

선박



해운 물류 분야  
전 세계 물동량 85%

선박



코로나 19 이후  
물류 정체 심각

다수의 항만 내  
선박 대기시간 증가


선박



# 기다리 면





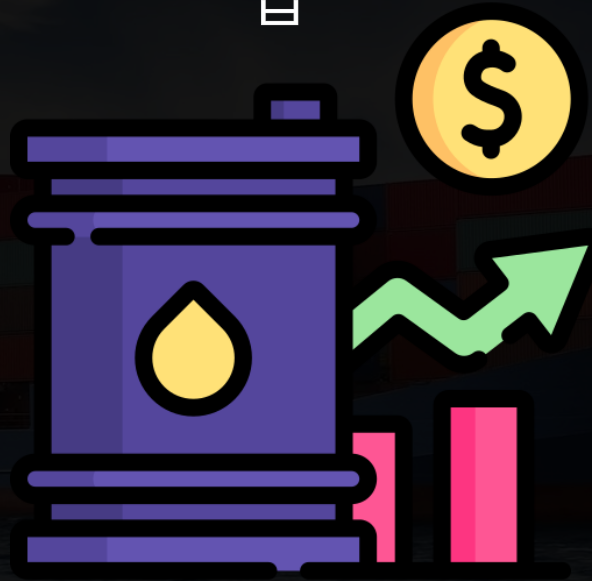
기다리면 

소비자



| 물류  
지연으로  
인한 불편

기업




| 기름값  
증가

세계



| 온실가스 배출로  
인한 지구 온난화  
가속화

예측하면 

소비자



물류 지연  
해결

기업



기름값  
감소

세계



배출 온실가스 절감

분석하면 



항만 및 해운 기업에  
대기 시간 예측을  
위한 AI 플랫폼 및  
솔루션 제공



항차(선박의 여정)  
데이터를 이용한 항만 내  
선박 대기시간 예측



P Port Terminal Ship Time  
Delay Prediction



PTSD



# PTSD : Port Terminal Ship Time Delay Prediction



# INDEX

---

01 주제 선정  
이유

02 데이터 구성 설명

03 데이터 전처리

04 모델링 및  
결과

인공지능 공학



## 02 데이터 구성 설명







## 02 데이터 구성 설명

내 드라이브 > 인공지능 공학 팀플 > open ▾

유형 ▾

사람 ▾

수정 날짜 ▾

이름 ↑

소유자



test.csv



나

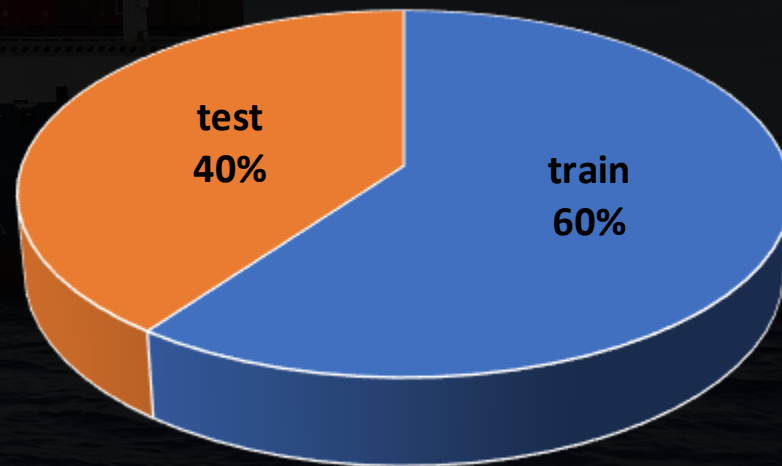


train.csv



나

Dataset



## 02 데이터 확인

✓  
0초



train.info()



<class 'pandas.core.frame.DataFrame'>

RangeIndex: 391939 entries, 0 to 391938

Data columns (total 22 columns):

#	Column	Non-Null Count	Dtype
0	ARI_CO	391939 non-null	object
1	ARI_PO	391939 non-null	object
2	SHIP_TYPE_CATEGORY	391939 non-null	object
3	DIST	391939 non-null	float64
4	ATA	391939 non-null	object
5	ID	391939 non-null	object
6	BREADTH	391938 non-null	float64
7	BUILT	391939 non-null	int64
8	DEADWEIGHT	391939 non-null	int64
9	DEPTH	391938 non-null	float64
10	DRAUGHT	391938 non-null	float64
11	GT	391939 non-null	int64
12	LENGTH	391938 non-null	float64
13	SHIPMANAGER	391939 non-null	object
14	FLAG	391939 non-null	object
15	U_WIND	228251 non-null	float64
16	V_WIND	228251 non-null	float64
17	AIR_TEMPERATURE	227309 non-null	float64
18	BN	228251 non-null	float64
19	ATA_LT	391939 non-null	int64
20	PORT_SIZE	391939 non-null	float64
21	CI_HOUR	391939 non-null	float64

dtypes: float64(11), int64(4), object(7)

memory usage: 65.8+ MB



## 02 데이터 구성 설명

Feature Name	Description	단위
ARI_CO	소속 국가	X
ARO_PO	항구명	X
SHIP_TYPE_CATEGORY	선종	X
...		
PORT_SIZE	접안지 크기	Km^2
CI_HOUR	대기시간	Hour

Columns = 22

Rows = 391939

target

## 02 Columns 정보 확인

	A	B	C
1	Feature Name	Description	단위
2	ARI_CO	도착항의 소속국가(도착항 앞 2글자)	
3	ARI_PO	도착항의 항구명(도착항 뒤 글자)	
4	SHIP_TYPE_CATEGORY	선종 통합 바탕으로 5대 선종으로 분류	
5	DIST	정박지(ber_port)와 접안지 사이의 거리	km
6	ATA	anc_port에 도착한 시점의 utc. 실제 정박 시각(Actual Time of Arrival)	hour
7	ID	선박식별 일련번호	
8	BREADTH	선박의 폭	m
9	BUILT	선박의 연령	year
10	DEADWEIGHT	선박의 재화중량톤수	ton
11	DEPTH	선박의 깊이	m
12	DRAUGHT	흘수 높이	m
13	GT	용적톤수(Gross Tonnage)값	GT(m^3)
14	LENGTH	선박의 길이	m
15	SHIPMANAGER	선박 소유주	
16	FLAG	선박의 국적	
17	U_WIND	풍향 u벡터	m/s
18	V_WIND	풍향 v벡터	m/s
19	AIR_TEMPERATURE	기온	°C
20	BN	보퍼트 풍력 계급	
21	ATA_LT	anc_port에 도착한 시점의 현지 정박 시각(Local Time of Arrival)(단위 : H)	hour
22	PORT_SIZE	접안지 폴리곤 영역의 크기	km^2
23	CI_HOUR	CI_HOUR(Target)	CI_HOUR

**DEADWEIGHT:** 선박이 적재할 수 있는 화물의 최대 중량

GT: 선체의 총 용적(부피)에서 추진, 항해, 안전에 관련된 공간을 제외한 모든 부분  
**U,V\_WIND:** 바람의 속도

BN: 해상의 풍랑상태를 기초로 하여 만든 풍력 계급

# 02 데이터 확인

0초

▶

train.head(5)

↑ ↓ ↺ ⌨ ⚙ 📄 🗑 :

	ARI_CO	ARI_PO	SHIP_TYPE_CATEGORY	DIST	ATA	ID	BREADTH	BUILT	DEADWEIGHT	DEPTH	...	LENGTH	SHIPMANAGER	FLAG	U_WIND	V_WIND	AIR_TEMPERATU
0	SG	GIW5	Container	30.881018	2018-12-17 21:29	Z618338	30.0	24	24300	10.0	...	180.0	CQSB78	Panama	NaN	NaN	N
1	IN	UJM2	Bulk	0.000000	2014-09-23 6:59	X886125	30.0	13	35900	10.0	...	180.0	SPNO34	Marshall Islands	NaN	NaN	N
2	CN	EUC8	Container	0.000000	2015-02-03 22:00	T674582	50.0	12	146000	30.0	...	370.0	FNPK22	Malta	NaN	NaN	N
3	JP	ZAG4	Container	0.000000	2020-01-17 4:02	Y847238	20.0	18	6910	10.0	...	120.0	PBZV77	Bahamas	-3.18	-1.61	
4	SG	GIW5	Container	27.037650	2020-01-26 7:51	A872328	50.0	10	116000	20.0	...	300.0	GUCE76	Liberia	-0.33	-3.28	2

5 rows × 22 columns



## 02 결측치 확인

✓  
0초



#결측치 확인

```
combined_df = pd.concat([train.isnull().sum(), test.isnull().sum()], axis=1)
combined_df.columns = ['Train', 'Test']
print(combined_df)
```



	Train	Test
ARI_CO	0	0.0
ARI_PO	0	0.0
SHIP_TYPE_CATEGORY	0	0.0
DIST	0	0.0
ATA	0	0.0
ID	0	0.0
BREADTH	1	0.0
BUILT	0	0.0
DEADWEIGHT	0	0.0
DEPTH	1	0.0
DRAUGHT	1	0.0
GT	0	0.0
LENGTH	1	0.0
SHIPMANAGER	0	0.0
FLAG	0	0.0
U_WIND	163688	91725.0
V_WIND	163688	91725.0
AIR_TEMPERATURE	164630	92246.0
BN	163688	91725.0
ATA_LT	0	0.0
PORT_SIZE	0	0.0
CI_HOUR	0	NaN

## 02 왜도, 첨도 확인

✓ 0초 # 각 칼럼의 왜도 확인  
`train.skew(numeric_only=True)`

DIST	2.482926
BREADTH	0.348941
BUILT	0.836319
DEADWEIGHT	1.571573
DEPTH	-0.029870
DRAUGHT	0.395419
GT	1.498998
LENGTH	0.203090
U_WIND	-0.031121
V_WIND	-0.513227
AIR_TEMPERATURE	-0.782502
BN	0.351398
ATA_LT	-0.067106
PORT_SIZE	0.865327
CI_HOUR	6.162590
dtype: float64	

✓ 0초 # 각 칼럼의 첨도 확인  
`train.kurtosis(numeric_only=True)`

DIST	6.959160
BREADTH	0.154824
BUILT	0.933388
DEADWEIGHT	2.557812
DEPTH	-0.294887
DRAUGHT	1.278828
GT	2.271098
LENGTH	-0.508877
U_WIND	1.141962
V_WIND	1.624910
AIR_TEMPERATURE	-0.038454
BN	0.410539
ATA_LT	-1.068353
PORT_SIZE	-0.495673
CI_HOUR	48.252184
dtype: float64	

# 03 데이터 전처리





1. 데이터 확인
2. 이상치 확인
3. 다중공산성 확인
4. 파생 변수 생성
5. 범주형 변수 인코딩
6. 불필요 데이터 제거
7. 결측치 처리

## 03 중복값 확인

---

```
[ ] #중복된 행 확인  
print(train.duplicated().sum())  
print(test.duplicated().sum())
```

0

0

중복된 행 없음

## 03 통계치 확인

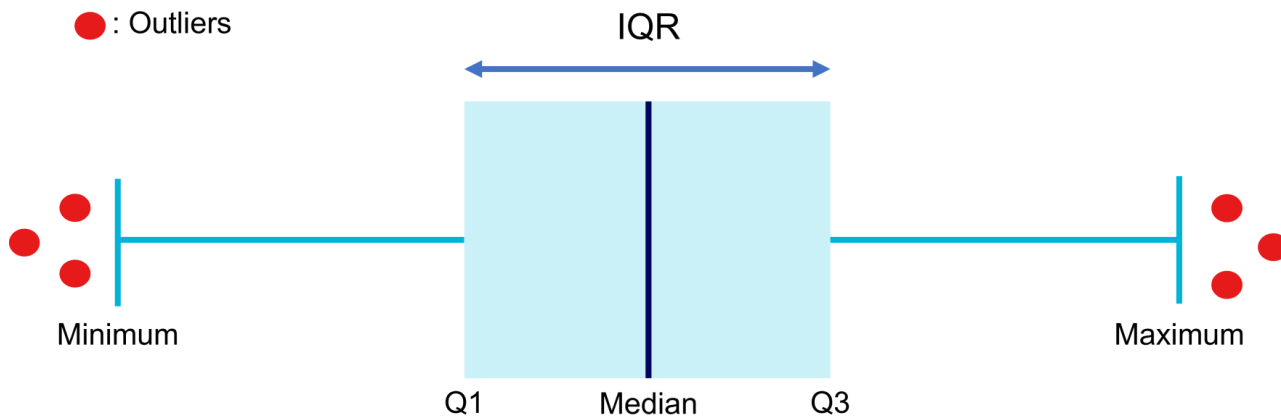
✓ 0초 train.describe().transpose()

	count	mean	std	min	25%	50%	75%	max
DIST	391939.0	19.328187	30.930059	0.000000	0.000000	6.929678	25.692827	199.980651
BREADTH	391938.0	30.550010	10.957070	10.000000	20.000000	30.000000	40.000000	60.000000
BUILT	391939.0	14.747989	7.043988	0.000000	10.000000	13.000000	18.000000	80.000000
DEADWEIGHT	391939.0	63577.506780	63515.531024	100.000000	14300.000000	47400.000000	81500.000000	404000.000000
DEPTH	391938.0	16.476177	6.045930	0.000000	10.000000	20.000000	20.000000	30.000000
DRAUGHT	391938.0	11.088106	4.599298	0.000000	10.000000	10.000000	10.000000	20.000000
GT	391939.0	41348.968666	39304.543664	150.000000	10000.000000	30700.000000	52100.000000	237000.000000
LENGTH	391938.0	205.121320	75.873891	20.000000	150.000000	190.000000	260.000000	400.000000
U_WIND	228251.0	-0.294910	3.366585	-25.330000	-2.150000	-0.130000	1.540000	17.910000
V_WIND	228251.0	-0.263849	3.736949	-21.450000	-2.280000	0.000000	2.000000	29.690000
AIR_TEMPERATURE	227309.0	18.862968	9.729976	-32.200000	12.200000	21.600000	26.600000	47.600000
BN	228251.0	2.706992	1.388026	0.000000	1.805246	2.618063	3.556236	11.179660
ATA_LT	391939.0	11.841580	6.644114	0.000000	7.000000	12.000000	17.000000	23.000000
PORT_SIZE	391939.0	0.000876	0.000838	0.000005	0.000142	0.000552	0.001614	0.002615
CI_HOUR	391939.0	61.940835	170.809558	0.000000	0.000000	7.944444	49.176111	2159.130556



## 03 이상치 확인

### 사분범위(InterQuartile Range, IQR)



$$Minimum = Q1 - (IQR * 1.5)$$

$$Maximum = Q3 + (IQR * 1.5)$$

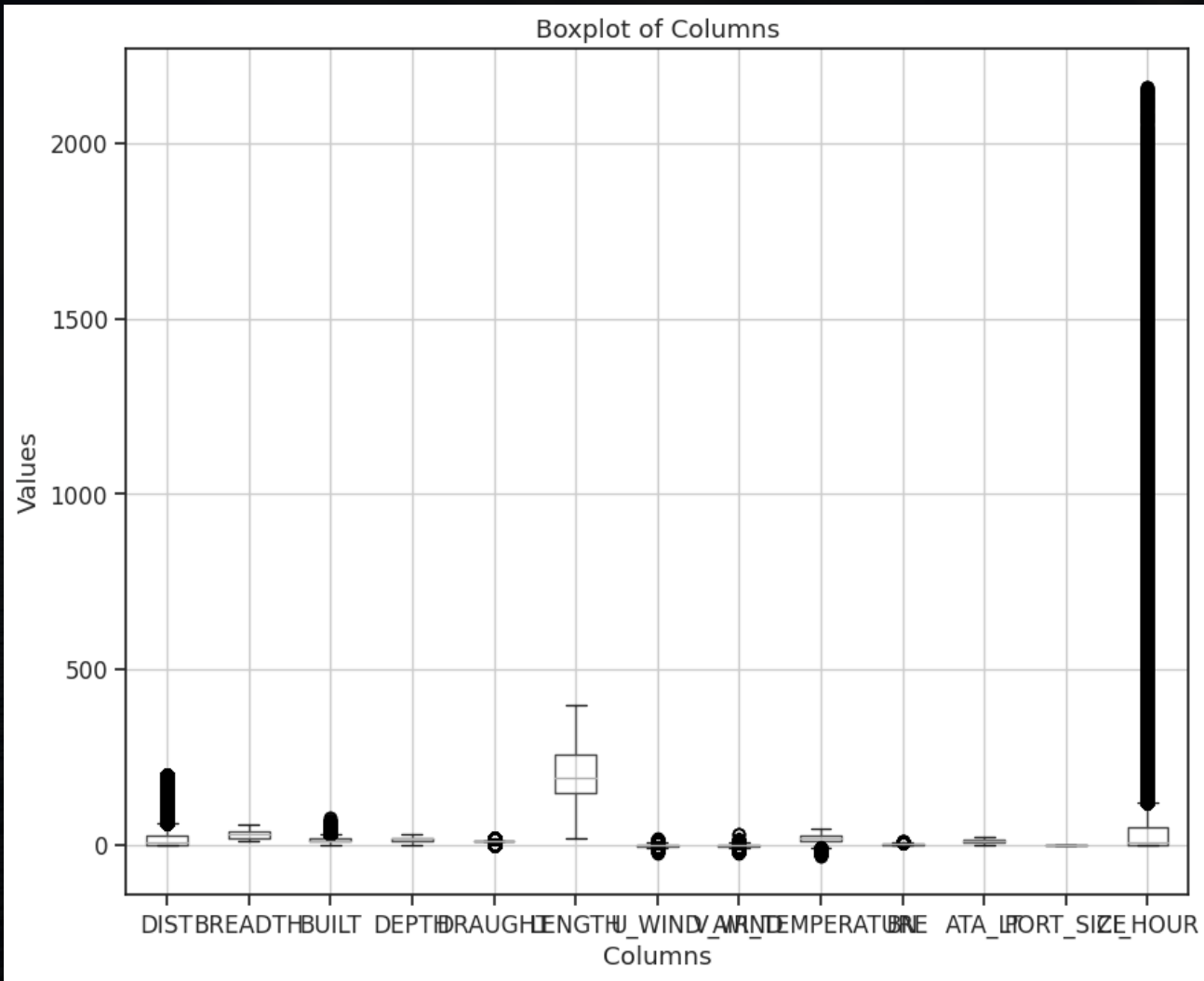
Minimum보다 작거나 Maximum보다 크면 이상치로 판단

A dark, atmospheric photograph of a large ship at night. The ship's hull is visible, with some lights reflecting on the water. The word 'DEADWE' is prominently displayed in large, white, serif capital letters across the upper right portion of the image. The overall mood is somber and mysterious.



'GT'

## 03 이상치 확인



'CI\_HOUR'





## 03 데이터 전처리

Column: DIST  
Outlier Count: 33820  
Outlier Ratio: 0.08628893781940557

Column: DRAUGHT  
Outlier Count: 87549  
Outlier Ratio: 0.22337404545095027

Column: BUILT  
Outlier Count: 9872  
Outlier Ratio: 0.02518759296727297

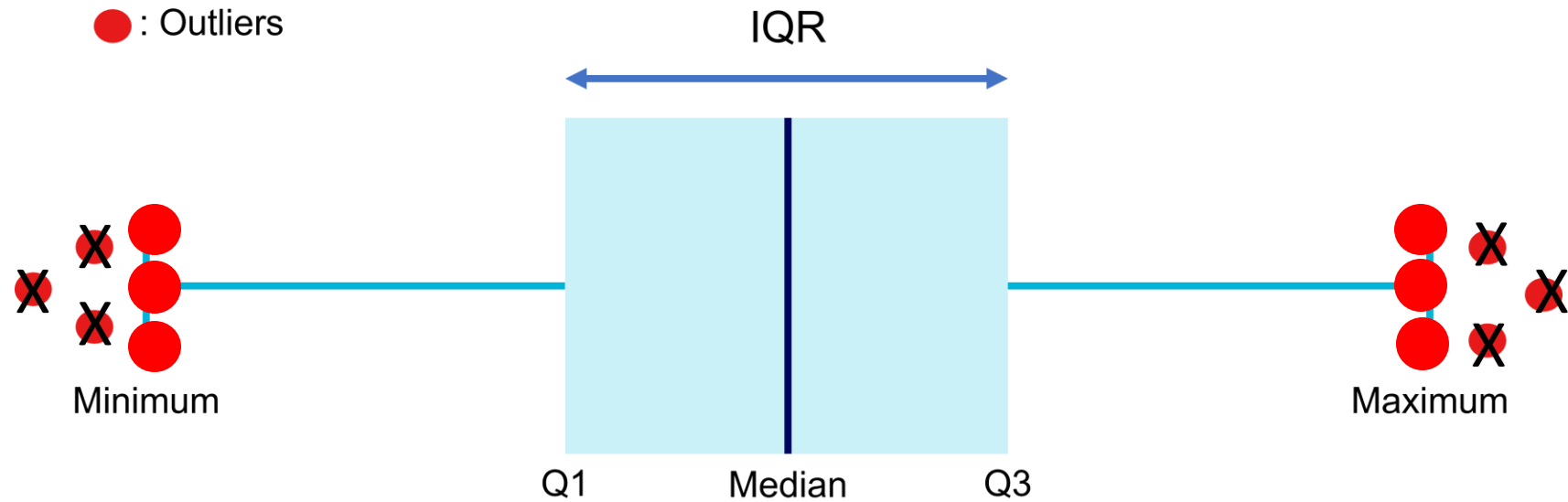
Column: GT  
Outlier Count: 18288  
Outlier Ratio: 0.04666032214196597

Column: DEADWEIGHT  
Outlier Count: 22899  
Outlier Ratio: 0.05842490795761585

Column: CI\_HOUR  
Outlier Count: 48820  
Outlier Ratio: 0.12456019941878711

수치형 데이터 15개 중 10개

### 03 이상치 처리 방법



이상치를 상한값과 하한값으로 치환

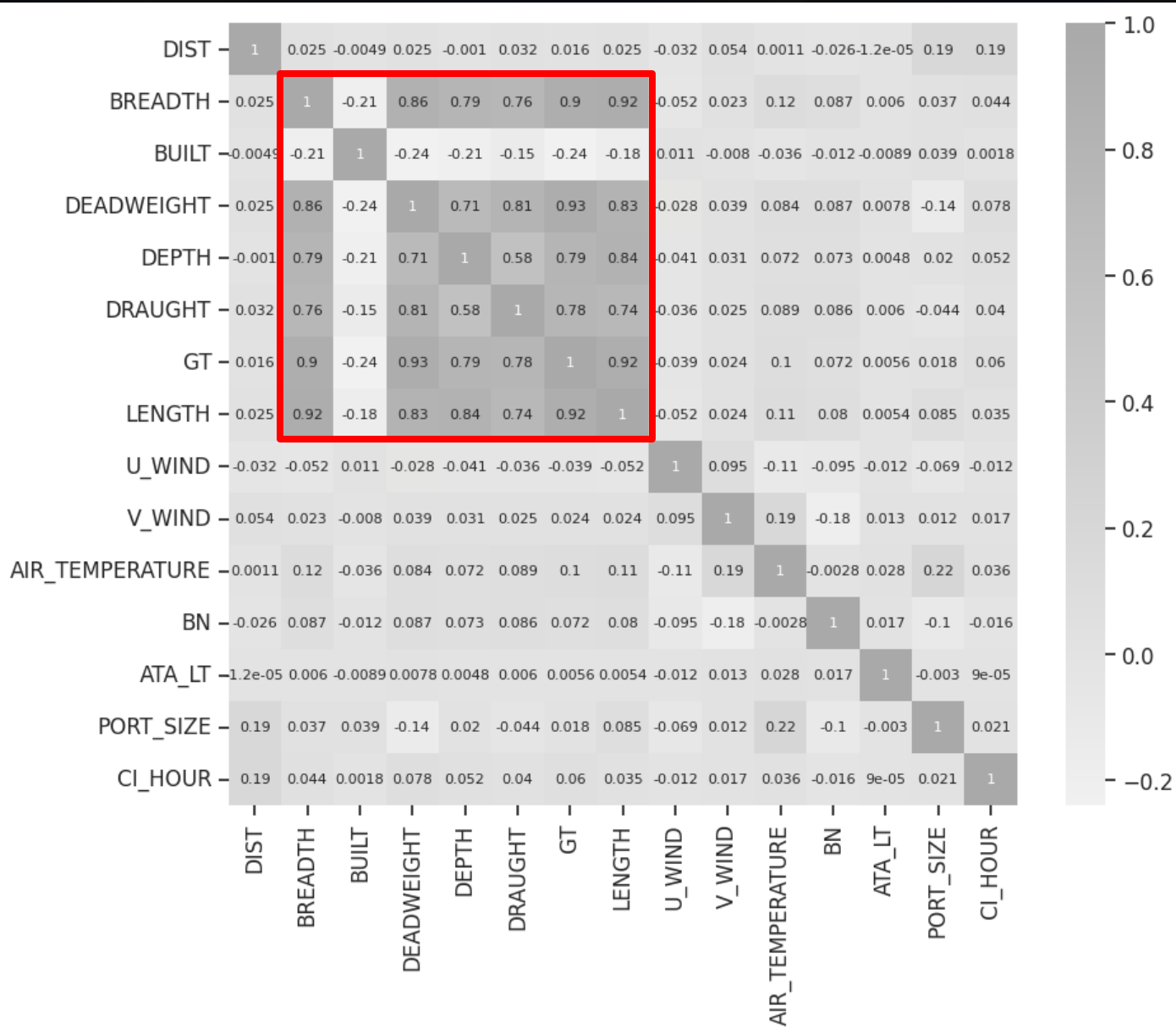
가장 마지막에



# 03 산점도 행렬 시각화

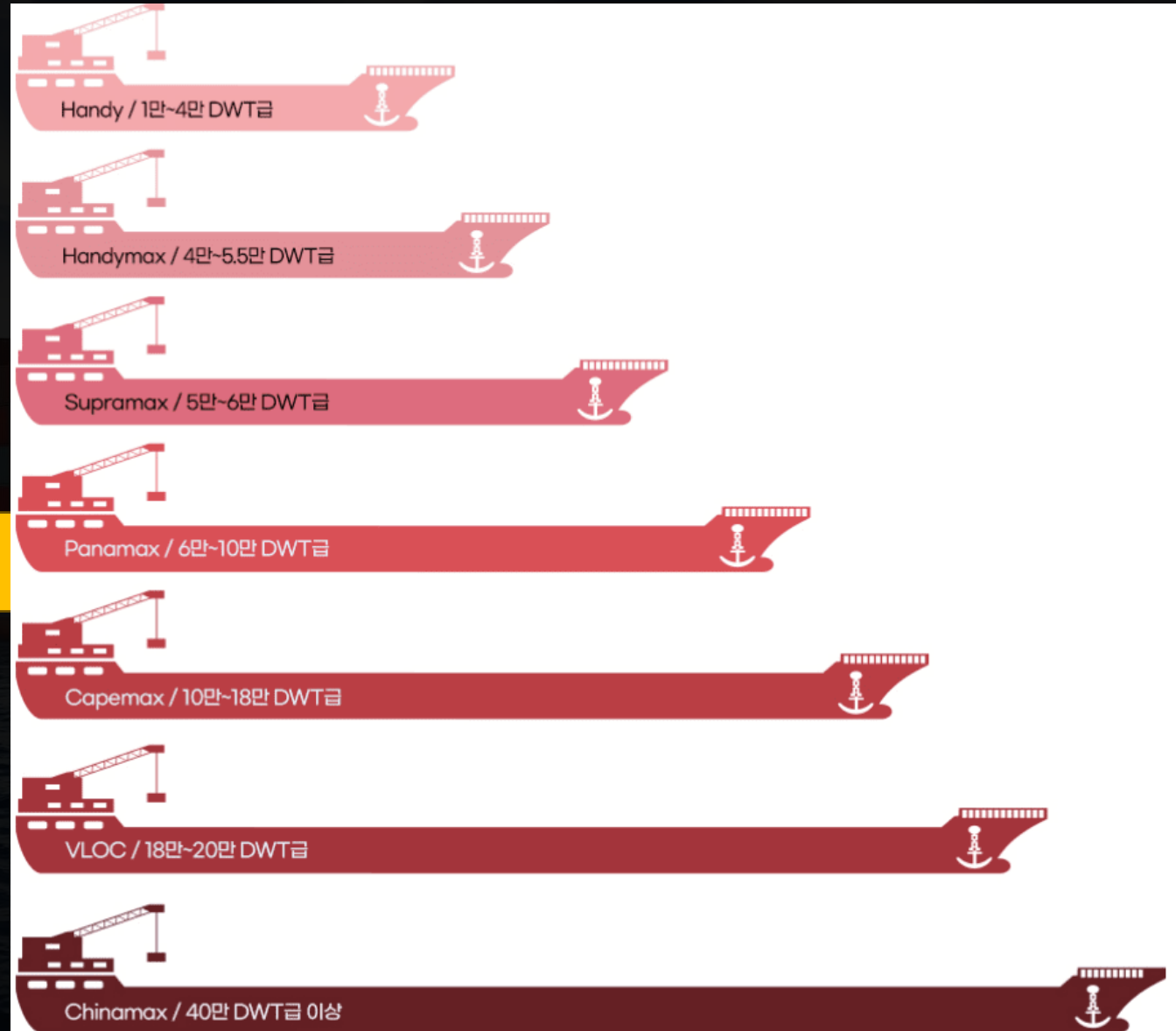


## 03 상관관계를 이용한 다중공산성 확인





‘선박의 크기’  
‘재화종류’  
‘선박의 길이’  
‘항수’  
‘용적톤수’  
‘선박의 길이’





## 03 데이터 전처리

LENGTH	-0.025	0.92	-0.18	0.83	0.84	0.74	0.92	1	-0.052	0.024
U_WIND	-0.032	-0.052	0.011	-0.028	-0.041	-0.036	-0.039	-0.052	1	0.095
V_WIND	-0.054	0.023	-0.008	0.039	0.031	0.025	0.024	0.024	0.095	1
AIR_TEMPERATURE	-0.0011	0.12	-0.036	0.084	0.072	0.089	0.1	0.11	-0.11	0.19
BN	-0.026	0.087	-0.012	0.087	0.073	0.086	0.072	0.08	-0.095	-0.18
ATA_LT	-1.2e-05	0.006	-0.0089	0.0078	0.0048	0.006	0.0056	0.0054	-0.012	0.013
PORT_SIZE	0.19	0.037	0.039	-0.14	0.02	-0.044	0.018	0.085	-0.069	0.012
CI_HOUR	0.19	0.044	0.0018	0.078	0.052	0.04	0.06	0.035	-0.012	0.017
	DIST	BREADTH	BUILT	DEADWEIGHT	DEPTH	DRAUGHT	GT	LENGTH	U_WIND	V_WIND

‘DEADWEIGHT’를 제외한  
나머지 변수 제거

## 03 파생 변수 변환

‘ATA’

ATA
2018-12-17 21:29
2014-09-23 6:59
2015-02-03 22:00
2020-01-17 4:02
2020-01-26 7:51



year	month	day	hour	minute	weekday
2018	12	17	21	29	0
2014	9	23	6	59	1
2015	2	3	22	0	1
2020	1	17	4	2	4
2020	1	26	7	51	6

‘year’

‘month’

‘day’

‘hour’

‘minute’

‘weekday’

## 03 범주형 변수 인코딩

	ARI_CO	ARI_PO	SHIP_TYPE_CATEGORY
0	SG	GIW5	Container
1	IN	UJM2	Bulk
2	CN	EUC8	Container
3	JP	ZAG4	Container
4	SG	GIW5	Container



	ARI_CO	ARI_PO	SHIP_TYPE_CATEGORY
0	17	21	2
1	7	81	0
2	4	14	2
3	8	101	2
4	17	21	2



# 03 데이터 전처리

BN	BN
NaN	3.0
NaN	3.0
NaN	3.0
2.629350	3.0
2.495953	2.0

결측치 채우고, 반올림하여 정수로 변환

계급	명칭	지상 10m에서의		육상상태
		상당풍속 (m/s)	상당풍속 (km/h)	
0	고요	0-02	<1	연기가 수직으로 올라감
1	실바람	03-15	1-5	풍향은 연기가 날리는 것으로 알 수 있으나, 풍향계는 움직이지 않음
2	남실바람	16-33	6-11	바람이 얼굴에 느껴짐 나뭇잎이 흔들리며 깃발이 가볍게 날림
3	산들바람	34-54	12-19	나뭇잎과 가는 가지가 끊임없이 흔들리고 깃발이 가볍게 날림
4	건들바람	55-79	20-28	먼지가 일고 종잇조각이 날리며 작은 가지가 흔들림
5	흔들바람	80-107	29-38	잎이 무성한 작은 나무 전체가 흔들리고 호수에 물결이 일어남
6	튀바람	108-138	39-49	큰 나뭇가지가 흔들리고 전선이 울리며 우산받기가 곤란함
7	센바람	139-171	50-61	나무 전체가 흔들리며, 바람을 안고서 걸기가 어려움
8	큰바람	172-207	62-74	작은 나뭇가지가 꺾이며, 바람을 안고서는 걸을 수가 없음
9	큰센바람	208-244	75-88	가옥에 다소 손해가 있음
10	노대바람	245-284	89-102	내륙 지방에서는 보기 드문 현상임 수목이 뿌리채 뽑히고 가옥에 큰 손해가 일어남
11	왕바람	285-326	103-117	이런 현상이 생기는 일은 거의 없음 광범위한 파괴가 일어남
12	삭쓸바람	327-	118-	-

보퍼트 풍력 계급

## 03 불필요 데이터 제거

'ATA'

'ID'(선박식별 일련번호)

'FLAG'(선박의 국적)

'SHIPMANAGER'(선박 소유주)

'U\_WIND'

'V\_WIND'

= 'BN'(보퍼트 풍력 계급)



제거

## 03 결측치 처리

	Train	Test
ARI_CO	0	0.0
ARI_PO	0	0.0
SHIP_TYPE_CATEGORY	0	0.0
DIST	0	0.0
ATA	0	0.0
ID	0	0.0
BREADTH	1	0.0
BUILT	0	0.0
DEADWEIGHT	0	0.0
DEPTH	1	0.0
DRAUGHT	1	0.0
GT	0	0.0
LENGTH	1	0.0
SHIPMANAGER	0	0.0
FLAG	0	0.0
U_WIND	163688	91725.0
V_WIND	163688	91725.0
AIR_TEMPERATURE	164630	92246.0
BN	163688	91725.0
ATA_LT	0	0.0
PORT_SIZE	0	0.0
CI_HOUR	0	NaN



	Train	Test
ARI_CO	0	0.0
ARI_PO	0	0.0
SHIP_TYPE_CATEGORY	0	0.0
DIST	0	0.0
BUILT	0	0.0
DEADWEIGHT	0	0.0
AIR_TEMPERATURE	0	0.0
BN	0	0.0
ATA_LT	0	0.0
PORT_SIZE	0	0.0
CI_HOUR	0	NaN
year	0	0.0
month	0	0.0
day	0	0.0
hour	0	0.0
minute	0	0.0
weekday	0	0.0



# 04 모델링 및 결과



## 04 회귀분석 진행(선형, 다항, 다중)

---



회귀모델 사용(선형, 다항,  
다중)



# 04 선형회귀, 다항회귀, 다중 선형회귀 진행

```
1 # 독립 변수와 종속 변수 분리
2 X = pre_processed_train.drop(columns='CI_HOUR') # 독립 변수 선택
3 y = pre_processed_train['CI_HOUR'] # 종속 변수 선택
4
5 # 데이터 분할 (학습 데이터와 테스트 데이터로)
6 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
7
8 # 선형 회귀 모델
9 linear_model = LinearRegression()
10 linear_model.fit(X_train, y_train)
11 y_pred_linear = linear_model.predict(X_test)
12
13 # 다항 회귀 모델
14 poly_degree = 3 # 다항 차수 선택
15 poly_features = PolynomialFeatures(degree=poly_degree)
16 X_poly = poly_features.fit_transform(X_train)
17 poly_model = LinearRegression()
18 poly_model.fit(X_poly, y_train)
19 X_test_poly = poly_features.transform(X_test)
20 y_pred_poly = poly_model.predict(X_test_poly)
21
22 # 다중 선형 회귀 모델
23 multi_linear_model = LinearRegression()
24 multi_linear_model.fit(X_train, y_train)
25 y_pred_multi_linear = multi_linear_model.predict(X_test)
26
27 # 각 모델의 성능 평가
28 linear_mse = mean_squared_error(y_test, y_pred_linear)
29 linear_r2 = r2_score(y_test, y_pred_linear)
30 print(f"Linear Regression - MSE: {linear_mse}, R-squared: {linear_r2}")
31
32 poly_mse = mean_squared_error(y_test, y_pred_poly)
33 poly_r2 = r2_score(y_test, y_pred_poly)
34 print(f"Polynomial Regression - MSE: {poly_mse}, R-squared: {poly_r2}")
35
36 multi_linear_mse = mean_squared_error(y_test, y_pred_multi_linear)
37 multi_linear_r2 = r2_score(y_test, y_pred_multi_linear)
38 print(f"Multiple Linear Regression - MSE: {multi_linear_mse}, R-squared: {multi_linear_r2}")
```

Linear Regression - MSE: 1334.3891168494767, R-squared: 0.29344512563592184  
Polynomial Regression - MSE: 1021.6274877725466, R-squared: 0.4590514324829841  
Multiple Linear Regression - MSE: 1334.3891168494767, R-squared: 0.29344512563592184



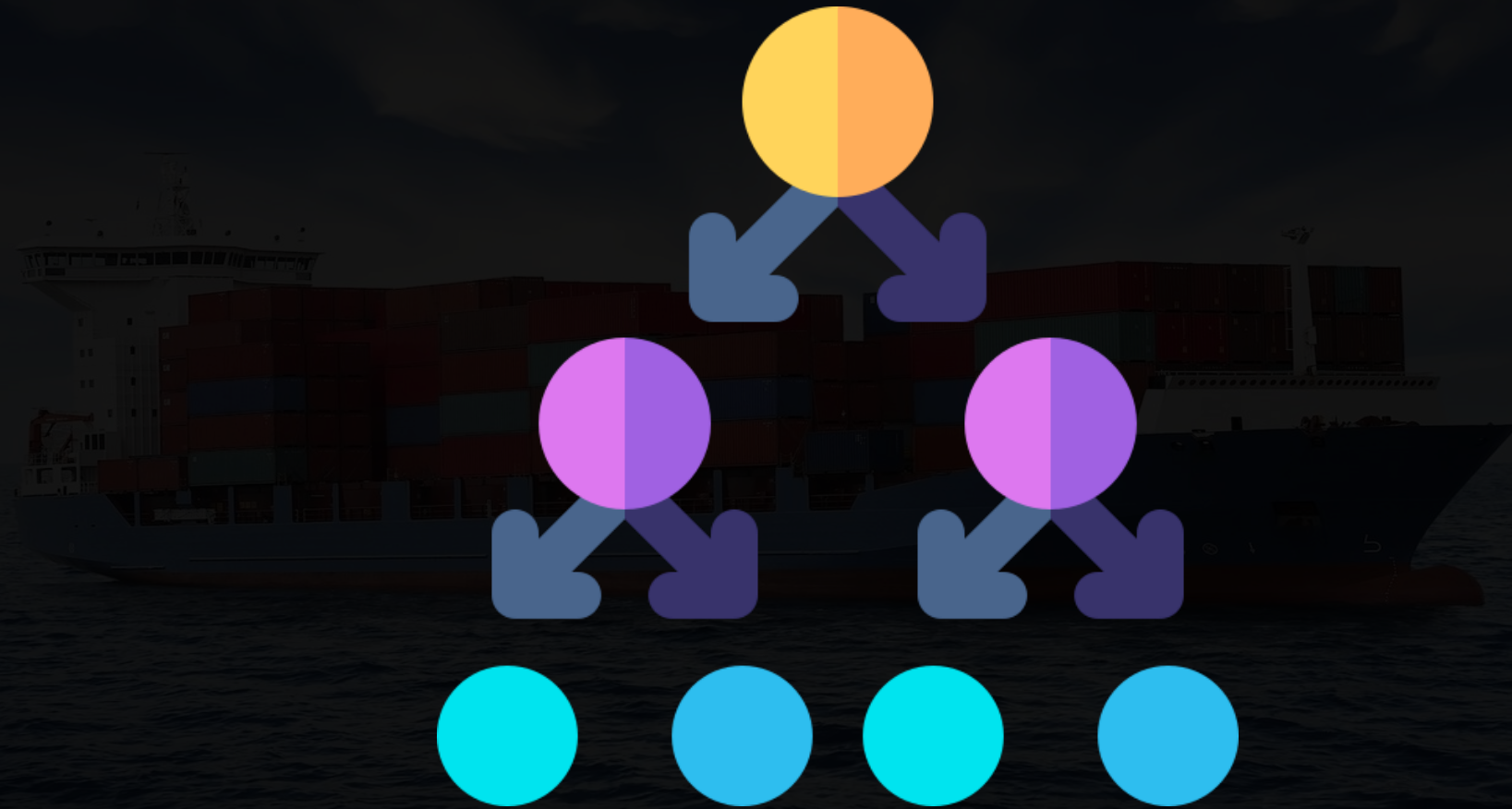
## 04 선형회귀, 다항회귀, 다중 선형회귀 진행

Model	MSE	R-squared
Linear Regression	1334.39	0.29
Polynomial Regression	1021.627	0.46
Multiple Linear Regression	1334.39	0.29

==> 결과값 상태가 많이  
==> 안 좋으 모델 말고 다른 모델  
사용

## 04 랜덤포레스트

---



랜덤포레스트 모델  
사용

## 04 랜덤포레스트

```
# 독립 변수와 종속 변수 분리
X = pre_processed_train.drop(columns="CI_HOUR") # 독립 변수 선택
y = pre_processed_train['CI_HOUR'] # 종속 변수 선택

# 데이터 분할 (학습 데이터와 테스트 데이터로)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 랜덤 포레스트 모델
rf_model = RandomForestRegressor()

# 탐색할 하이퍼파라미터 범위 설정
param_dist = {
    "n_estimators": [10, 50, 100], # 나무의 수
    "max_depth": [None, 10, 20, 30], # 나무의 깊이
    "min_samples_split": [2, 5, 10], # 노드 분할을 위한 최소 샘플 수
    "min_samples_leaf": [1, 2, 4], # 리프 노드에 필요한 최소 샘플 수
    "max_features": ["auto", "sqrt", "log2"], # 분할할 때 고려할 특성의 수
}
```



## 04 랜덤포레스트

```
# 랜덤 서치를 사용한 모델 튜닝
random_search = RandomizedSearchCV(
    | rf_model, param_distributions=param_dist, n_iter=30, cv=5, n_jobs=-1, verbose=2
)
random_search.fit(X_train, y_train)

# 최적의 하이퍼파라미터 조합
best_params = random_search.best_params_
print("Best Hyperparameters:", best_params)

# 튜닝된 모델
tuned_rf_model = random_search.best_estimator_

# 모델 평가
y_pred_rf_tuned = tuned_rf_model.predict(X_test)
rf_tuned_mse = mean_squared_error(y_test, y_pred_rf_tuned)
rf_tuned_r2 = r2_score(y_test, y_pred_rf_tuned)
print(f"Random Forest Regression (Tuned) - MSE: {rf_tuned_mse}, R-squared: {rf_tuned_r2}")
```

성능을 높이기 위해  
모델 튜닝 과정  
진행

## 04 랜덤포레스트 모델 - 랜덤서치

```
# 랜덤 서치를 사용한 모델 튜닝
random_search = RandomizedSearchCV(
    rf_model, param_distributions=param_dist, n_iter=30, cv=5, n_jobs=-1, verbose=2
)
random_search.fit(X_train, y_train)

# 최적의 하이퍼파라미터 조합
best_params = random_search.best_params_
print("Best Hyperparameters:", best_params)

# 튜닝된 모델
tuned_rf_model = random_search.best_estimator_

# 모델 평가
y_pred_rf_tuned = tuned_rf_model.predict(X_test)
rf_tuned_mse = mean_squared_error(y_test, y_pred_rf_tuned)
rf_tuned_r2 = r2_score(y_test, y_pred_rf_tuned)
print(f"Random Forest Regression (Tuned) - MSE: {rf_tuned_mse}, R-squared: {rf_tuned_r2}")
```

## 04 랜덤포레스트 결과

```
Fitting 5 folds for each of 30 candidates, totalling 150 fits
C:\Users\user\Desktop\portTerminalTimeDelay prediction\venv\Lib\site-packages\sklearn\model_selection\_validation.py:425: FitFailedWarning:
55 fits failed out of a total of 150.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
-----
55 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\user\Desktop\portTerminalTimeDelay prediction\venv\Lib\site-packages\sklearn\model_selection\_validation.py", line 729, in _fit
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\user\Desktop\portTerminalTimeDelay prediction\venv\Lib\site-packages\sklearn\base.py", line 1145, in wrapper
    estimator._validate_params()
  File "C:\Users\user\Desktop\portTerminalTimeDelay prediction\venv\Lib\site-packages\sklearn\base.py", line 638, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\user\Desktop\portTerminalTimeDelay prediction\venv\Lib\site-packages\sklearn\utils\_param_validation.py", line 95, in validate
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of RandomForestRegressor must be an int in the range [1,

warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\user\Desktop\portTerminalTimeDelay prediction\venv\Lib\site-packages\sklearn\model_selection\_search.py:979: UserWarning: One or more
nan 0.53162458      nan 0.57882883      nan 0.56129335
0.58251002 0.58309839      nan      nan 0.5812704 0.55550852
0.57852521 0.56404499 0.57953929      nan 0.56131202 0.53718509
      nan      nan      nan      nan 0.5834929 0.58027044]

warnings.warn(
Best Hyperparameters: {'n_estimators': 100, 'min_samples_split': 2, 'min_samples_leaf': 4, 'max_features': 'log2', 'max_depth': 30}
Random Forest Regression (Tuned) - MSE: 770.5355977967733, R-squared: 0.592003804872337
```

Best Hyperparameters: {'n\_estimators': 100, 'min\_samples\_split': 2, 'min\_samples\_leaf': 4, 'max\_features': 'log2', 'max\_depth': 30}  
Random Forest Regression (Tuned) - MSE: 770.5355977967733, R-squared: 0.592003804872337

0.55551



0.5920



## 04 랜덤포레스트 결과

Model	MSE	R-squared
Linear Regression	1334.39	0.29
Polynomial Regression	1021.627	0.46
Multiple Linear Regression	1334.39	0.29
Random Forest	770.53	0.59

➡ Random Forest로 성능 향상

## 04 테스트 데이터셋의 대기시간 예측 후 저장

```
# 이전에 튜닝한 최적의 랜덤 포레스트 모델
tuned_rf_model = RandomForestRegressor(
    n_estimators=100, # 예시로 설정된 하이퍼파라미터
    min_samples_split=5, # 예시로 설정된 하이퍼파라미터
    min_samples_leaf=2, # 예시로 설정된 하이퍼파라미터
    max_features='log2', # 예시로 설정된 하이퍼파라미터
    max_depth=None # 예시로 설정된 하이퍼파라미터
)

# 학습 데이터로 모델 학습
tuned_rf_model.fit(X_train, y_train) # 여기서 X_train, y_train은 학습 데이터와 해당 데이터의 대기시간

X = pre_processed_test # test 데이터셋

# 테스트 데이터로 예측
y_pred_test = tuned_rf_model.predict(X)

# 테스트 데이터 예측값 확인
pred_CI_HOUR = pd.DataFrame(y_pred_test, columns=['CI_HOUR'])
```

pred_CI_HOUR	
CI_HOUR	
0	41.265550
1	38.468911
2	57.006649
3	64.437459
4	0.000000
...	...
220486	68.796555
220487	54.072873
220488	44.364489
220489	0.454487
220490	74.176460
220491 rows × 1 columns	