2021년 해양수산 빅데이터 분석
경진대회

재결서 텍스트 마이닝과 요인분석(FA)을 통한 해양사고 위험성 예측 모델링

NEXT(I조)
고영진
권유진
정재은
동아름

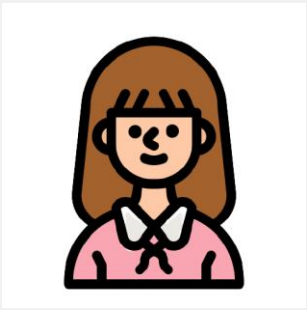
팀원 소개

고영진
(UNIST / 에너지화학공학과)

- √ 팀장
- √ 응원단장
- √ EDA
- √ 특성공학 및 모델링

권유진
(국민대학교 / 빅데이터 경영통계전공)

- √ 텍스트마이닝
- √ EDA
- √ 특성공학



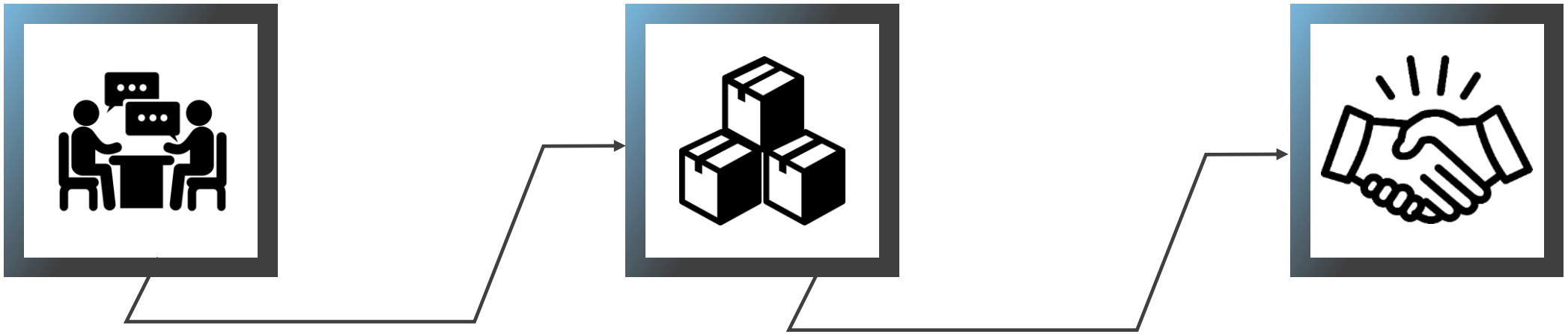
정재은
(공주대학교 / 도시교통공학전공)

- √ 시각화
- √ EDA
- √ Q-GIS

동아름
(인하대학교 / 아태물류학부)

- √ 도메인 지식 제공
- √ PPT 제작

Contents



개요

1 문제 배경

2 목적 및 가설 설정

본론

3 재결서 텍스트 마이닝

4 EDA, 데이터 전처리, 특성공학

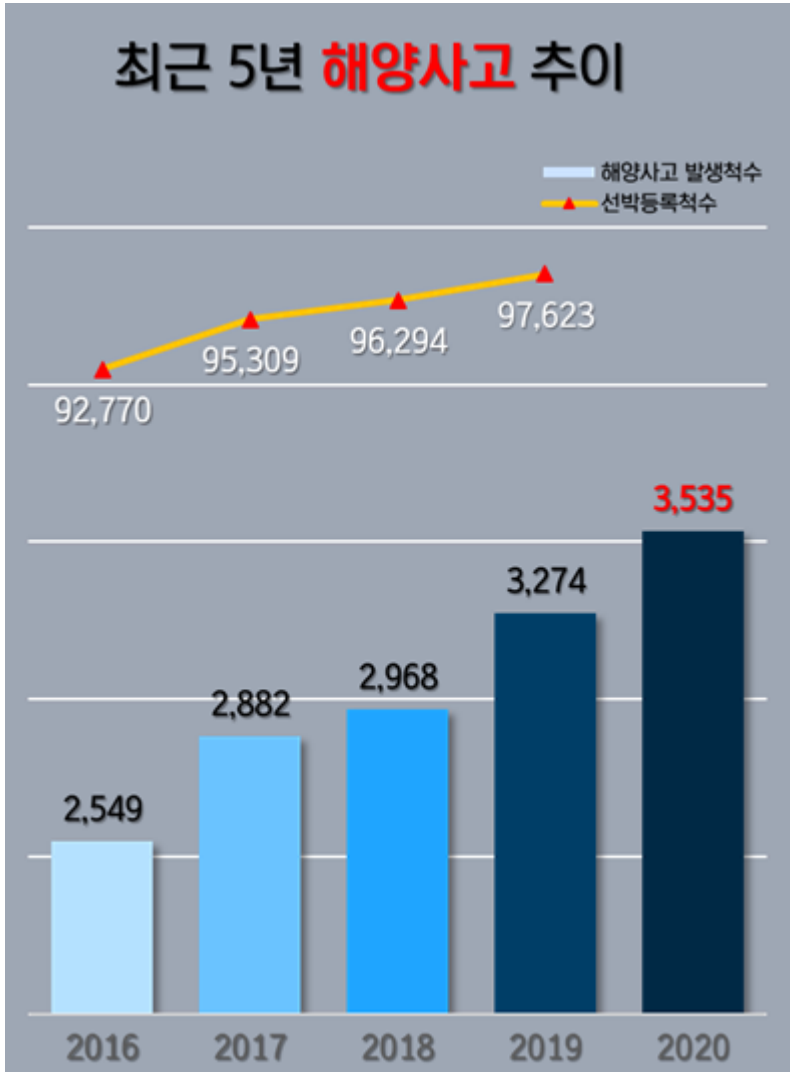
5 요인분석

6 모델링 및 평가

결론

7 결론 및 활용 방안

01. 문제 배경



출처 - 해양수산부 중앙해양안전심판원 2020년 해양사고 통계

해양사고의 심각성

13,687
최근 5년간
사고 발생(건)

15,208
최근 5년간
사고 선박(척)

2,489
최근 5년간
인명 피해(명)

해양사고발생률은 2016년 2.75%에서 2019년 3.35%로 증가
우리나라의 최근 해양사고 추이는 **매년 증가 추세로**,
해양사고 원인 분석 필요

01. 문제 배경

재 결 요 약 서

사 건 명	어선 제71용득호 침몰사건 (부산해심 제2021-033호)
해양사고관련자 (직책, 해기면허)	A(제71용득호 선장, 6급항해사)
판 시 사 항	1. 판시요지 가. 기선권현망 선단의 주선과 종선이 접현·결합한 상태로 기상이 악화되는 가운데 무리하게 항해하면서 양선을 연결한 계류줄 상태를 제대로 확인하지 아니하여 계류줄이 풀리면서 주선이 파도에 복원력을 상실하여 발생한 사건 2. 관련법규 선원법 제16조(항해의 안전 확보)
주제어	기선권현망, 주선, 계류줄
사 건 개 요	☐ 관련선박 선명 / 용도 / 총톤수/길이(GT/m) / 선박소유자/선적항 / 피해 제71용득호 / 어선 / 27톤/21.38m / B/경남 창원시 마산합포구 / - 선체 침몰 - 연료유 일부 해상 유출 ☐ 일시 : 2021. 2. 19. 20:34경 ☐ 장소 : 북위 35도 18분 05초 · 동경 129도 20분 38초 (부산 기장군 고리에서 방위 약 115도, 거리 약 2.4해리 해상) ☐ 사고경위 기선권현망 선단의 주선과 종선이 접현·결합한 상태로 기상이 악화되는 가운데 무리하게 항해하면서 양선을 연결한 계류줄 상태를 제대로 확인하지 아니하여 계류줄이 풀리면서 주선이 파도에 복원력을 상실하여 위 일시와 장소에서 제71용득호가 침몰한 사건임.
주 문	이 침몰사건은 기선권현망 선단의 주선과 종선이 접현·결합한 상태로 기상이 악화되는 가운데 무리하게 항해하면서 양선을 연결한 계류줄 상태를 제대로 확인하지 아니하여 계류줄이 풀리면서 주선이 파도에 복원력을 상실하여 발생한 것이다. 해양사고관련자 A의 6급항해사 업무를 2개월 정지한다.

출처 - 부산해양안전심판원 제2021-033호 재결서

재결서 정보 활용

재결요약서	내용
사건명	사건명, 사건번호
해양사고관련자	직책, 해기면허
판시사항	판시요지, 관련법규
주제어	주요 주제어
사건개요	관련 선박 정보, 일시, 장소, 사고경위
주문	사건경위, 판결내용
교훈	교훈

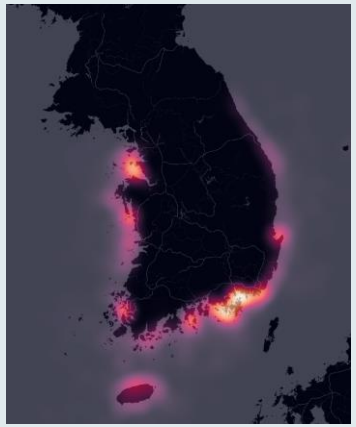
사고발생 원인

"기상이 악화되는 가운데 무리하게 항해하면서 양선을 연결한 계류줄 상태를 제대로 확인하지 아니하여 계류줄이 풀리면서 주선이 파도에 복원력을 상실하여 발생한 사건"

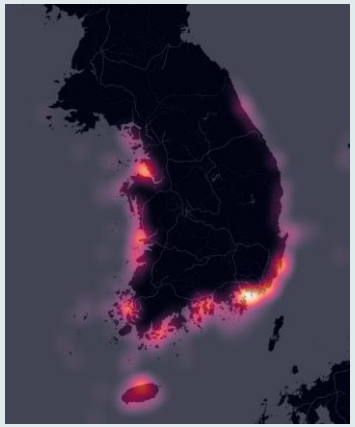
사고유형에 대한 여러 **복합적인 원인**을 담고 있음
재결서는 비정형데이터로 분석을 위해 정형화할 필요가 있음

02. 가설설정 및 목적

요인분석

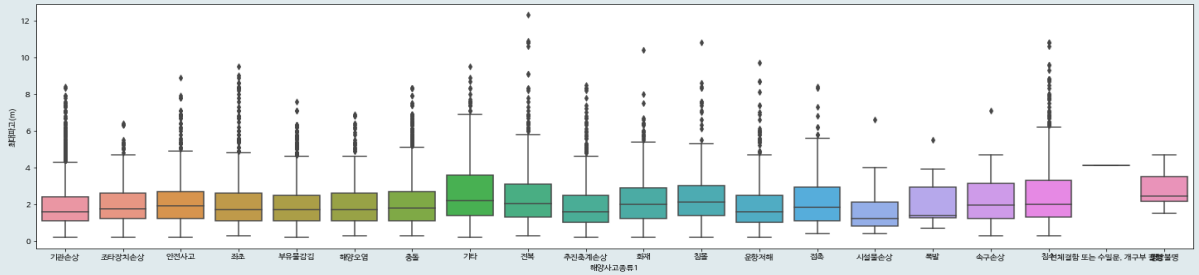


▲ 기관손상에 의한 사고 위치

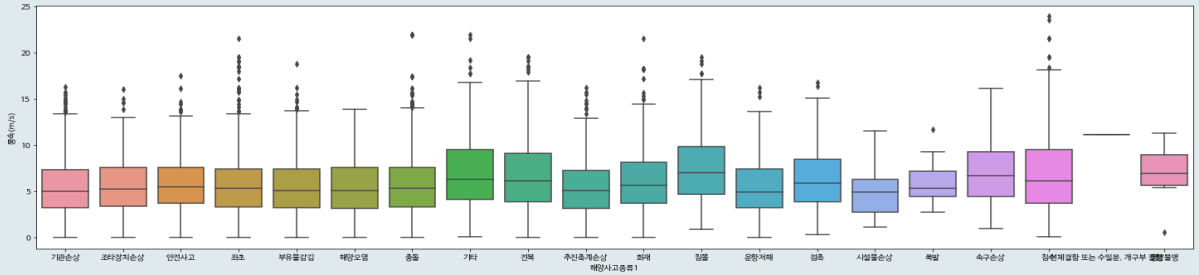


▲ 전복에 의한 사고 위치

▲ 화재·폭발에 의한 사고 위치



▲ 최대파고와 해양사고종류와의 관계



▲ 풍속과 해양사고종류와의 관계

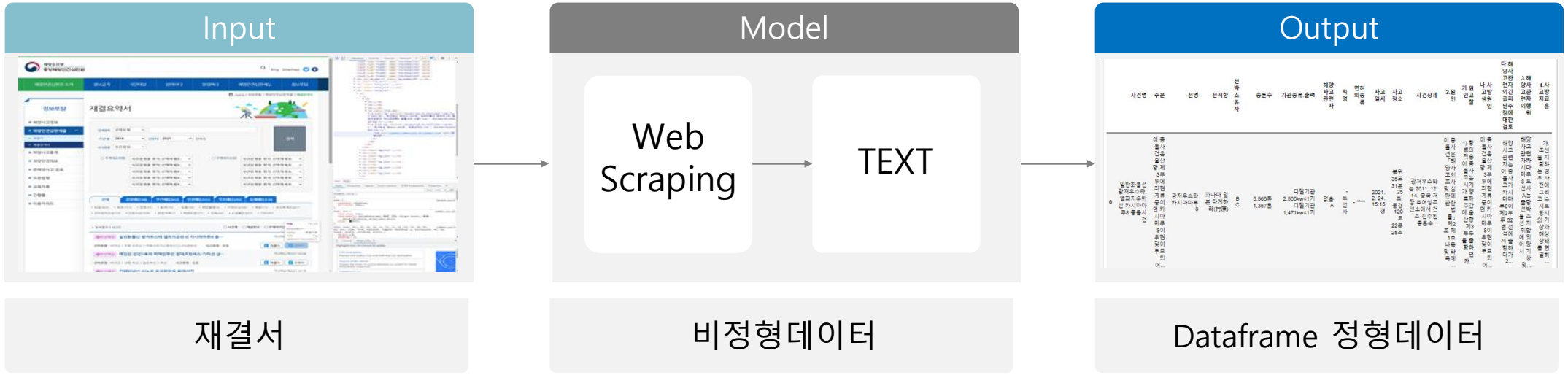
가설 설정

- 재결서를 이용하면 사고유형에 대한 **복합적인 원인을 식별**할 수 있음
- 요인분석으로 사고 위험성에 대한 **잠재변수 도출**할 수 있음

목적

- 재결서를 **정형화**하고 사고 원인 변수화
- 데이터들을 **요인 분석**하여 사고를 예측함으로써 사고 예방률 증진

03. 재결서 텍스트 마이닝



03. 재결서 텍스트 마이닝

사고 A, B가 함께 발생 가능!

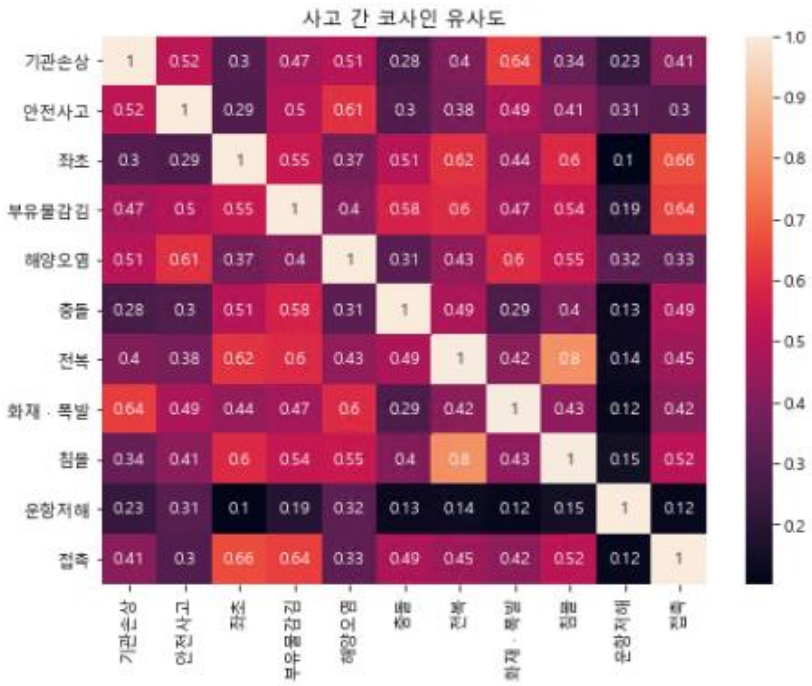
사건상세

column

FastText

사고 간 연관성

- 전복, 침몰 0.795988
- 좌초, 접촉 0.663724
- 부유물감김, 접촉 0.640012
- …



사고 A의 원인을 기술할 때 자주 등장하는 단어!

사고발생 원인

column

FastText

사고 별 원인

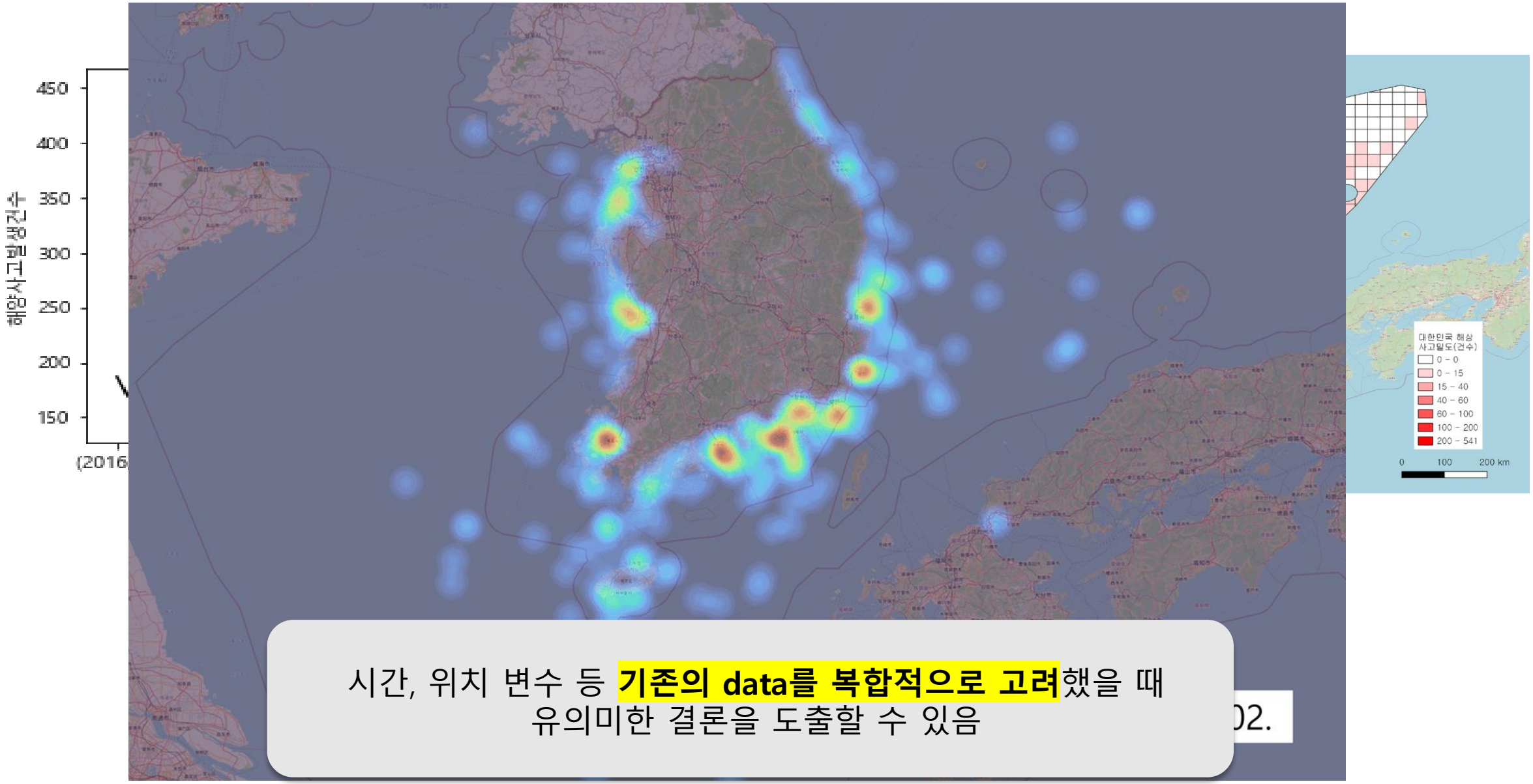
- 안전사고 : 작업자, 검증, 통제 …
- 화재 · 폭발 : 산소, 물질, 발전기 …
- 충돌 : 소극, 과속, 바위섬 …
- …

clustering

DBSCAN

Feature

04. EDA, 데이터 전처리, 특성공학



시간, 위치 변수 등 **기존의 data를 복합적으로 고려**했을 때
유의미한 결론을 도출할 수 있음

04. EDA, 데이터 전처리, 특성공학

기존 데이터
(시간, 위치, 선박 연령, 선박 용도 등)

+

외부데이터
(기상, 파고부이)

+

재결서 분석을 통한 사고 종류 원인

- 중복데이터 제거
- 결측치 제거 및 대체
- 사고 위험도 3가지 지표 설정(target)
  - 인명피해여부, 선박피해정도, 해양사고종류

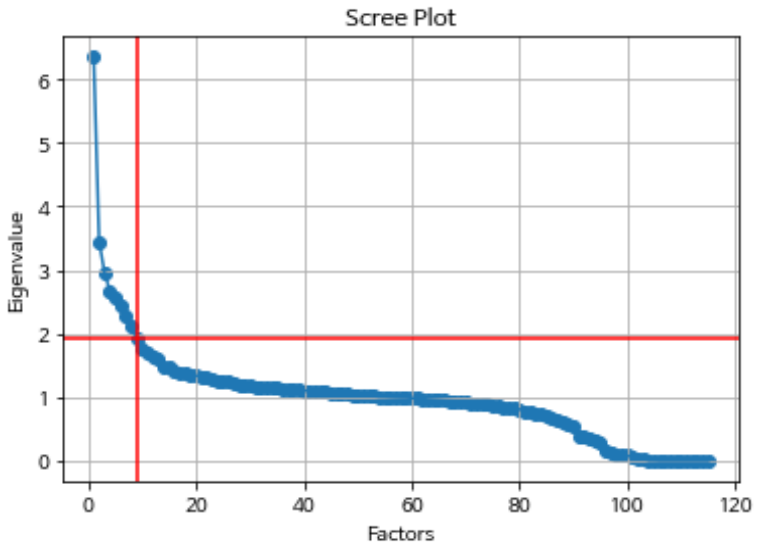
- Categorical 변수 One-Hot Encoding
- Target의 imbalance 해소(oversampling)
- 이후 train set과 test set으로 분리

115가지의 변수
3가지 target

```
 0   해양사고발생(년도)      15051 non-null  int64
 1   해양사고발생(월)       15051 non-null  int64
 2   해양사고발생시간대       15051 non-null  object
 3   계절              15051 non-null  object
 4   해양사고발생지역(줌)     15051 non-null  object
 5   선박연령            15051 non-null  float64
 6   톤수범위(통계용)       15051 non-null  object
 7   선박용도(줌)         15051 non-null  object
 8   latitude        15051 non-null  float64
 9   longitude       15051 non-null  float64
 10  풍속(m/s)         15051 non-null  float64
 11  풍향(deg)         15051 non-null  float64
 12  GUST풍속(m/s)     15051 non-null  float64
 13  현지기압(hPa)       15051 non-null  float64
 14  습도(%)           15051 non-null  float64
 15  기온(°C)          15051 non-null  float64
 16  수온(°C)          15051 non-null  float64
 17  최대파고(m)         15051 non-null  float64
 18  유의파고(m)         15051 non-null  float64
 19  평균파고(m)         15051 non-null  float64
 20  파주기(sec)        15051 non-null  float64
 21  파향(deg)         15051 non-null  float64
 22  인적요인            15051 non-null  object
 23  물적요인            15051 non-null  object
 24  위치요인            15051 non-null  object
 25  기상요인            15051 non-null  object
 26  시간요인            15051 non-null  object
 27  인명 피해 및 부상 여부  15051 non-null  float64
 28  선박피해            15051 non-null  object
 29  해양사고종류          15051 non-null  object
dtypes: float64(16), int64(2), object(12)
memory usage: 3.6+ MB
```

05. 요인분석(Factor Analysis)

One-Hot encoding후 115개의 변수



▲ 핵심 요인 개수 선정을 위한 Scree Plot

공통 성질을 묶어낸 9개의 핵심 요인

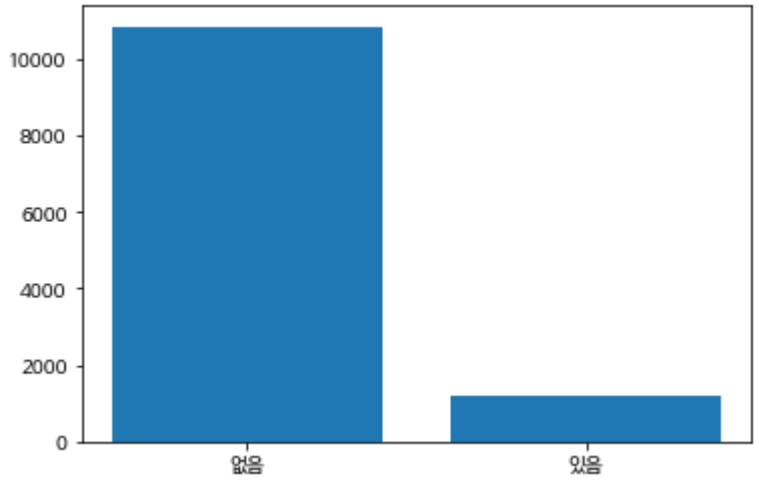
```
1 train_X_human = fa_human.transform(train_X_human)
2 train_X_human.shape
```

```
(21704, 9)
```

```
1 test_X_human = fa_human.transform(test_X)
2 test_X_human.shape
```

```
(3011, 9)
```

06. 모델링 및 평가



▲ dataset imbalance(인명피해발생여부)

평가지표를 **정확도**보다는
AUC(Area Under Curve)를 사용

Model	Accuracy	AUC
Extra Trees Classifier	0.9793	0.9986
Random Forest Classifier	0.9718	0.9980
Light Gradient Boosting Machine	0.8565	0.9298

**Train Set에 대해
상위 3개 classifier 선정**

**Blending을 통한
앙상블 모델**

06. 모델링 및 평가

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9694	0.9945	0.9987	0.9433	0.9702	0.9388	0.9404
1	0.9641	0.9940	0.9947	0.9373	0.9651	0.9283	0.9300
2	0.9710	0.9955	0.9993	0.9457	0.9718	0.9421	0.9436
3	0.9612	0.9947	0.9947	0.9321	0.9624	0.9223	0.9244
4	0.9700	0.9971	0.9974	0.9456	0.9708	0.9401	0.9415
Mean	0.9672	0.9952	0.9970	0.9408	0.9681	0.9343	0.9360
SD	0.0038	0.0011	0.0019	0.0053	0.0037	0.0077	0.0074

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Voting Classifier	0.9754	0.9963	0.9969	0.956	0.976	0.9509	0.9517

▲ 인명 피해 및 부상 여부 예측 모델 CV

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9245	0.9900	0.9246	0.9238	0.9230	0.9056	0.9061
1	0.9225	0.9904	0.9226	0.9216	0.9210	0.9032	0.9037
2	0.9153	0.9890	0.9154	0.9133	0.9135	0.8941	0.8945
3	0.9216	0.9912	0.9217	0.9212	0.9201	0.9020	0.9027
4	0.9190	0.9894	0.9191	0.9188	0.9178	0.8987	0.8993
Mean	0.9206	0.9900	0.9207	0.9197	0.9191	0.9007	0.9013
SD	0.0032	0.0008	0.0032	0.0036	0.0032	0.0040	0.0040

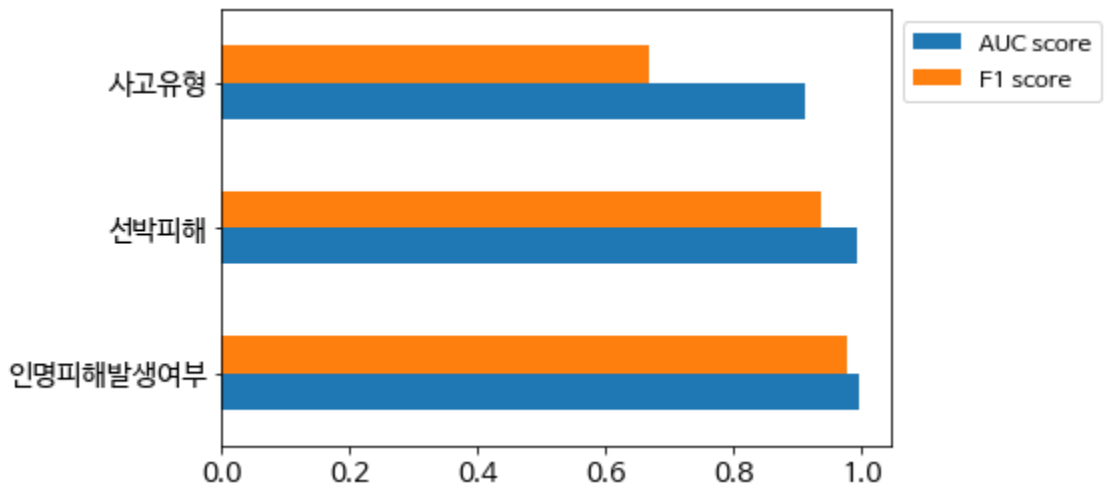
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Voting Classifier	0.9382	0.9931	0.938	0.9382	0.9372	0.9228	0.9233

▲ 선박 피해 구분 예측 모델 CV

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.6536	0.8970	0.6520	0.6305	0.6393	0.5843	0.5853
1	0.6603	0.9035	0.6587	0.6387	0.6469	0.5923	0.5933
2	0.6557	0.8974	0.6541	0.6321	0.6410	0.5868	0.5880
3	0.6556	0.8950	0.6540	0.6344	0.6425	0.5867	0.5876
4	0.6415	0.8952	0.6398	0.6145	0.6248	0.5697	0.5709
Mean	0.6533	0.8976	0.6517	0.6300	0.6389	0.5840	0.5850
SD	0.0063	0.0031	0.0064	0.0082	0.0075	0.0076	0.0075

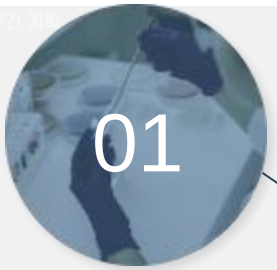
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Voting Classifier	0.6805	0.9123	0.6838	0.6592	0.6675	0.6165	0.6174

▲ 발생가능 사고 유형 예측 모델 CV



▲ 각 모델의 F1, AUC scores

07. 결론 및 활용 방안



[다양한 변수]

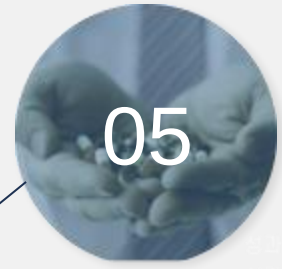
톤수, 선박 노후화,
시간, 기상, 해역, 선박용도 등
다양한 변수 존재

[Factor Analysis]

복합적으로 작용하는 변수들의
요인분석(FA)을 통해 잠재 변수도출

[예측 모델링]

인명피해 및 부상 발생,
선박 피해정도,
발생 가능 사고 종류
예측 모델링 구상



**사고발생 예측 및
해양안전사고 예방**

[재결서 분석]

재결서 텍스트마이닝을 통해
사고유형에 따른 원인 분석
및 변수화

07. 결론 및 활용 방안

해양수산 빅데이터 플랫폼

출처 - 해양수산부

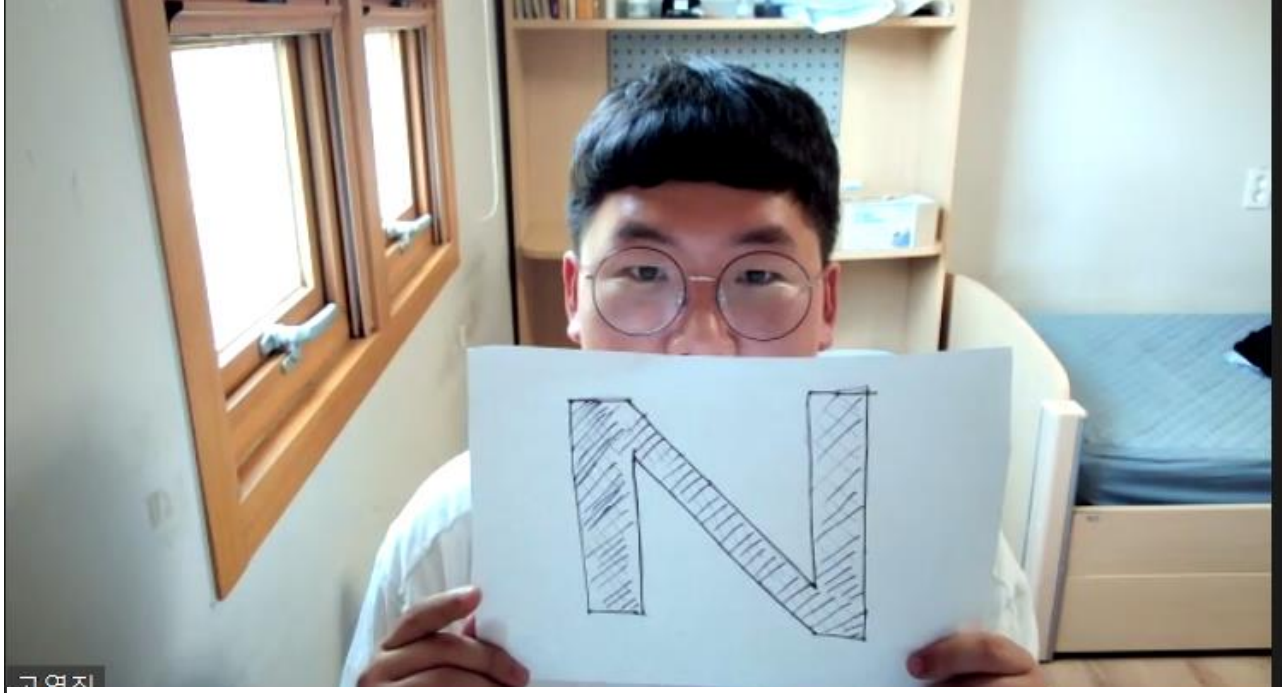
✓ 재결서 분석법을 활용하여 복합적 원인을 분석하여 양질의 데이터를 생성할 수 있음.

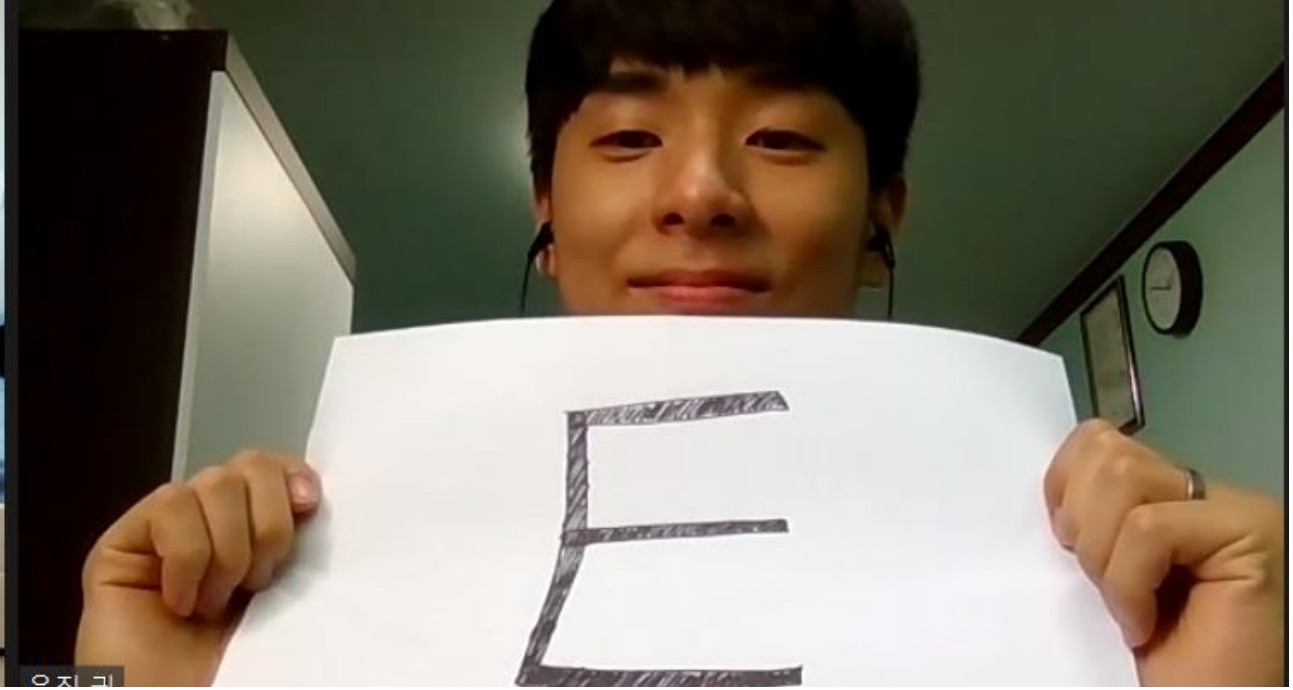
e-Navigation

출처 - 해양수산부

✓ 선박의 위치와 정보를 실시간으로 파악하여 현 상황의 위험 요인을 사용자에게 제공할 수 있음.

✓ 해양 사고 발생률을 감소시킬 수 있음.





“ 감사합니다 ”

Appendix A. – Target의 class 지정

```python
def loss_of_ship(x):

    if '무손' in x:
        new_loss = '무손'

    elif ('경손' in x) or ('감항능력 유지' in x):
        new_loss = '경손'

    elif ('중손' in x) or ('감항능력 상실' in x):
        new_loss = '중손'

    elif ('전손' in x) or ('침몰후인양' in x):
        new_loss = '전손'

    else:
        new_loss = '기타'

    return new_loss

raw_data['선박피해'] = raw_data['선박피해구분'].apply(loss_of_ship)
raw_data.loc[raw_data['선박피해'] == '무손', '선박피해'] = 0.0
raw_data.loc[raw_data['선박피해'] == '경손', '선박피해'] = 1.0
raw_data.loc[raw_data['선박피해'] == '중손', '선박피해'] = 2.0
raw_data.loc[raw_data['선박피해'] == '전손', '선박피해'] = 3.0
raw_data.loc[raw_data['선박피해'] == '기타', '선박피해'] = 4.0
```

선박피해구분 18가지를 5가지 class로 축소

```python
def accident_case(x):
    if x == '기관손상':
        new_case = '기관손상'

    elif x == '화재 · 폭발':
        new_case = '화재 · 폭발'

    elif x in ['좌초', '충돌', '전복', '침몰', '접촉']:
        new_case = '충돌 · 전복 · 침몰'

    elif x == '부유물감김':
        new_case = '부유물감김'

    elif x == '안전사고':
        new_case = '안전사고'

    else:
        new_case = '기타'

    return new_case

raw_data['해양사고종류'] = raw_data['해양사고종류(통계용)'].apply(accident_case)

raw_data.loc[raw_data['해양사고종류'] == '기관손상', '해양사고종류'] = 0.0
raw_data.loc[raw_data['해양사고종류'] == '화재 · 폭발', '해양사고종류'] = 1.0
raw_data.loc[raw_data['해양사고종류'] == '충돌 · 전복 · 침몰', '해양사고종류'] = 2.0
raw_data.loc[raw_data['해양사고종류'] == '부유물감김', '해양사고종류'] = 3.0
raw_data.loc[raw_data['해양사고종류'] == '안전사고', '해양사고종류'] = 4.0
raw_data.loc[raw_data['해양사고종류'] == '기타', '해양사고종류'] = 5.0
```

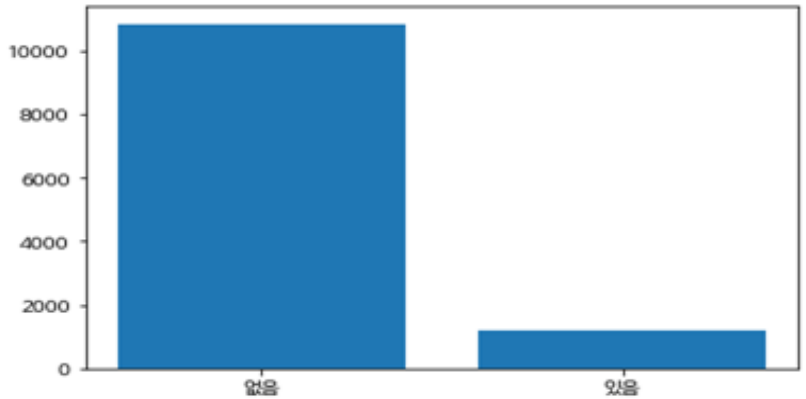
해양사고 종류를 6가지로 축소

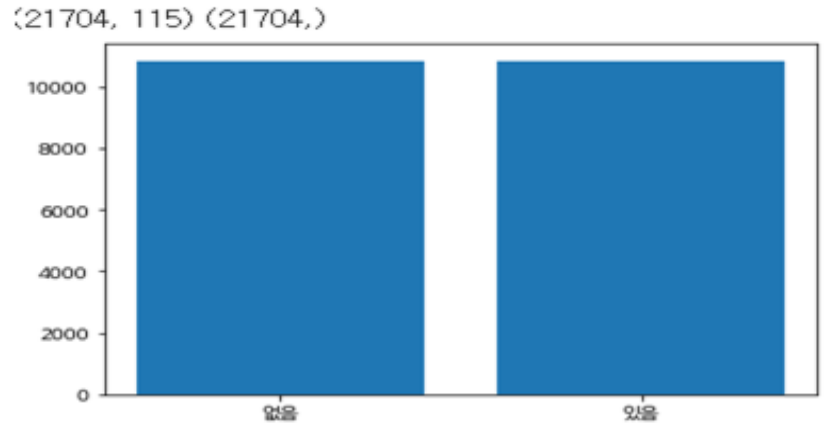
Appendix B. – 재결서 텍스트마이닝에 의한 요인 변수 생성

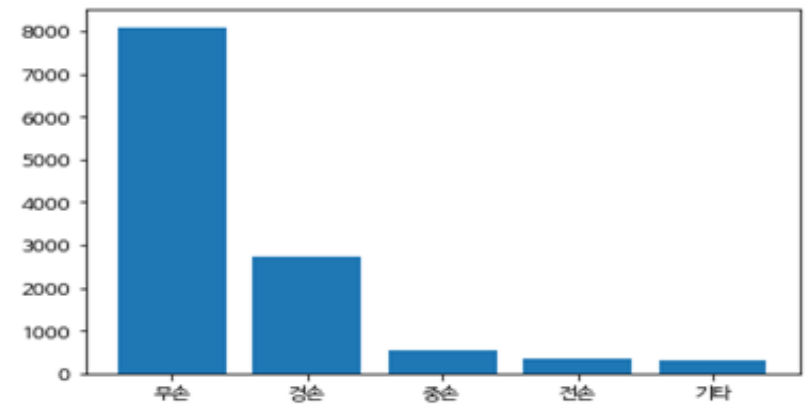
```python
def human_factor(x):
    if x in ['기타', '전복']:
        factor = 'X'
    else:
        factor = '인적요인'
    return factor

def material_factor(x):
    if x in ['기관손상', '해양오염', '전복', '화재 · 폭발', '침몰']:
        factor = '물적요인'
    else:
        factor = "X"
    return factor

def location_factor(x):
    if x in ['좌초', '부유물감김', '충돌', '접촉']:
        factor = '위치요인'
    else:
        factor = "X"
    return factor

def weather_factor(x):
    if x in ['접촉', '침몰', '운항저해', '전복', '충돌', '부유물감김']:
        factor = '기상요인'
    else:
        factor = 'X'
    return factor

def time_factor(x):
    if x == '운항저해':
        factor = '시간요인'
    else:
        factor = 'X'
    return factor
```

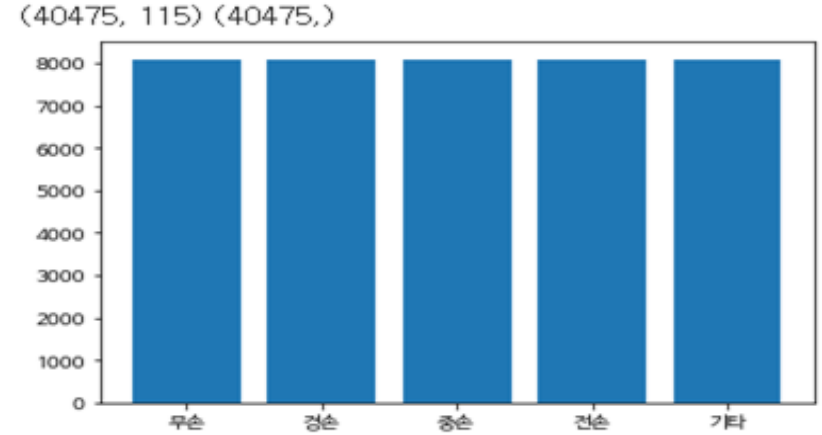
```python
raw_data['인적요인'] = raw_data['해양사고종류(통계용)'].apply(human_factor)
raw_data['물적요인'] = raw_data['해양사고종류(통계용)'].apply(material_factor)
raw_data['위치요인'] = raw_data['해양사고종류(통계용)'].apply(location_factor)
raw_data['기상요인'] = raw_data['해양사고종류(통계용)'].apply(weather_factor)
raw_data['시간요인'] = raw_data['해양사고종류(통계용)'].apply(time_factor)

raw_data.loc[raw_data['인적요인'] == '인적요인', '인적요인'] = 1.0
raw_data.loc[raw_data['인적요인'] == 'X', '인적요인'] = 0.0
raw_data.loc[raw_data['물적요인'] == '물적요인', '물적요인'] = 1.0
raw_data.loc[raw_data['물적요인'] == 'X', '물적요인'] = 0.0
raw_data.loc[raw_data['위치요인'] == '위치요인', '위치요인'] = 1.0
raw_data.loc[raw_data['위치요인'] == 'X', '위치요인'] = 0.0
raw_data.loc[raw_data['기상요인'] == '기상요인', '기상요인'] = 1.0
raw_data.loc[raw_data['기상요인'] == 'X', '기상요인'] = 0.0
raw_data.loc[raw_data['시간요인'] == '시간요인', '시간요인'] = 1.0
raw_data.loc[raw_data['시간요인'] == 'X', '시간요인'] = 0.0
raw_data.info()
```

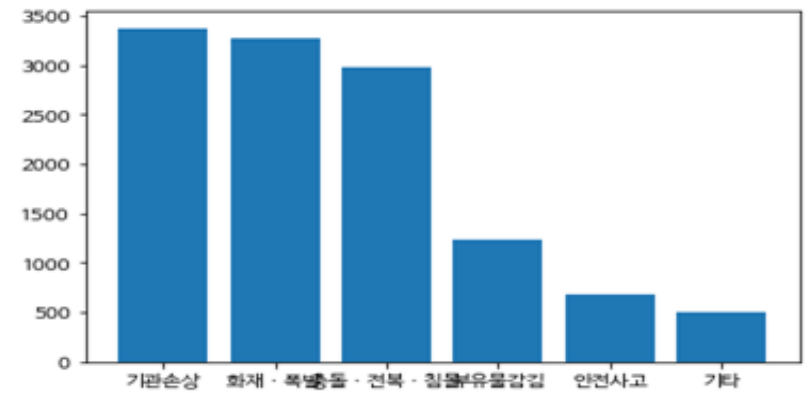
Appendix C. – Oversampling에 의한 data 변화

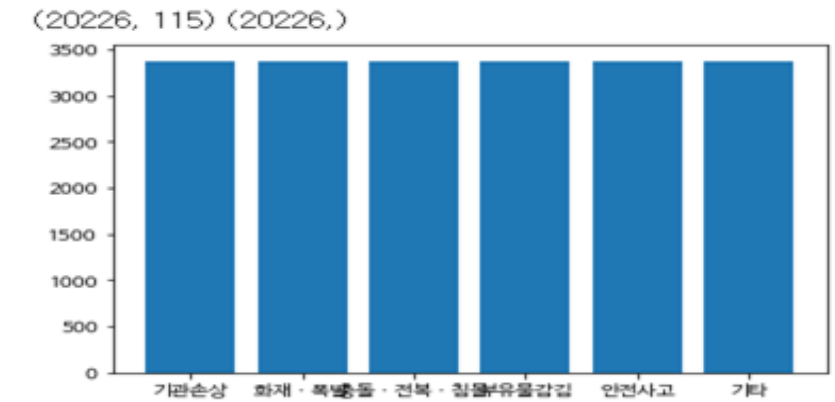












Appendix D. – 기상 데이터 매칭 방법

```
def nearest_point(data):
    start = data['해양사고장소(위도)'], data['해양사고장소(경도)']
    a = data['해양사고발생일시'][:13]
    b = int(a[:4] + a[5:7] + a[8:10] + a[11:13])

    if b < standard1:
        distances = []
        for i, x in enumerate(until_20191029):
            idx = list(longi_lati['지점 번호']).index(x)
            goal = longi_lati['위도'][idx], longi_lati['경도'][idx]
            distance = haversine(start, goal)
        distances.append(distance)
        nearest = np.argmin(distances)
        nearest_point = until_20191029[nearest]

    elif (b >= standard1) and (b < standard2):
        distances = []
        for i, x in enumerate(after_2019102916):
            idx = list(longi_lati['지점 번호']).index(x)
            goal = longi_lati['위도'][idx], longi_lati['경도'][idx]
            distance = haversine(start, goal)
        distances.append(distance)
        nearest = np.argmin(distances)
        nearest_point = after_2019102916[nearest]

    elif (b >= standard2) and (b < standard3):
        distances = []
        for i, x in enumerate(after_2019103015):
            idx = list(longi_lati['지점 번호']).index(x)
            goal = longi_lati['위도'][idx], longi_lati['경도'][idx]
            distance = haversine(start, goal)
        distances.append(distance)
        nearest = np.argmin(distances)
        nearest_point = after_2019103015[nearest]

    elif (b >= standard3) and (b < standard4):
        distances = []
        for i, x in enumerate(after_2019110117):
            idx = list(longi_lati['지점 번호']).index(x)
            goal = longi_lati['위도'][idx], longi_lati['경도'][idx]
            distance = haversine(start, goal)
        distances.append(distance)
        nearest = np.argmin(distances)
        nearest_point = after_2019110117[nearest]
```

```
    elif (b >= standard4) and (b < standard5):
        distances = []
```

```
clssifier = setup(data = train_set_human,
                  target = '인명 피해 및 부상 여부',
                  numeric_imputation = 'mean',
                  normalize = True,
                  silent = True)
```

```
# 좀 오래걸림 데이터 2만개, 2분 소요

best_3_human = compare_models(sort = 'AUC', n_select=3)
```

```
# 좀 오래걸림 데이터 2만개, 2분 소요

blended_human = blend_models(estimator_list= best_3_human, fold = 5, optimize='AUC')
pred_holdout = predict_model(blended_human)
final_model_human = finalize_model(blended_human)
pred_esb_human = predict_model(final_model_human, test_set_human)
```

```
start = time.time()
data['지점'] = data.apply(nearest_point, axis = 1)
end = time.time()

print(end - start)
```

Appendix E. – pycaret.clasffication을 통한 학습 및 평가

```
clssifier = setup(data = train_set_human,
                  target = '인명 피해 및 부상 여부',
                  numeric_imputation = 'mean',
                  normalize = True,
                  silent = True)
```

```
# 좀 오래걸림 데이터 2만개, 2분 소요

best_3_human = compare_models(sort = 'AUC', n_select=3)
```

```
# 좀 오래걸림 데이터 2만개, 2분 소요

blended_human = blend_models(estimator_list= best_3_human, fold = 5, optimize='AUC')
pred_holdout = predict_model(blended_human)
final_model_human = finalize_model(blended_human)
pred_esb_human = predict_model(final_model_human, test_set_human)
```