	test precision	test recall
0	Decision Tree	0.825259	0.008611	0.515151	0.77389	0.00939	0.81541

Train fraud data => 17 / 33      Test fraud data => 6 / 7

- Data 비율 : 0.85 : 0.15
- Feature Selection : 도메인 지식 + EDA 기반
- 데이터 전처리 : ① 불필요 판단 컬럼 제거 ② noise 판단 데이터 부분 제거 ③ 명목형 변환  
④ OneHotEncoding
- Parameter Tuning :
  - Logistic Regression : random\_state=13, solver='liblinear'
  - Decision Tree : random\_state=13, max\_depth=6
  - Random Forest : random\_state=13, n\_estimators=100
  - Support Vector Machine : kernel='linear', gamma='auto', C=9

# 프로젝트 진행

## 요약 - Model ②

### • Data Set

### • EDA

### • Preprocessing

### • Model Test

### • Conclusion

## Model ②

	model name	train accuracy	train precision	train recall	test accuracy	test precision	test recall
0	RandomForest	0.886515	0.022341	0.757575	0.84272	0.01028	0.57143

Train fraud data => 25 / 33      Test fraud data => 4 / 7

- Data 비율 : 0.85 : 0.15
- Feature Selection : EDA + 무작위 추출 기반
- 데이터 전처리 : ① 무작위 컬럼 제거 ② noise 판단 데이터 부분 제거 ③ Scaler 적용  
④ 명목형 변환
- Parameter Tuning :
  - Logistic Regression : random\_state=13, solver='liblinear'
  - Decision Tree : random\_state=13, max\_depth=3
  - Random Forest : random\_state=13, n\_estimators=200, max\_depth=10
  - lightGBM : num\_leaves=10, n\_estimators=100, boost\_from\_average=False
  - SVC : kernel='linear'
  - LinearSVC : C=200, random\_state=13

- Data Set
  - 1. 새로운 Test data 부재로 인한 더욱 정확한 모델 성능 판단 한계
- EDA
- Preprocessing
  - 2. 결과 모델이 해당 fraud train data와 test data에만 적합할 가능성 존재
  - 3. 구간이 나눠진 컬럼들의 기준 파악 어려움으로 인한 더 다양한 feature engineering 시도 한계
- Model Test
- Conclusion
  - 4. 다수의 ‘알 수 없음’이 포함된 여러 컬럼들로 인한 데이터 전처리 제거 기준 정의 어려움

- Data Set

1. Imbalanced data에 대한 다양한 Sampling 기법

- EDA

2. Feature selection 기준에 대한 다양한 접근법

- Preprocessing

3. 다양한 모델링 기법에 대한 경험적 지식

- Model Test

4. Imbalanced Data Set 모델 성능 지표 해석 및 평가

- Conclusion

5. 비대면 업무 시 의사소통 방법

# 에필로그

- **About SCUT**

- **SCUT Algorithm**

- **Best Model**

- **Reference**

## SCUT algorithm

- : Multi-Class Imbalanced Data Classification using SMOTE and Cluster-based Undersampling Technic)
- : 비지도 학습인 군집 모델링을 바탕으로 샘플링하는 방법

기존

2개의 클래스



K-means Clustering

4개의 클래스



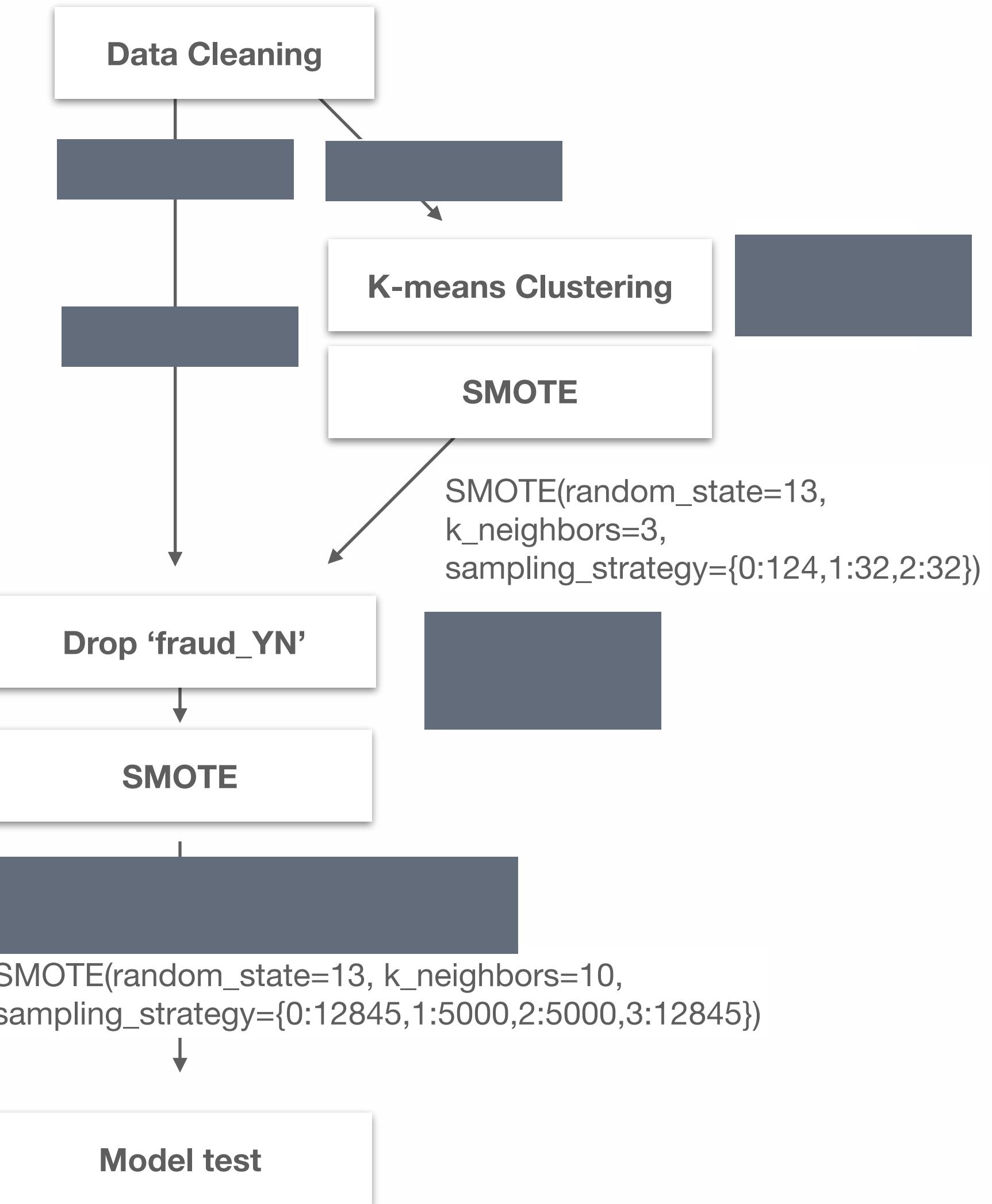
## 또 하나의 샘플링 방법 : SCUT

- About SCUT

## SCUT Algorithm

- Best Model

- Reference



### SCUT의 장점

클래스 내 소수 데이터 패턴을  
잘 살려서 샘플링 가능

### 단순 SMOTE

클래스를 두개로만 나누어 샘플링 =  
모델이 클래스 내에서의 소수 데이터 패턴,  
즉 train data set의 아웃라이어를  
학습하지 못함

### SCUT 적용

각 클래스 내에서의 소수 데이터의  
클래스를 분리하여  
소수의 '소수' 클래스 역시 학습 가능

### 예시

SMOTE : 1 개 / 35690 개  
SCUT : 2869 개 / 35690 개

- About SCUT

- SCUT Algorithm

- Best Model

- Reference

### 단순 SMOTE BEST model

```
from sklearn.linear_model import LogisticRegression  
  
lo_clf = LogisticRegression(random_state=13, solver='liblinear')  
  
lo_clf.fit(X_over, y_over)  
lo_pred = lo_clf.predict(X_test)  
  
print_clf_eval(y_test, lo_pred)  
  
=> confusion matrix  
[[1801 1313]  
 [ 4   3]]
```

Test fraud data => 3 / 7  
Test normal data => 1801 / 3114  
auc : 0.578 recall : 0.428 pre : 0.002

### SCUT BEST model

```
from sklearn.tree import DecisionTreeClassifier  
  
dt_clf = DecisionTreeClassifier(max_depth=7, random_state=13)  
  
dt_clf.fit(X_over_yf, y_over_yf)  
dt_pred = dt_clf.predict(X_test)  
  
print_clf_eval_3(y_test_, dt_pred)  
  
=> confusion matrix  
[[ 3   0   0   4]  
 [ 0   0   0   0]  
 [ 0   0   0   0]  
 [ 574  4   1 2535]]
```

Test fraud data => 3 / 7  
Test normal data => 2535 / 3114  
auc : 0.813 recall : 0.428 pre : 0.005

accuracy가 대폭 상승했다는 점에서 유의미

fraud 데이터의 소수 클래스를 학습함으로써, normal 사고 예측 정확도가 높아진 것으로 추정

- About SCUT

- SCUT Algorithm

- Best Model

- Reference

Best Model !

```
from sklearn.tree import DecisionTreeClassifier  
  
dt_clf = DecisionTreeClassifier(max_depth=7, random_state=13)  
  
dt_clf.fit(X_over_yf, y_over_yf)  
dt_pred= dt_clf.predict(X_test_)  
  
print_clf_eval_3(y_test_, dt_pred)  
  
=> confusion matrix  
[[ 3  0  0  4]  
 [ 0  0  0  0]  
 [ 0  0  0  0]  
 [ 574  4  1 2535]]
```

Test fraud data => 3 / 7  
Test normal data => 2535 / 3114  
Auc : 0.813 recall : 0.428 pre : 0.005

```
from sklearn.tree import DecisionTreeClassifier  
  
dt_clf = DecisionTreeClassifier(max_depth=7, random_state=13)  
  
dt_clf.fit(X_over_yf, y_over_yf)  
dt_pred= dt_clf.predict(X_train_)  
  
print_clf_eval_3(y_train_, dt_pred)  
  
=> confusion matrix  
[[ 26  0  0  5]  
 [ 0  2  0  0]  
 [ 0  0  1  0]  
 [ 1981  7  0 10857]]
```

Train fraud data => 29 / 34  
Train normal data => 10857 / 12845  
auc : 0.845 recall : 0.852 pre : 0.014

- Data 비율 : 0.85 : 0.15
- Feature Selection : All
- 데이터 전처리 : Only 보험사 출동 유무/ 경찰 출동 유무/ B2B 의 변수 정리
- Parameter Tuning :
  - Logistic Regression : random\_state=13, solver='liblinear'
  - Decision Tree : random\_state=13, max\_depth=7
  - Random Forest : random\_state=13, n\_estimators=100

- **About SCUT**

- Jalal Ahammad, Nazia Hossain, January 2020, Credit Card Fraud Detection using Data Pre-processing on Imbalanced Data - both Oversampling and Undersampling(ICCA 2020: Proceedings of the International Conference on Computing Advancements, Article No.: 68, pp 1–4)

- **SCUT Algorithm**

- 정한나, 이정화, 전치혁, March 2010, 불균형 이분 데이터 분류분석을 위한 데이터마이닝 절차 (포항공과대학교 산업경영공학과, Journal of the Korean Institute of Industrial Engineers Vol. 36, No. 1, pp. 13-21)

- **Best Model**

- Astha Agrawal , Herna L. Viktor and Eric Paquet ,2015, SCUT: Multi-Class Imbalanced Data Classification using SMOTE and Cluster-based Undersampling (In Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015) - Volume 1: KDIR, pages 226-234 ISBN: 978-989-758-158-8)

- **Reference**

감사합니다