차량 수리비용 예측

Redhands

3팀

에이콘아카데미 강남

contents

1. 주제 및 기획의도

[주제, 기획의도]

본 프로젝트는 자동차 사고 시 발생하는 다양한 유형의 사고 이미지를 분석하여 손상 유형,
손상 부위 등을 도출하고 이를 기반으로 차량의 예상 수리시간 및 비용을 예측하는 목적을 가지고 있다.
전반적인 자동차 관련사업에서 활용될 것으로 기대되며, 차량부품에 관한 지식이 없는 소비자들도
빠르게 수리비용 견적을 알려주는 서비스를 제공할 수 있을 것으로 기대된다.

HYUNDAI MOBIS

Bluehands

meritz 메리츠화재

KIA MOTORS

KB 손해보험 | 다이렉트

현대해상 다이렉트

활용 장비 및 재료

[개별 환경]

Language/Library

python
jupyter
pandas
NumPy
ggplot2
CO PRO
PC
R Studio
TensorFlow
YOLOv5
K Keras
MariaDB

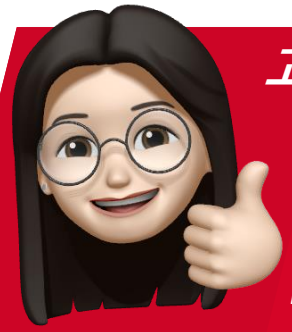
Peristalsis

GitHub
Jira
PowerPoint
Visual Studio Code

Source

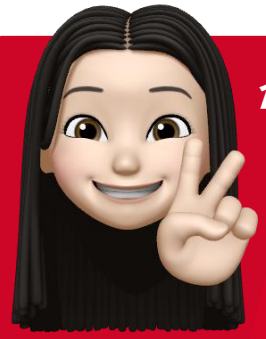
과학기술정보통신부
NIA 한국지능정보사회진흥원 NATIONAL INFORMATION SOCIETY AGENCY
HYUNDAI MOBIS
AI Hub
SAMSUNG 르노삼성자동차
Bluehands
KIA MOTORS

2. 업무 분담



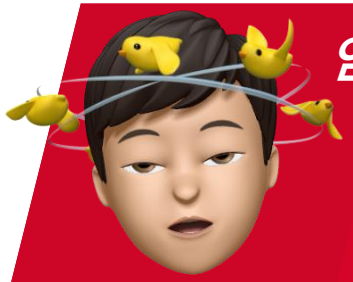
고은경
데이터 수집, 전처리
데이터 분석
통계검증
ML모델학습

김도현
데이터 수집, 전처리
데이터 분석
DL/ML모델학습
웹페이지제작



김이경
데이터 수집, 전처리
데이터 분석
모델학습

김현욱
데이터 수집, 전처리
DB 제작
통계검증
DL/ML모델학습

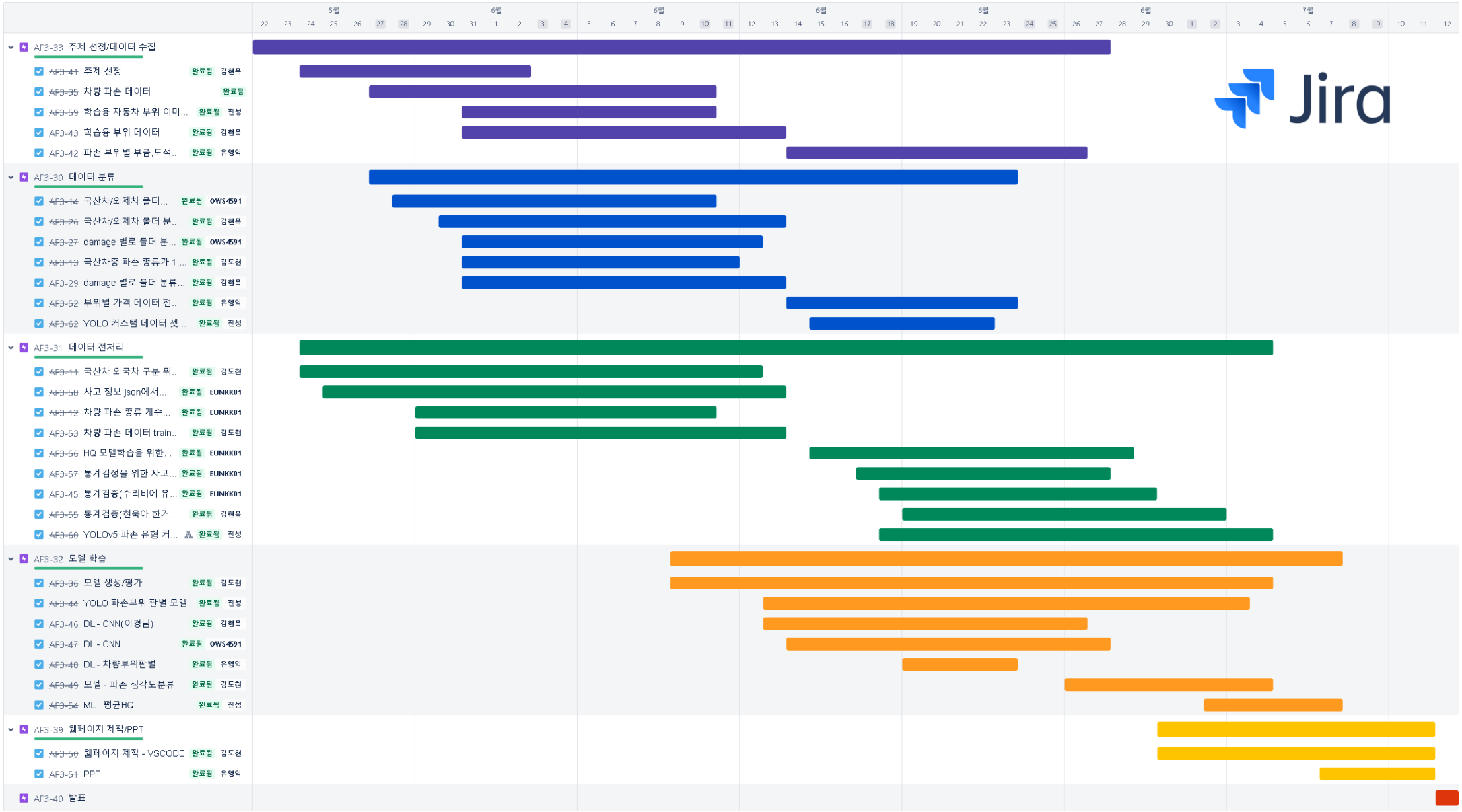


엄진성
데이터 수집, 전처리
데이터 분석
YOLO 모델 학습
웹페이지 제작

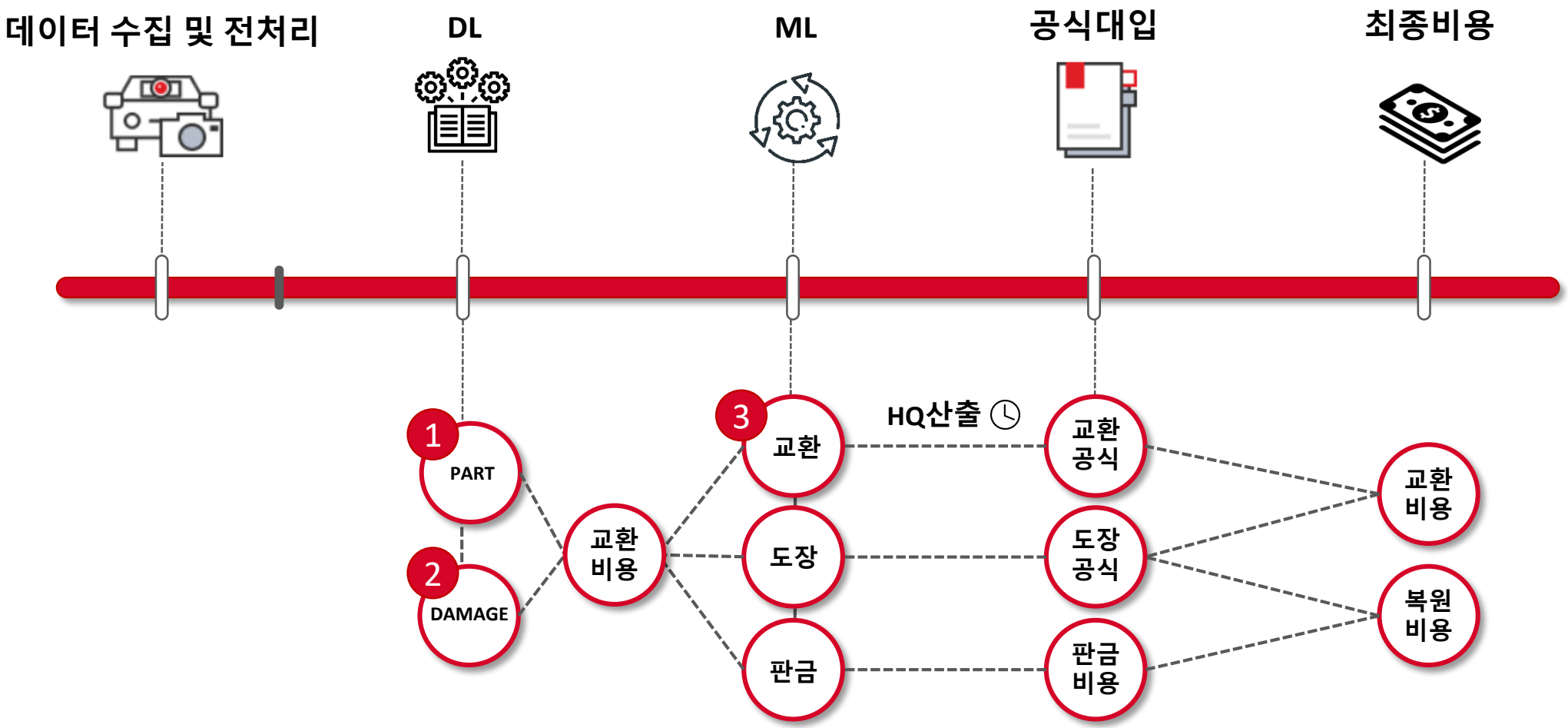
오원석
데이터 수집, 전처리
데이터 분석
DL모델학습
발표

유영익
데이터 수집, 전처리
데이터 분석
DB 제작
PPT

3. 일정, 계획


5월
6월
6월
6월
6월
6월
7월
AF3-33 주제 선정/데이터 수집
AF3-41 주제 선정 완료됨 김현욱
AF3-35 차량 파손 데이터 완료됨
AF3-59 학습용 자동차 부위 이미... 완료됨 진성
AF3-43 학습용 부위 데이터 완료됨 김현욱
AF3-42 파손 부위별 부품,도색... 완료됨 유영익
AF3-30 데이터 분류
AF3-14 국산차/외제차 폴더... 완료됨 OWS4591
AF3-26 국산차/외제차 폴더 분... 완료됨 김현욱
AF3-27 damage 별로 폴더 분... 완료됨 OWS4591
AF3-13 국산차중 파손 종류가 1,... 완료됨 김도현
AF3-29 damage 별로 폴더 분류... 완료됨 김현욱
AF3-52 부위별 가격 데이터 전... 완료됨 유영익
AF3-62 YOLO 커스텀 데이터 셋... 완료됨 진성
AF3-31 데이터 전처리
AF3-11 국산차 외국차 구분 위... 완료됨 김도현
AF3-58 사고 정보 json에서... 완료됨 EUNKK01
AF3-12 차량 파손 종류 개수... 완료됨 EUNKK01
AF3-53 차량 파손 데이터 train... 완료됨 김도현
AF3-56 HQ 모델학습을 위한... 완료됨 EUNKK01
AF3-57 통계검정을 위한 사고... 완료됨 EUNKK01
AF3-45 통계검증(수리비에 유... 완료됨 EUNKK01
AF3-55 통계검증(현욱아 한거... 완료됨 김현욱
AF3-60 YOLOv5 파손 유형 커... 완료됨 진성
AF3-32 모델 학습
AF3-36 모델 생성/평가 완료됨 김도현
AF3-44 YOLO 파손부위 판별 모델 완료됨 진성
AF3-46 DL - CNN(이경님) 완료됨 김현욱
AF3-47 DL - CNN 완료됨 OWS4591
AF3-48 DL - 차량부위판별 완료됨 유영익
AF3-49 모델 - 파손 심각도분류 완료됨 김도현
AF3-54 ML - 평균HQ 완료됨 진성
AF3-39 웹페이지 제작/PPT
AF3-50 웹페이지 제작 - VSCODE 완료됨 김도현
AF3-51 PPT 완료됨 유영익
AF3-40 발표
Jira

3. 일정, 계획


데이터 수집 및 전처리
DL
ML
공식대입
최종비용
1
PART
2
DAMAGE
교환
비용
3
교환
도장
판금
HQ산출
교환
공식
도장
공식
판금
비용
교환
비용
복원
비용

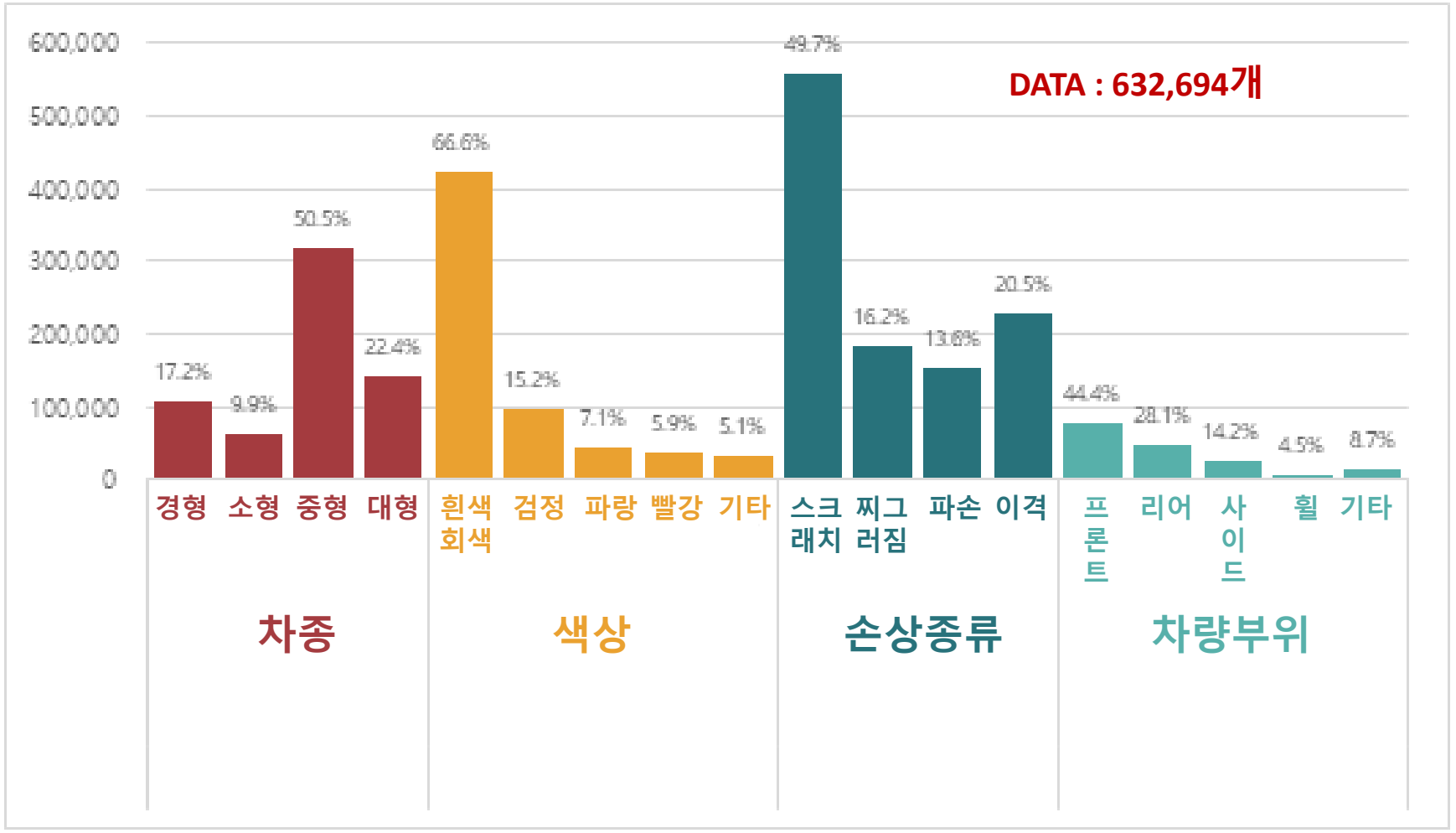
데이터 수집

A. 데이터 수집

[1] 차량 파손 데이터

AI허브 홈페이지를 통하여 자동차 사고 발생시 다양한 유형의 사고 이미지로 이루어진 데이터를 수집하여 이를 이용한 차량 파손 여부 식별 프로세스 개발

AI Hub



A. 데이터 수집

[2] 모델 학습용 자동차 부위 데이터

자동차 파손부위 판별 모델 학습을 위한 데이터셋, Train데이터로 사용하기위해 불량품 제거.

AI Hub

데이터 분류		품질상태			비율
대분류(부품)	소분류(불량유형)	양품	불량품	합계	
도어	스크래치	4,000	6,000	10,000	5.0%
	외관 손상	3,000	3,000	6,000	3.0%
라디에이터 그릴	단차	3,000	3,000	6,000	3.0%
루프사이드	장착 불량	6,000	6,000	12,000	6.0%
배선	고정 불량	6,000	6,000	12,000	6.0%
범퍼	스크래치	4,000	10,000	14,000	7.0%
카울커버	고정핀 불량	6,000	6,000	12,000	6.0%
	연계 불량	6,000	6,000	12,000	6.0%
커넥터	유격 불량	6,000	9,000	15,000	7.5%
	체결 불량	6,000	12,000	18,000	9.0%
테일 램프	단차	7,000	7,000	14,000	7.0%
프레임	외관 손상	3,000	4,000	7,000	3.5%
	실링 불량	2,000	4,000	6,000	3.0%
	헤밍 불량	5,000	5,000	10,000	5.0%
	홀 변형	4,000	4,000	8,000	4.0%
헤드 램프	단차	7,000	7,000	14,000	7.0%
휀더	외관 손상	4,000	4,000	8,000	4.0%
	단차	8,000	8,000	16,000	8.0%
합계		90,000	110,000	200,000	100.0%

A. 데이터 수집

[2] 모델 학습용 자동차 부위 데이터

학습데이터 부족으로 인한 크롤링 코드 작성 후 차량 부위 데이터 추가

```
driver = webdriver.Chrome("chromedriver")
driver.get("https://www.google.com")

element = driver.find_element(By.CLASS_NAME, "gLFyf")
element.send_keys("아반떼 후미등")                    # 검색할 키워드 여기에 넣으시면 됩니다.

element.send_keys(Keys.ENTER)

driver.find_element(By.CLASS_NAME, "zItAnd ").click()  # 이미지 클릭
```

```
# 스크롤 다 내렸을 때의 위치
bottom = driver.execute_script("return document.body.scrollHeight")

while True:
    # 맨 아래로 스크롤
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")

    # 화면을 맨 밑으로 내렸을 때, 이미지가 다 로드 될 수 있게 기다려주기
    time.sleep(3)

    # 맨 위가 되었다면, 다시 스크롤을 맨 밑으로 내려주는 과정을 반복해주어야 함.
    new_bottom = driver.execute_script("return document.body.scrollHeight")
    if new_bottom == bottom:    # 더 이상 내릴 부분이 없으면
        try:
            driver.find_element_by_css_selector(".mye4qd").click()  # 검색어 더

        except:
            break

    bottom = new_bottom   # 스크롤을 맨 밑으로 내렸을 때, 로딩이 끝나면 그 부분이
```

```
# 이미지의 클래스가 rg_i로 모두 동일함.
images = driver.find_elements(By.CLASS_NAME, "rg_i")

print(len(images))

count = 1
for i in range(len(images)):
    src = images[i].get_attribute('src')
    if src is not None:
        req.urlretrieve(src, "C:₩youyoung₩Final_Project₩Rear_lamp3/Rear_lamp3_{}.jpeg".format(count)) # 경로, 이미지 파일명
        # ex) 경로는 본인에게 맞게 조정, AI/____/____ 이 부분은 본인의 검색 키워드에 맞게 설정해줄 것.
        count += 1
    else:
        # 이미지 URL이 없는 경우가 존재 -> 이럴 땐 무시하고, 다음 사진으로 넘어가는 코드
        continue
```

```
import os

folder_path = "D:₩DATA₩Desktop₩Rear_lamp"  # 대상 폴더 경로
common_file_name  = "Rear_lamp_"    # 변경된 파일명의 공통 부분  -> rearfender_1 , rearfender_2 에서 rearfender 공통

file_list = os.listdir(folder_path)  # 폴더 내의 모든 파일 목록을 가져옴
file_list.sort()                     # 파일 목록을 오름차순으로 정렬

for i, filename in enumerate(file_list, start=1):
    new_filename = "{}_{}.jpeg".format(common_file_name,i)     # 새로운 파일명의 형태
    old_filepath = os.path.join(folder_path, filename)         # 원래 이름
    new_filepath = os.path.join(folder_path, new_filename)     # 바꿀 이름

    os.rename(old_filepath, new_filepath)  # 파일 이름 변경
```

A. 데이터 수집

[3-1] 차량 부위별 가격 데이터

차량 파손시 가격예측 계산에 필요한 공식적인 부품 가격 데이터 크롤링

HYUNDAI MOBIS

검색 방식
- (●) 일반검색
- ( ) 부품번호로 검색

제조사
- (●) 현대
- ( ) 기아

차량 구분
- (●) 승용/친환경
- ( ) SUV/RV
- ( ) 상용

모델: 그랜저 23

한글 부품명: 판넬 어셈블리

검색하기

부품번호	한글
64502N1000	판넬 어셈블리-

차량정보

제조사/모델	연식	연료	등급	색상코드/코트/색상	부위별예상수리비용
제네시스 G80	2020 ~ 현재	가솔린 외	전등급 참고	UYH/3코트/백색계열	

앞범퍼 교환/복원수리 예상수리비용

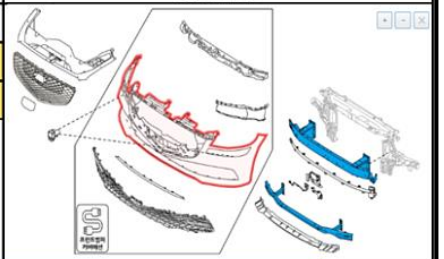
-한국 자동차 보상센터-

2018 국토부,보험개발원 표준정비요금 35,000 (작업시간*공임수가) (AOS)

구분	교환탈부착및O/H				교환도장				
	부품액	공임수가	작업시간	소계	공임	작업시간	소계	부수비용	합계
교환	160,000	35,000	2.21	77,350	35,000	2.03	71,050	190,769	499,169

구분	탈부착및O/H				그린수가(판금)			수리도장				
	부품액	공임수가	작업시간	소계	공임	작업시간	소계	공임수가	작업시간	소계	부수비용	합계
판금,수리		35,000	1.33	46,550	35,000	2.74	96,000	35,000	3.39	118,650	190,969	452,169

앞범퍼 부품번호 86550T1000



Kvcc 한국자동차보상센터 1577-5494

A. 데이터 수집

[3-2] 차량 부위별 가격 데이터

차량 파손시 가격예측 계산에 필요한 공식적인 부품 가격 DB 생성

HYUNDAI MOBIS

final_project.car_coating_price: 6 행 (총) (대략적)

다음 | 모두 보기 | 정렬 | 열 (15/15) | 필터

차량크기	Bumper	Bonnet	Roof	Head lights	Front door	Rear door	Front Fender	Rocker panel	Side mirror	Rear Fender	Trunk lid	Rear Bumper	Rear lamp	Wheel
경형	40100	37000	30800	0	31000	31000	33100	32340	0	30000	41200	38500	0	0
소형	47320	43680	36400	0	35945	36400	39130	38220	0	33670	48685	45500	0	0
중형	52000	48000	40000	0	39500	40000	43000	42000	0	37000	53500	50000	0	0
대형	56000	515000	44000	0	42500	43500	51000	44000	0	39500	43800	52500	0	0
SUV	55300	50800	44000	0	47800	43700	47700	44000	0	38700	43700	55700	0	0
VAN	56000	52000	44000	0	56000	44000	48000	44000	0	38000	43000	57000	0	0

final_project.car_part_price: 112 행 (총) (대략적)

다음 | 모두 보기 | 정렬 | 열 (18/18)

제조사	차이름	차량크기	연식	Bumper	Bonnet	Roof	Head lights	Front door	Rear door	Front Fender	Rocker panel	Side mirror	Rear Fender	Trunk lid	Rear Bumper	Rear lamp
기아	모닝	경형	8	63000	147000	137000	79000	165000	145000	51000	110000	11000	127000	140000	63000	46000
기아	모닝	경형	11	67000	158000	137000	81000	187000	167000	50000	215000	36000	200000	125000	67000	39000
기아	모닝	경형	15	67000	158000	137000	81000	187000	167000	50000	215000	37000	200000	125000	68000	39000
기아	모닝	경형	17	67000	160000	137000	109000	187000	154000	50000	187000	39000	220000	164000	63000	52000
기아	모닝	경형	20	69000	160000	137000	109000	187000	154000	50000	187000	39000	226000	164000	68000	52000
기아	k3	소형	12	120000	286000	150000	130000	300000	310000	90000	180000	94000	308000	300000	120000	110000
기아	k3	소형	16	120000	286000	150000	130000	390000	310000	90000	180000	94000	308000	300000	120000	120000
기아	k3	소형	18	110000	420000	170000	310000	390000	330000	90000	167000	120000	280000	324000	120000	120000
기아	k3	소형	21	130000	420000	170000	450000	390000	400000	90000	167000	120000	280000	324000	120000	120000
기아	k5	중형	11	126000	410000	180000	130000	336000	330000	88000	290000	88000	250000	370000	166000	140000
기아	k5	중형	14	116000	337000	180000	277000	336000	330000	88000	290000	127000	250000	310000	166000	130000
기아	k5	중형	16	140000	337000	170000	335000	336000	330000	88000	235000	110000	330000	338000	164000	121000
기아	k5	중형	18	140000	337000	170000	370000	336000	330000	88000	235000	110000	330000	338000	164000	121000
기아	k5	중형	20	140000	360000	170000	410000	383000	369000	115000	235000	120000	340000	279000	143000	146000
기아	k7	대형	8	125000	416000	179000	433000	422000	347000	110000	288000	140000	330000	339000	154000	130000
기아	k7	대형	13	130000	416000	179000	500000	422000	369000	110000	288000	134000	330000	339000	157000	159000
기아	k7	대형	16	155000	416000	179000	497000	422000	403000	132000	250000	134000	346000	398000	173000	152000
기아	k7	대형	19	155000	416000	179000	649000	422000	403000	132000	250000	200000	346000	398000	173000	155000
기아	k9	대형	11	133000	446000	205000	1235000	422000	336000	132000	544000	152000	484000	339000	157000	251000
기아	k9	대형	15	133000	446000	205000	123500	422000	404000	132000	545000	172000	484000	339000	157000	234000
기아	k9	대형	18	133000	430000	223000	892000	422000	336000	169000	256000	332000	573000	339000	112000	394000
기아	카니발	승합	14	122000	240000	248000	170000	280000	258000	110000	300000	124000	380000	440000	102000	160000
기아	카니발	승합	18	130000	240000	248000	170000	280000	258000	110000	300000	124000	380000	440000	102000	160000
기아	카니발	승합	20	130000	240000	248000	440000	3190000	258000	990000	300000	130000	380000	390000	102000	160000
현대	투싼	SUV	10	78000	268000	175000	118000	276000	275000	76000	159000	66000	288000	230000	89000	48000
현대	투싼	SUV	14	78000	268000	175000	190000	276000	275000	76000	159000	72000	288000	230000	89000	48000
현대	투싼	SUV	15	76000	268000	175000	250000	294000	256000	83000	159000	136000	382000	230000	76000	59000
현대	투싼	SUV	18	84000	268000	175000	179000	294000	256000	83000	159000	136000	382000	230000	760000	59000

final_project.car_color_price: 7 행 (총) (대략적)

color	price 1L	white contrast
화이트	68000	1.0
레드	136000	2.0
오렌지	128000	1.9
그레이	84800	1.2
블루	110000	1.6
실버	73000	1.1
블랙	60000	0.9

데이터 전처리

IMAGE, JSON

- 국산차/ 외제차 분류, 견적서가 존재하지 않는 이미지 삭제

- 견적서, 파손부위 JSON내 작업유형, 부품명 통일

- 이미지, 라벨 필터링, 중복 DAMAGE 삭제

- 모델 학습 중 성능 저하로 인하여 이미지 데이터 품질 검수

- 수리 방법(도장, 판금, 교환) 모두 통일하게 진행 파손 유형 및 부위별 HQ 분포 확인 후 이상치 제거

- 변수별 HQ를 BOXPLOT으로 시각화 이상치 확인 후 제거

B. 전처리

[1] 차량 부위별 가격 데이터

비용 null값 제거, 화폐단위 비용을 정수형으로 변경, 문자열로 지정된 손상부위와 수리방법 별도 추출

```
repair_price = repair_price.drop(repair_price[repair_price["total"].isnull()].index, axis=0)
```

```
#### "nan"값인 행 제거 ####
coating_null_list = repair_price[repair_price["coating"]=="nan"].index
repair_price = repair_price.drop(coating_null_list, axis=0)

sheet_metal_null_list = repair_price[repair_price["sheet_metal"]=="nan"].index
repair_price = repair_price.drop(sheet_metal_null_list, axis=0)

exchange_null_list = repair_price[repair_price["exchange"]=="nan"].index
repair_price = repair_price.drop(exchange_null_list, axis=0)

total_null_list = repair_price[repair_price["total"]=="nan"].index
repair_price = repair_price.drop(total_null_list, axis=0)

len(repair_price)
```

```
232714
```

```
# 화폐단위로 저장된 숫자를 정수형으로 변환
for i in tqdm(range(len(repair_price))):
    repair_price["coating"][i] = int(str(repair_price["coating"][i]).replace(",", ""))
    repair_price["sheet_metal"][i] = int(str(repair_price["sheet_metal"][i]).replace(",", ""))
    repair_price["exchange"][i] = int(str(repair_price["exchange"][i]).replace(",", ""))
    repair_price["total"][i] = int(str(repair_price["total"][i]).replace(",", ""))
```

```
#### 하나의 문자열로 이루어진 repair값에서 필요한 단어만 추출 ####
ex_list = []

for idx in tqdm(range(len(repair_price["repair"]))):
    repair_parts = repair_price["repair"][idx][1:-2].split(":")
    repair_parts = [ part.split("(")[0] for part in repair_parts ]
    repair_parts = [ part.split(",") for part in repair_parts ]
    repair_parts = [ repair_parts[i][j].replace("'", "").strip() for i in range(len(repair_parts))
                                                                 for j in range(len(repair_parts[i])) ]
    ex = []
    for i in range(len(repair_parts)):
        if (repair_parts[i] in part_list) == True :
            ex.append(repair_parts[i])
    ex_list.append(ex)
```

```
  | 0/232714 [00:00<?, ?it/s]
```

```
#### 합체 ####
repair_price["repair"] = ex_list
```

	category_id	coating	sheet_metal	exchange	total	repair	supercategory_name	damage
0	as-0000025	488500	175000	145750	1121128	['Front door', 'Rear fender', 'Front bumper', ...	Full-size car	Separated
1	as-0000025	488500	175000	145750	1121128	['Front door', 'Rear fender', 'Front bumper', ...	Full-size car	Scratched
2	as-0000025	488500	175000	145750	1121128	['Front door', 'Rear fender', 'Front bumper', ...	Full-size car	Crushed
3	as-0000027	496500	262500	267750	1293228	['Front door', 'Rear fender', 'Rear door', 'Re...	Mid-size car	Breakage
4	as-0000027	496500	262500	267750	1293228	['Front door', 'Rear fender', 'Rear door', 'Re...	Mid-size car	Scratched

B. 전처리

[2-1] HQ모델 학습을 위한 데이터셋 준비

작업 항목 및 부품에서 이미지 학습시키는 부위 외 부품 삭제 및 이름 변경 작업을 위한 전처리

```
category_id=[]
part=[]
repair=[]
HQ=[]

for file_name in tqdm(label_list):
    with open(file_name,'rt', encoding='UTF8') as file:
        contents=json.loads(file.read())
    for i in range(len(contents["수리내역"])):
        category_id.append(file_name.split("##")[-1][:-5])
        part.append(contents["수리내역"][i]["작업항목 및 부품
        repair.append(contents["수리내역"][i]["작업"])
        HQ.append(contents["수리내역"][i]["HQ%"])

data = pd.DataFrame({"category_id" : category_id, "part" : part, "repair" : repair, "HQ" : HQ})
```

	category_id	part	repair	HQ
0	as-0000002	앞범퍼(스틸형)하단	교환	1.24
1	as-0000002	앞범퍼	교환	3.01
2	as-0000002	헤드램프(좌)	탈착	0.42
3	as-0000002	헤드램프(우)	탈착	0.42
4	as-0000002	앞패널	판금	2.5

```
data["repair"].unique()
```

```
array(['교환', '탈착', '판금', '1/20H', '도장', '수리', '', '오버홀', '조정', '견인',
       '1/40H', '1/30H', '견인비', '1/2오버홀', '1/4오버홀', '구난비', '구난', '1/3오버홀',
       '불인정', '작업'], dtype=object)
```

```
print(len(data[data["repair"]=="교환"])) # exchange -> 탈착교환
print(len(data[data["repair"]=="탈착"]))

print(len(data[data["repair"]=="도장"])) # coating -> 도장공임

print(len(data[data["repair"]=="판금"])) # 판금 sheet_metal -> 판금수리
print(len(data[data["repair"]=="수리"]))
```

```
#### 해당 작업 항목 행 삭제 ####
data_drop = data.copy()
drop_list = ["", "구난", "구난비", "불인정", "차감소계", "감가상각", "작업", "견인", "견인비",
             "1/20H", "오버홀", "조정", "1/40H", "1/30H", "1/2오버홀", "1/4오버홀", "1/3오버홀"]

for drop in drop_list:
    data_drop = data_drop.drop(data_drop[data_drop["repair"]==drop].index, axis=0)
```

```
data_drop = data_drop.reset_index()
data_drop.drop("index", axis=1, inplace=True)
```

```
data_drop["repair"].unique()
```

```
array(['교환', '탈착', '판금', '도장', '수리'], dtype=object)
```

```
part1 = pd.read_csv("part1_part.csv", encoding="euc-kr")
part2 = pd.read_csv("part2_part.csv", encoding="euc-kr")
part3 = pd.read_csv("part3_part.csv", encoding="euc-kr")
part4 = pd.read_csv("part4_part.csv", encoding="euc-kr")
part5 = pd.read_csv("part5_part.csv", encoding="euc-kr")
```

```
part2 = part2[["part", "Unnamed: 1"]]
part3.rename(columns = {"part.1":"Unnamed: 1"}, inplace = True)
```

```
part_cc = pd.concat([part1, part2, part3, part4, part5], axis=0, ignore_index=True)
part_cc.head()
```

```
HQ_data = pd.merge(data_drop, part_cc, how='inner', on="part")
HQ_data.head()
```

category_id	part	repair	HQ	Unnamed: 1
as-0000025	후론트 범퍼	교환	1.79	Bumper
as-0000596	후론트 범퍼	교환	2.45	Bumper
as-0001501	후론트 범퍼	탈착	1.5	Bumper
as-0001551	후론트 범퍼	탈착	0.81	Bumper
as-0001576	후론트 범퍼	탈착	0.81	Bumper

```
HQ_data = HQ_data[["category_id", "Unnamed: 1", "repair", "HQ"]]
HQ_data.rename(columns = {"Unnamed: 1" : "part"}, inplace = True)
HQ_data.head()
```

category_id	part	repair	HQ
as-0000025	Bumper	교환	1.79
as-0000596	Bumper	교환	2.45
as-0001501	Bumper	탈착	1.5
as-0001551	Bumper	탈착	0.81
as-0001576	Bumper	탈착	0.81

B. 전처리

[2-2] HQ모델 학습을 위한 데이터셋 준비

모델 학습에 맞게 작업 방식과 부품명을 통일하고 사고 유형 정보에서
HQ(공업시간)에 영향을 주는 변수 merge시킴

```
# 작업 방식 이름 변경 #
idx = HQ_data[HQ_data["repair"]=="교환"].index
HQ_data["repair"][idx] = "exchange"
idx = HQ_data[HQ_data["repair"]=="탈착"].index
HQ_data["repair"][idx] = "exchange"

idx = HQ_data[HQ_data["repair"]=="도장"].index
HQ_data["repair"][idx] = "coating"

idx = HQ_data[HQ_data["repair"]=="판금"].index
HQ_data["repair"][idx] = "sheet_metal"
idx = HQ_data[HQ_data["repair"]=="수리"].index
HQ_data["repair"][idx] = "sheet_metal"
```

```
HQ_data["repair"].unique()
```

```
array(['exchange', 'sheet_metal', 'coating'], dtype=object)
```

```
# 부품명 통일 #
idx = HQ_data[HQ_data["part"]=="door"].index
HQ_data["part"][idx] = "Door"
```

```
HQ_data["part"].unique()
```

```
array(['Bumper', 'Head lights', 'Fender', 'Door', 'Wheel', 'Rocker panel',
       'Rear lamp', 'Side mirror', 'Trunk lid', 'Bonnet', 'Roof'],
      dtype=object)
```

```
# HQ와 사고유형 합체 #
merge_data = pd.merge(HQ_data, merge_car, how='inner',on='category_id')
merge_data.head()
```

	category_id	part	repair	HQ	damage	supercategory_name
0	as-0000025	Bumper	exchange	1.79	Crushed	Full-size car
1	as-0000025	Bumper	exchange	1.79	Separated	Full-size car
2	as-0000025	Bumper	exchange	1.79	Scratched	Full-size car
3	as-0000025	Bumper	exchange	1.79	Crushed	Full-size car
4	as-0000025	Bumper	exchange	1.79	Scratched	Full-size car

```
merge_data = merge_data[["category_id", "part", "damage", "supercategory_name", "repair", "HQ"]]
```

```
# 중복 없애주기 #
m_data = merge_data.drop_duplicates(keep = 'first', ignore_index=True)
m_data.head()
```

	category_id	part	damage	supercategory_name	repair	HQ
0	as-0000025	Bumper	Crushed	Full-size car	exchange	1.79
1	as-0000025	Bumper	Separated	Full-size car	exchange	1.79
2	as-0000025	Bumper	Scratched	Full-size car	exchange	1.79
3	as-0000025	Head lights	Crushed	Full-size car	exchange	0.44
4	as-0000025	Head lights	Separated	Full-size car	exchange	0.44

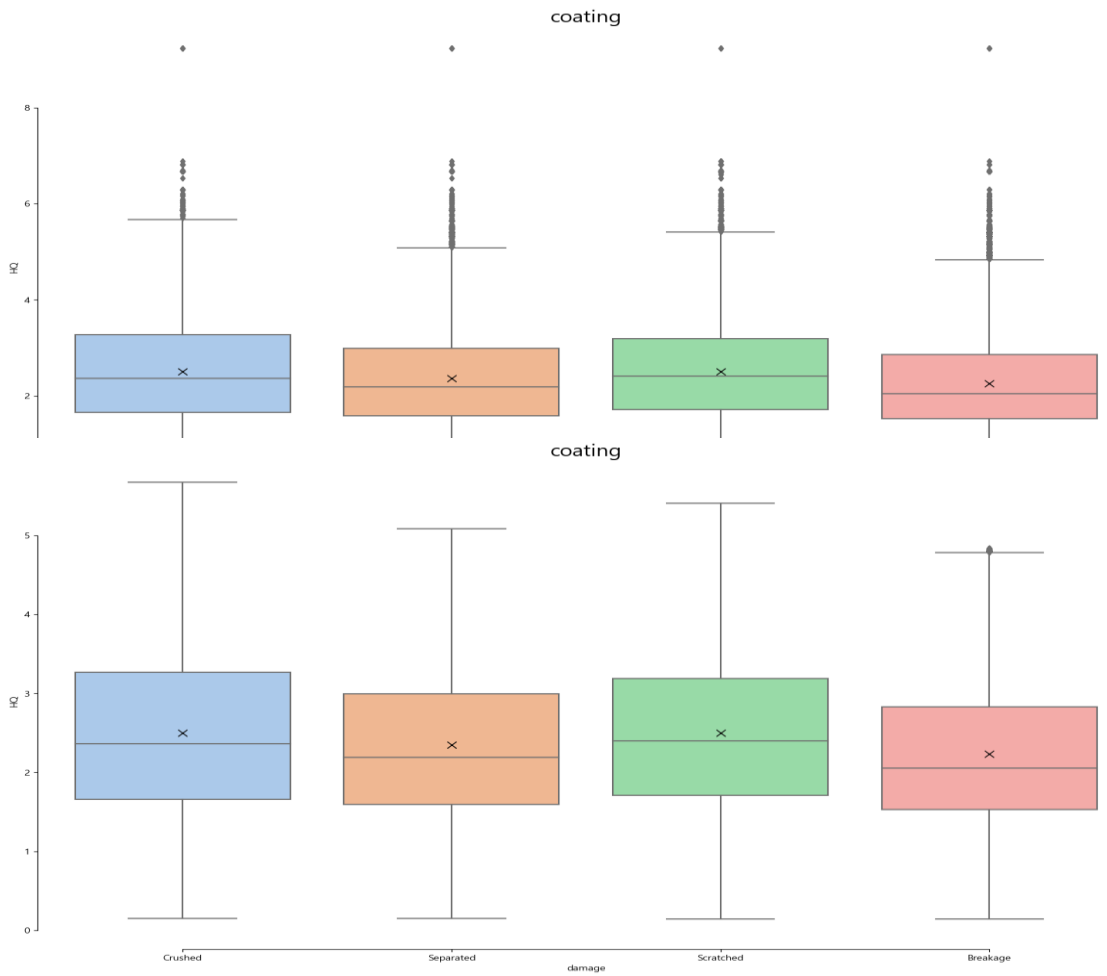
B. 전처리

[3-1] HQ모델 학습을 위한 데이터셋 전처리

수리 방법(도장, 판금, 교환) 모두 통일하게 진행 파손 유형 및 부위별 HQ 분포 확인 후 이상치 제거

- IQR = Q3(3사분위수)-Q1(1사분위수)
- 제거기준 = Q3 + IQR * 1.5 보다 큰 값

```
Crushed = coating[coating["damage"]=="Crushed"]
Separated = coating[coating["damage"]=="Separated"]
Scratched = coating[coating["damage"]=="Scratched"]
Breakage = coating[coating["damage"]=="Breakage"]


q1=Crushed.quantile(0.25)
q3=Crushed.quantile(0.75)
iqr=q3-q1
Crushed = Crushed[Crushed[Crushed[['HQ']] <= (q3 + 1.5*iqr)]["HQ"].isnull()!=True]


q1=Separated.quantile(0.25)
q3=Separated.quantile(0.75)
iqr=q3-q1
Separated = Separated[Separated[Separated[['HQ']] <= (q3 + 1.5*iqr)]["HQ"].isnull()!=True]


q1=Scratched.quantile(0.25)
q3=Scratched.quantile(0.75)
iqr=q3-q1
Scratched = Scratched[Scratched[Scratched[['HQ']] <= (q3 + 1.5*iqr)]["HQ"].isnull()!=True]


q1=Breakage.quantile(0.25)
q3=Breakage.quantile(0.75)
iqr=q3-q1
Breakage = Breakage[Breakage[Breakage[['HQ']] <= (q3 + 1.5*iqr)]["HQ"].isnull()!=True]

HQ_coating = pd.concat([Crushed, Separated, Scratched, Breakage])
HQ_coating.head()
```



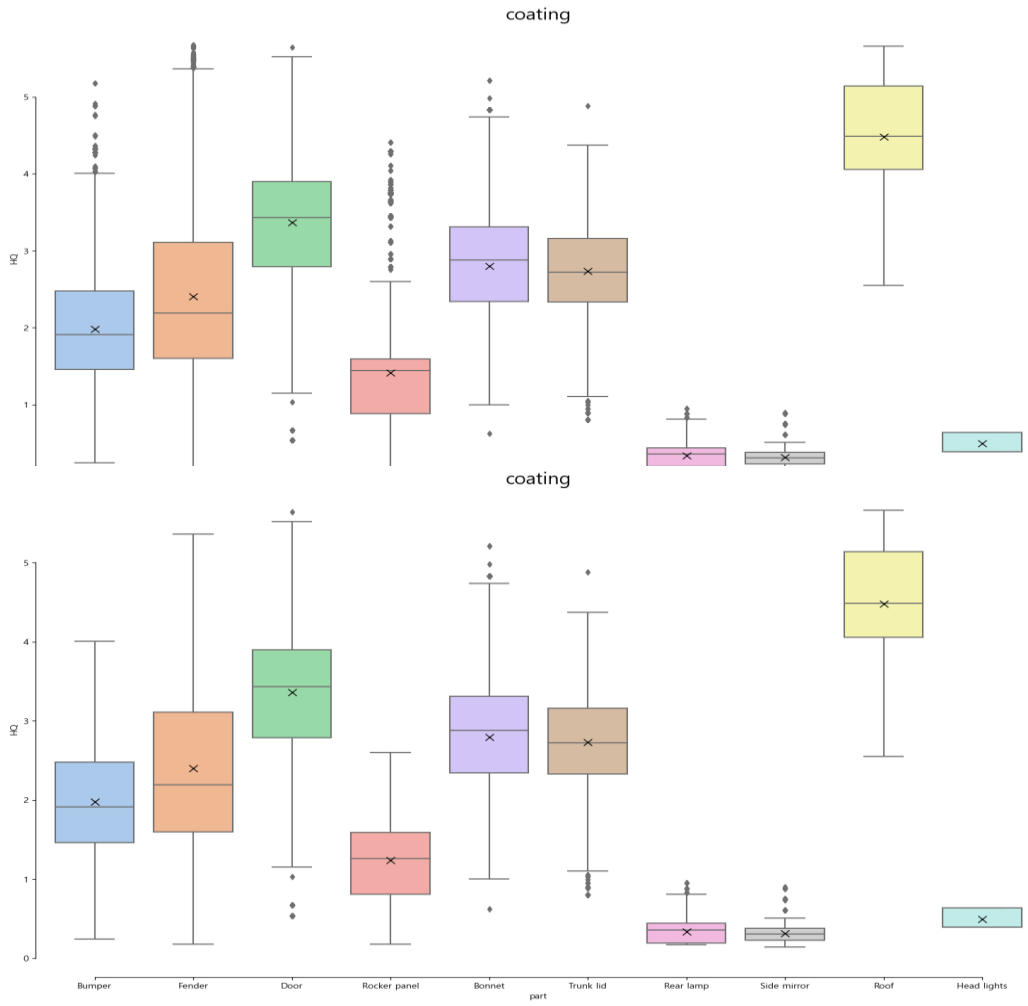
B. 전처리

[3-2] HQ모델 학습을 위한 데이터셋 전처리

수리 방법(도장, 판금, 교환) 모두 통일하게 진행 파손 유형 및 부위별 HQ 분포 확인 후 이상치 제거

- IQR = Q3(3사분위수)-Q1(1사분위수)
- 제거기준 = Q3 + IQR * 1.5 보다 큰 값

```
Fender = HQ_coating[HQ_coating["part"]=="Fender"]
Rocker_panel = HQ_coating[HQ_coating["part"]=="Rocker panel"]
Bumper = HQ_coating[HQ_coating["part"]=="Bumper"]

q1=Fender.quantile(0.25)
q3=Fender.quantile(0.75)
iqr=q3-q1
Fender_idx = Fender[Fender[Fender[['HQ']] <= (q3 + 1.5*iqr)]["HQ"].isnull()==True].index


q1=Rocker_panel.quantile(0.25)
q3=Rocker_panel.quantile(0.75)
iqr=q3-q1
Rocker_idx = Rocker_panel[Rocker_panel[Rocker_panel[['HQ']] <= (q3 + 1.5*iqr)]["HQ"].isnull()==True].index


q1=Bumper.quantile(0.25)
q3=Bumper.quantile(0.75)
iqr=q3-q1
Bumper_idx = Bumper[Bumper[Bumper[['HQ']] <= (q3 + 1.5*iqr)]["HQ"].isnull()==True].index

HQ_coating.drop(Fender_idx, axis=0, inplace=True)
HQ_coating.drop(Rocker_idx, axis=0, inplace=True)
HQ_coating.drop(Bumper_idx, axis=0, inplace=True)
HQ_coating.head()
```



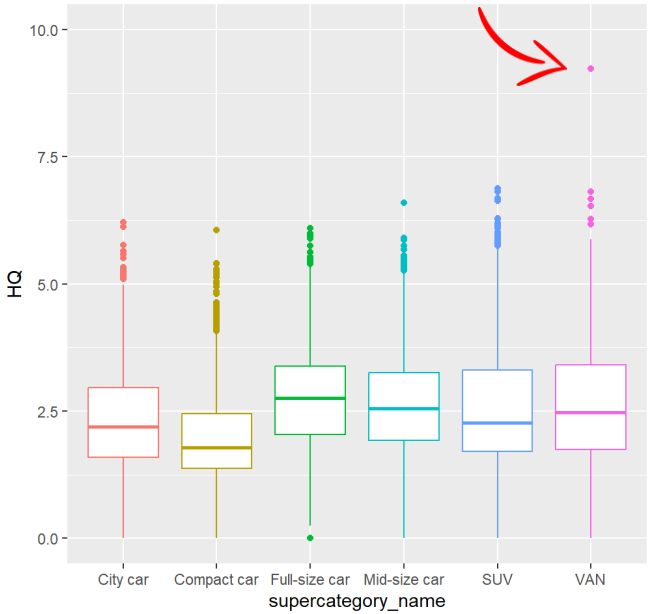
B. 전처리

[4] HQ모델 학습을 위한 데이터셋 전처리

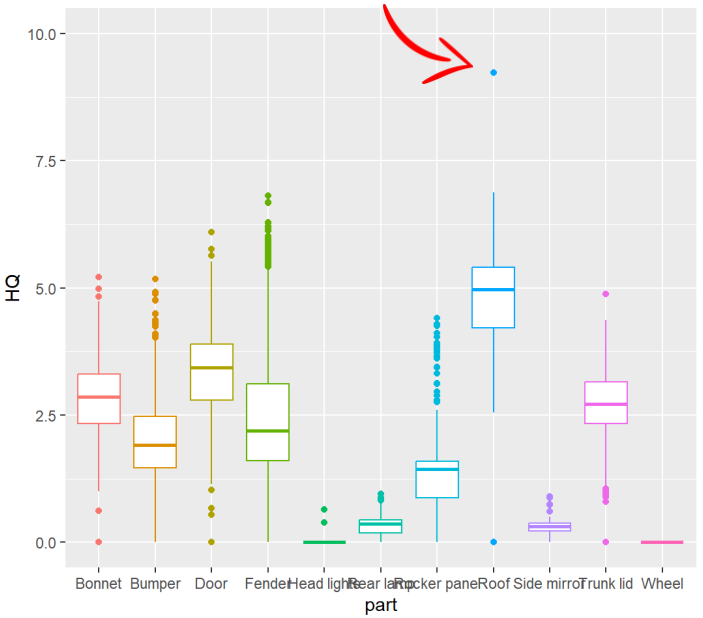
변수별 HQ를 BOXPLOT으로 시각화 이상치 확인 후 제거

HQ = 작업시간

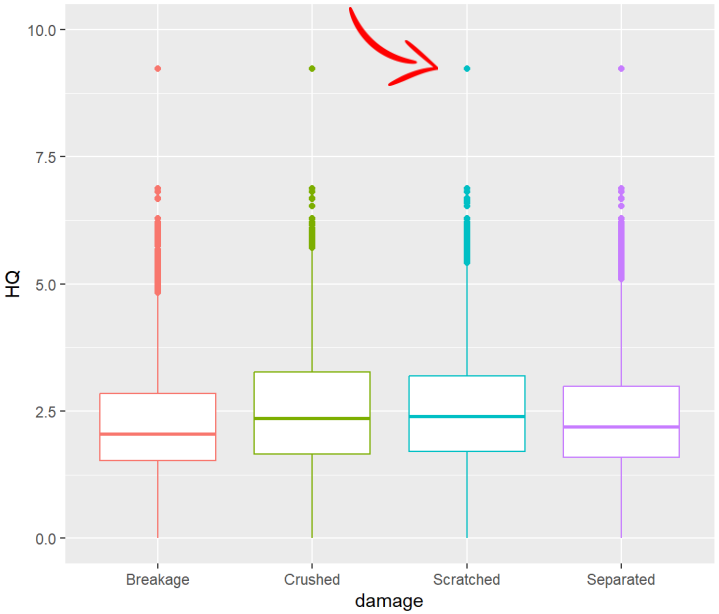
SIZE



PART



DAMAGE



통계검증

c. 통계검증

ANOVA, t검정을 이용하여 변수가 가격에 유의미한 영향을 미치는지 판단하기 위한 검정으로 모든 변수에 영향을 끼치는 것으로 판단.

Preview

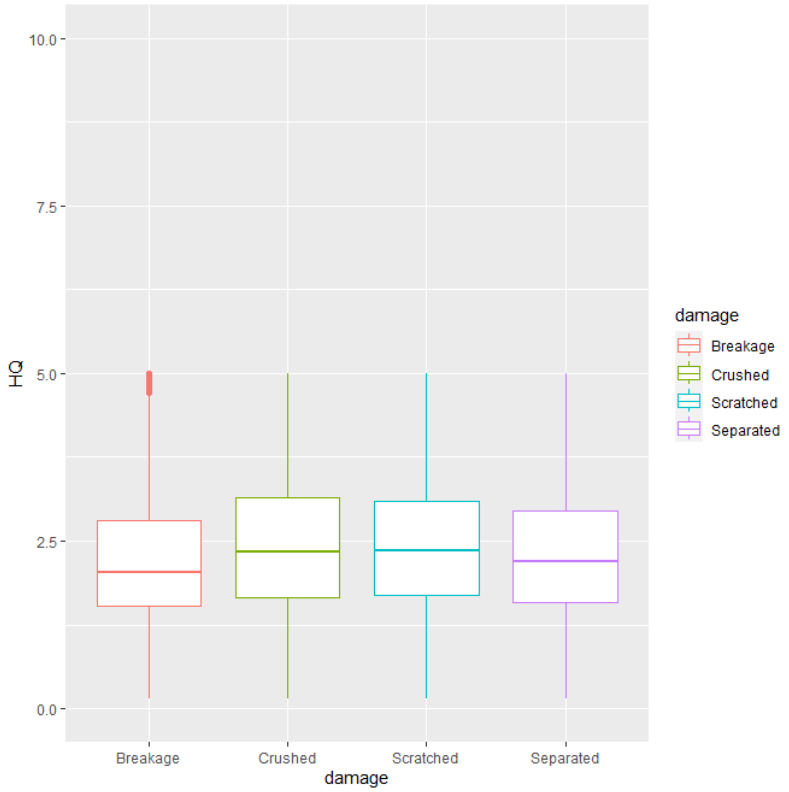
Purpose

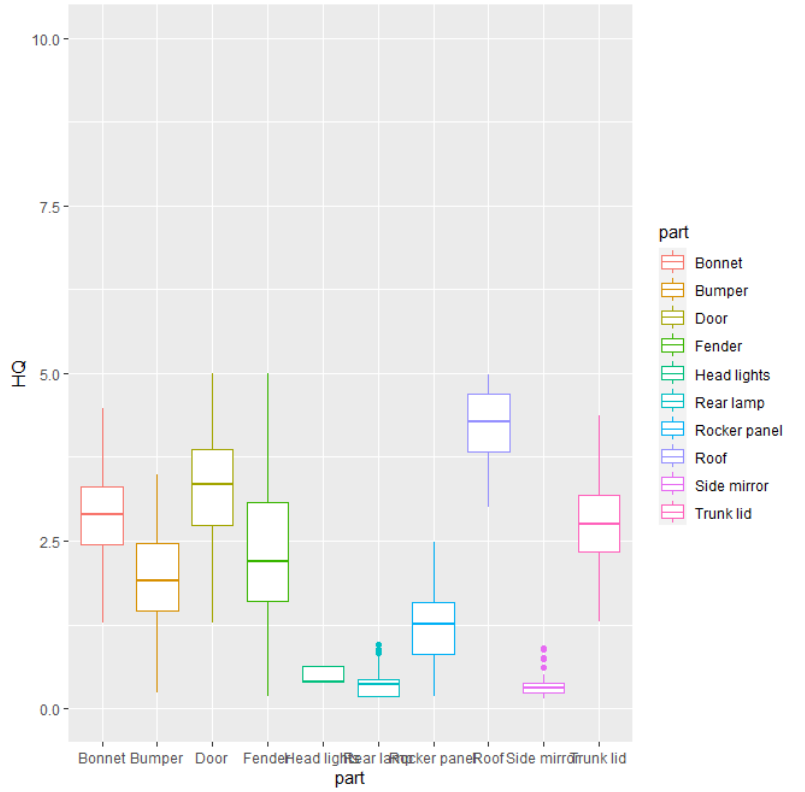
차량 수리 비용 예측 모델에 사용할 차종, 파손 유형, 색상, 연식이 수리비에 유의미한 영향을 미치는 변수인지 검정하기 위해 진행한 분석

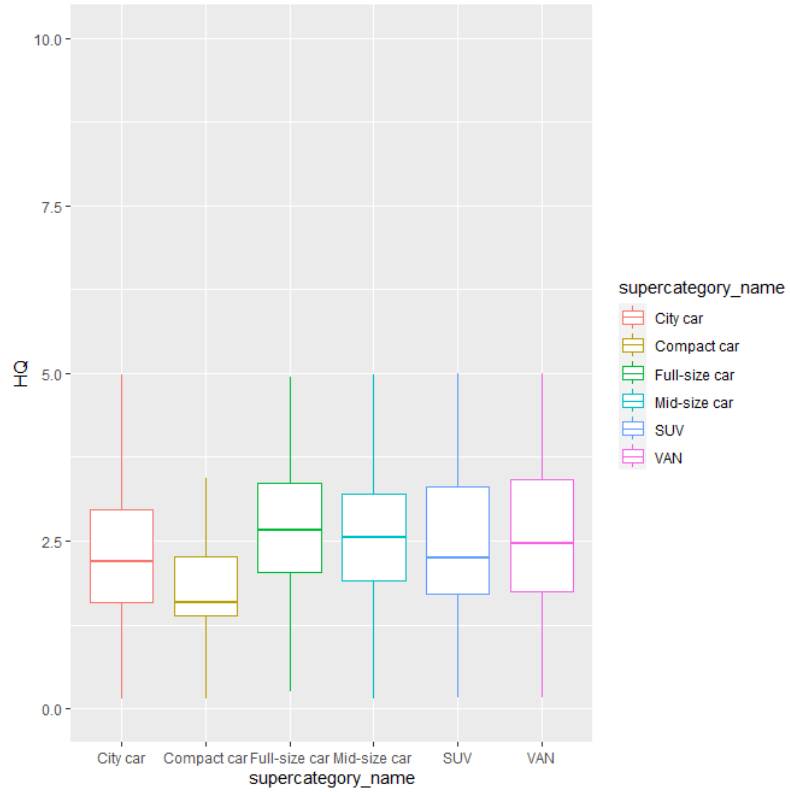
변수	도장 (coating)	판금수리 (sheet_metal)	탈착교환 (exchange)	비고
차종 (supercategory_name)	True	True	True	모든 변수의 P-value가 유의미하게 작음
파손유형(damage)	True	True	True	모든 변수의 P-value가 유의미하게 작음
색상(color)	True	True	True	coating과 exchage가 유의미한 차이가 존재하는 색상 조합이 많음
연식(year)	True	True	True	모든 변수의 P-value가 유의미하게 작음

c. 통계검증

HQ에 대한 PART, DAMAGE, SIZE별 ANOVA 검정 실시, 각 변수가 HQ에 영향을 미친다고 판단 됨
변수별 HQ에 대한 평균차이는 미미하지만 눈으로 보이지 않을 수도 있음







c. 통계검증

HQ에 대한 PART, DAMAGE, SIZE별 ANOVA 검정 실시, 각 변수가 HQ에 영향을 미친다고 판단 됨
변수별 HQ에 대한 평균차이는 미미하지만 눈으로 보이지 않을 수도 있음

2. 수리시간에 영향을 미치는 요인

1. coating

1. part

```
setwd("C:₩₩Users₩₩acorn₩₩Desktop₩₩김현욱 (2)")
coating<-read.csv("coating_part1.csv", encoding = "EUC-KR")

fit <- aov(HQ~part, data=coating)
summary(fit)
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## part            9 165255   18362   32038 <2e-16 ***
## Residuals   488062 279722       1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Pr(>F) <2e-16 : 파손부위와 도색시간은 영향이 있다고 판단됨
- 파손부위별 수리시간 평균

	part	mean
	<chr>	<dbl>
1	Bonnet	2.86
2	Bumper	1.96
3	Door	3.33
4	Fender	2.38
5	Head lights	0.497
6	Rear lamp	0.336
7	Rocker panel	1.24
8	Roof	4.23
9	Side mirror	0.313
10	Trunk lid	2.75

2. damage

```
fit <- aov(HQ~damage,data=coating)
summary(fit)
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## damage          3   4137  1379.0    1527 <2e-16 ***
## Residuals   488068 440841     0.9
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Pr(>F) <2e-16 : 파손유형과 도색시간은 영향이 있다고 판단됨
- 파손유형별로 도색시간의 평균

	damage	mean
	<chr>	<dbl>
1	Breakage	2.22
2	Crushed	2.45
3	Scratched	2.46
4	Separated	2.32

3. supercateory_name

```
fit1 <- aov(HQ~supercategory_name,data=exchange)
summary(fit1)
```

```
##                      Df Sum Sq Mean Sq F value Pr(>F)
## supercategory_name    5  10416  2083.1    2312 <2e-16 ***
## Residuals        563044 507304     0.9
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Pr(>F)<2e-16 : 차량의 크기별로 교환시간은 영향이 있다고 보임
- 차량 크기별 교환시간 평균

	supercategory_name	mean
	<chr>	<dbl>
1	City car	0.835
2	Compact car	0.715
3	Full-size car	1.08
4	Mid-size car	0.932
5	SUV	1.09
6	VAN	1.03

모델학습

[1] 파손 부위 판별 모델

가격 산출을 위한 차량 부위별 판별모델 생성, L1 > L2 과적합 최소화

CNN

MobileNet
- L2 norm
- Filter:128
- lr=1e-5
- Epochs=80

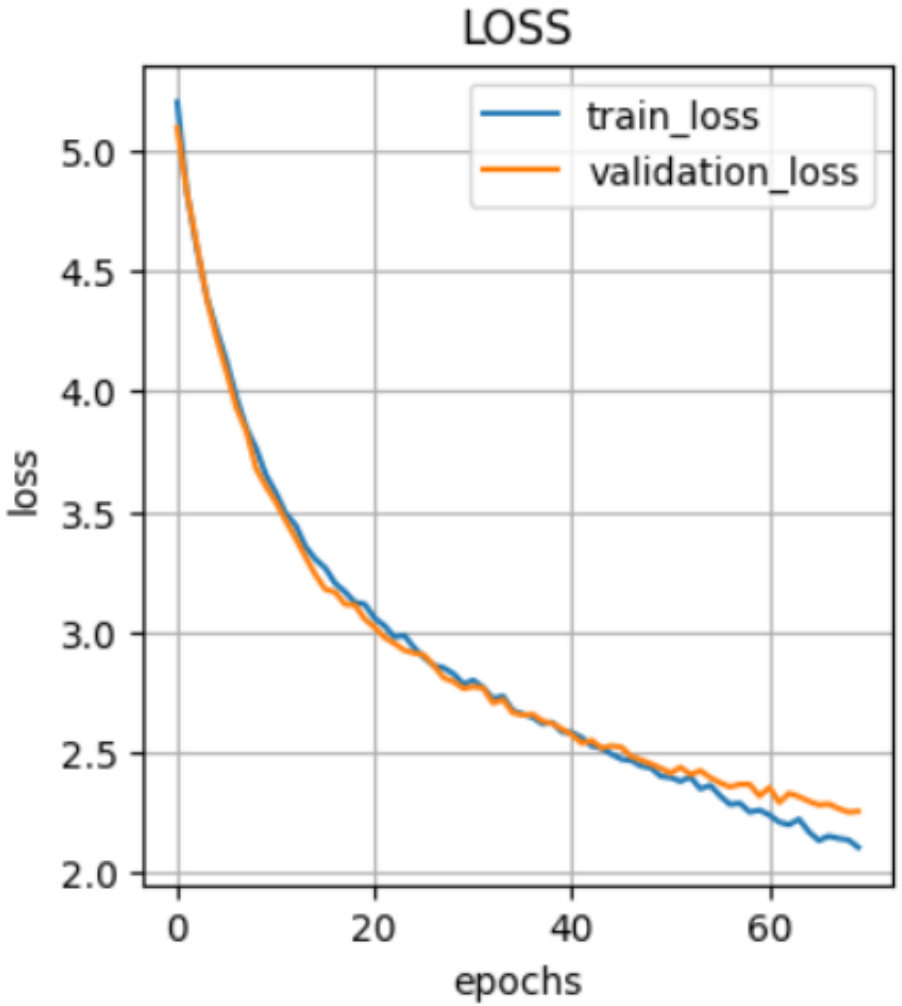
```
base_model = MobileNetV2(input_shape=(image_size[0], image_size[1], 3), include_top=False, weights='imagenet')

model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(128, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(Dropout(0.5))
model.add(Dense(12, activation='softmax'))

model.compile(optimizer=Adam(learning_rate=1e-5), loss='categorical_crossentropy', metrics=['accuracy'])

early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

start_time = datetime.now()

result = model.fit(x=train_generator, epochs=80, validation_data=test_generator, callbacks=[early_stopping])

end_time = datetime.now()

print('걸린 시간:', end_time - start_time)

test_loss, test_accuracy = model.evaluate(test_generator)
print(f'Test Loss: {test_loss}')
print(f'Test Accuracy: {test_accuracy}')
```

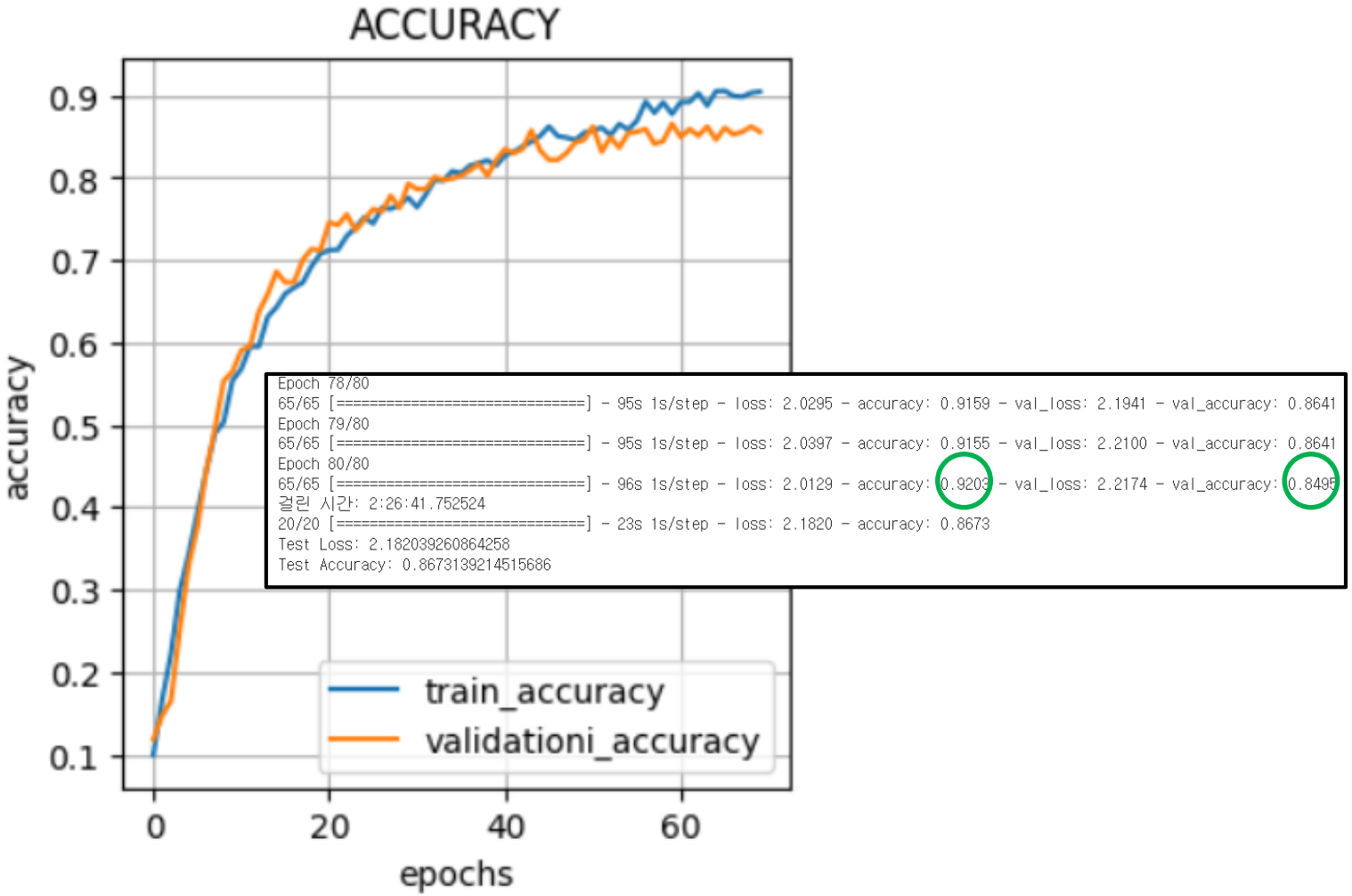
D. 모델 학습

[1] 파손 부위 판별 모델

CNN

가격 산출을 위한 차량 부위별 판별모델 생성 점수는 높게 나오나, 과적합이 아쉬움.






```
Epoch 78/80
65/65 [==============================] - 95s 1s/step - loss: 2.0295 - accuracy: 0.9159 - val_loss: 2.1941 - val_accuracy: 0.8641
Epoch 79/80
65/65 [==============================] - 95s 1s/step - loss: 2.0397 - accuracy: 0.9155 - val_loss: 2.2100 - val_accuracy: 0.8641
Epoch 80/80
65/65 [==============================] - 96s 1s/step - loss: 2.0129 - accuracy: 0.9203 - val_loss: 2.2174 - val_accuracy: 0.8495
걸린 시간: 2:26:41.752524
20/20 [==============================] - 23s 1s/step - loss: 2.1820 - accuracy: 0.8673
Test Loss: 2.182039260864258
Test Accuracy: 0.8673139214515686
```

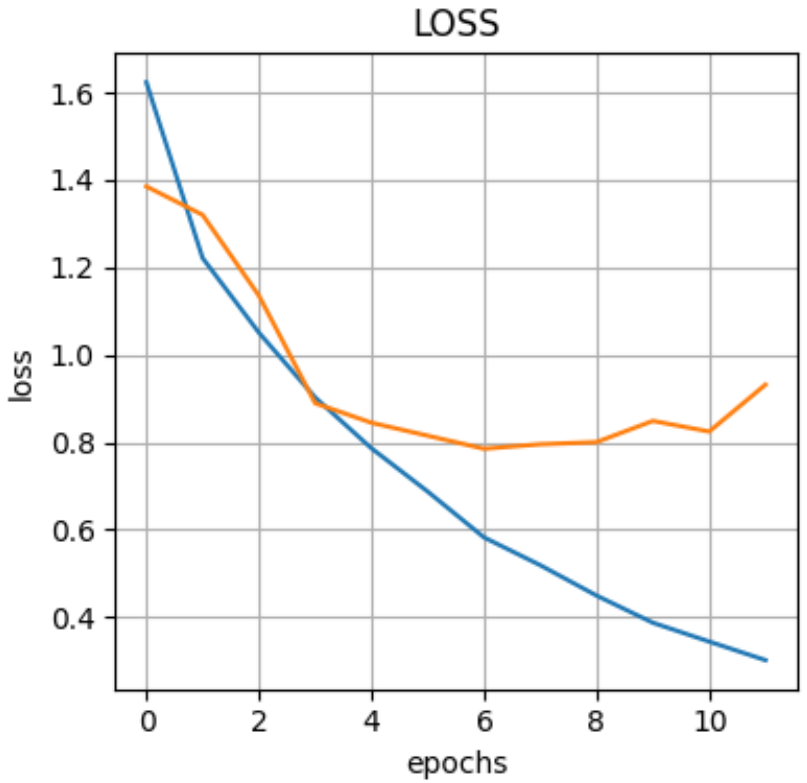
[2] 파손 데미지 판별 모델

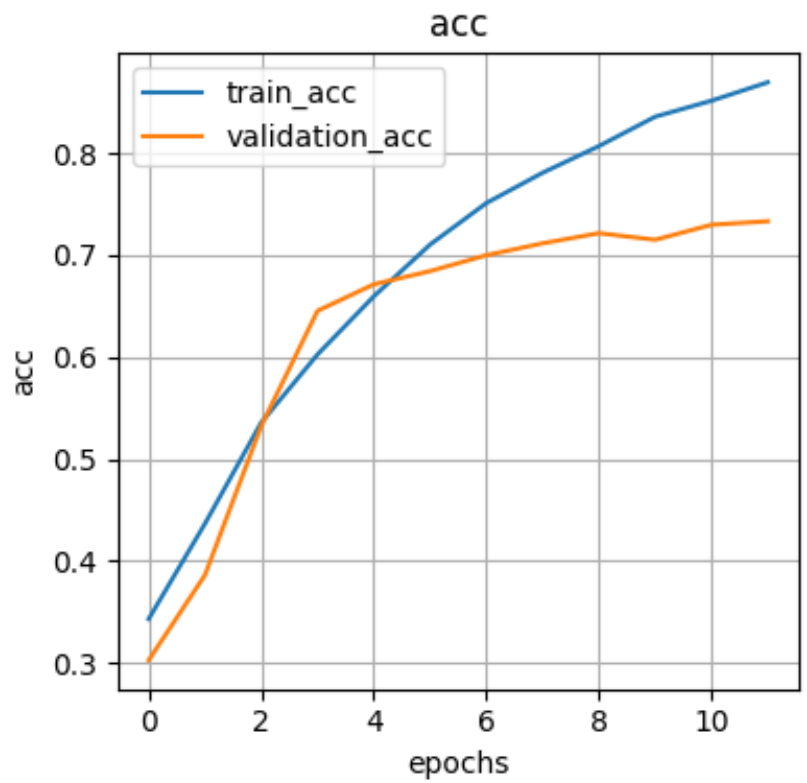
데미지별 수리가격이 상이하기에 가격산출을 위해 데미지 손상도를 4가지로 분류해주는 모델 생성

CNN

ResNet
- Filter:64
- lr=1e-5
- Epochs=30

```
from matplotlib.cbook import flatten
base_model = ResNet50(weights='imagenet',
                      include_top=False,
                      input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3))

model = Sequential()  # Fc계층 만들어주기

model.add(base_model)
model.add(Flatten())
model.add(Dense(64, activation="relu"))  # 은닉층
model.add(Dropout(0.65))

model.add(Dense(4, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy',
              optimizer=Adam(learning_rate=1e-5),
              metrics=['accuracy'])

tf_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)

model.summary()
```





YOLO

[2] 파손 데미지 판별 모델

YOLO모델을 활용하여 데미지 판별을 위해 bbox정보들을 TXT파일로 만들기 위한 전처리 과정

1. 견적서 JSON파일 기반, BBOX 절대값을 상대비율로 조정, DF 생성

```
image_id=[]
category_id=[]
width=[]
height=[]
bbox=[]
damage=[]

# conner 기반 bbox를 center 기반 bbox로 변환
def corner_to_center(bbox):
    x, y, width, height = bbox

    center_x = int(x + width / 2)
    center_y = int(y + height / 2)

    return [center_x, center_y, width, height]

num=0  # 00001 과 같이 폴더명에 있는 이미지 파일 이름 가져오기
for file_name in tqdm(label_list):
    with open(file_name,'r') as j:
        contents=json.loads(j.read())

    for i in range(len(contents["annotations"])):
        image_id.append(label_list[num].split("\\")[-1][:7])
        category_id.append(contents["annotations"][i]["category_id"])
        width.append(contents["images"]["width"])
        height.append(contents["images"]["height"])
        bbox.append(corner_to_center(contents["annotations"][i]["bbox"]))
        damage.append(contents["annotations"][i]["damage"])

    num+=1

car_repair=pd.DataFrame({"image_id":image_id, "category_id":category_id, "W":width, "H":height,
                         "bbox":bbox, "damage":damage})
```

```
100%|██████████| 255485/255485 [35:16<00:00, 120.72it/s]
```

2. 코랩에서 YOLO모델을 학습을 위한 압축

```
import glob
import zipfile
from tqdm import tqdm

# 경로와 압축 파일 이름 지정
file_pattern = 'D:/car/data/Training/feature/TS_damage/damage/*.jpg'
output_path = 'D:/car/data/Training/feature/TS_damage/damage/damage_train_images.zip'

# 파일 경로 리스트 가져오기
file_paths = glob.glob(file_pattern)

with zipfile.ZipFile(output_path, 'w', zipfile.ZIP_DEFLATED) as zipf:
    # tqdm을 사용하여 진행 상황 표시
    for file_path in tqdm(file_paths[80000:97500], desc='Compressing files', unit='files'):
        zipf.write(file_path, arcname=file_path.split('/')[-1])
```

```
Compressing files: 100%|██████████| 17500/17500 [07:56<00:00, 36.70files/s]
```

train-txt 파일 압축

```
file_pattern = 'C:/labels/*.txt'
output_path = 'C:/labels/damage_train_labels_97500.zip'

# 파일 경로 리스트 가져오기
file_paths = glob.glob(file_pattern)

with zipfile.ZipFile(output_path, 'w', zipfile.ZIP_DEFLATED) as zipf:
    # tqdm을 사용하여 진행 상황 표시
    for file_path in tqdm(file_paths[80000:97500], desc='Compressing files', unit='files'):
        zipf.write(file_path, arcname=file_path.split('/')[-1])
```

```
Compressing files: 100%|██████████| 17500/17500 [01:14<00:00, 233.91files/s]
```

YOLO

사진의 해상도가 적용된 BBOX의 좌표를 상대 비율로 전환하기 위한 전처리 과정

견적서 원본


```
{"id": 3, "image_id": 1, "category_id": "as-0082746",
"area": 1830.0, "bbox": [568, 470, 57, 49], "damage": "Scratched",

{"id": 4, "image_id": 1, "category_id": "as-0082746",
"area": 273.5, "bbox": [594, 493, 31, 15], "damage": "Crushed",
```


Corner BBOX (X, Y, W, H): [568, 470, 57, 49]

↓

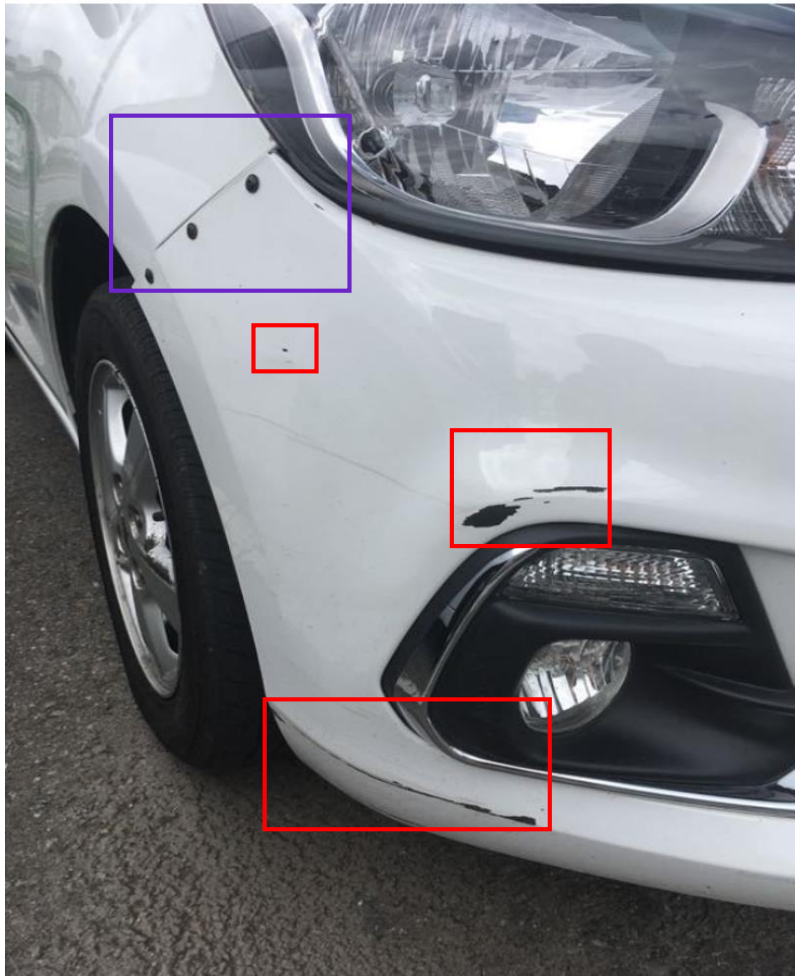
Center BBOX (X, Y, W, H): [598 , 495, 57, 49]

↓

해상도 : (800 * 600)

↓

Final BBOX: [0.747 , 0.825, 0.071, 0.081]



[0001061_sc_123724.jpg]

YOLO

BBOX = 클래스, X좌표, Y좌표, W(너비), Y(높이) / YAML 파일 생성

[0001061_sc_123724.txt]

{Crushed:0, Scratched:1, Breakage:2, Separated:3}

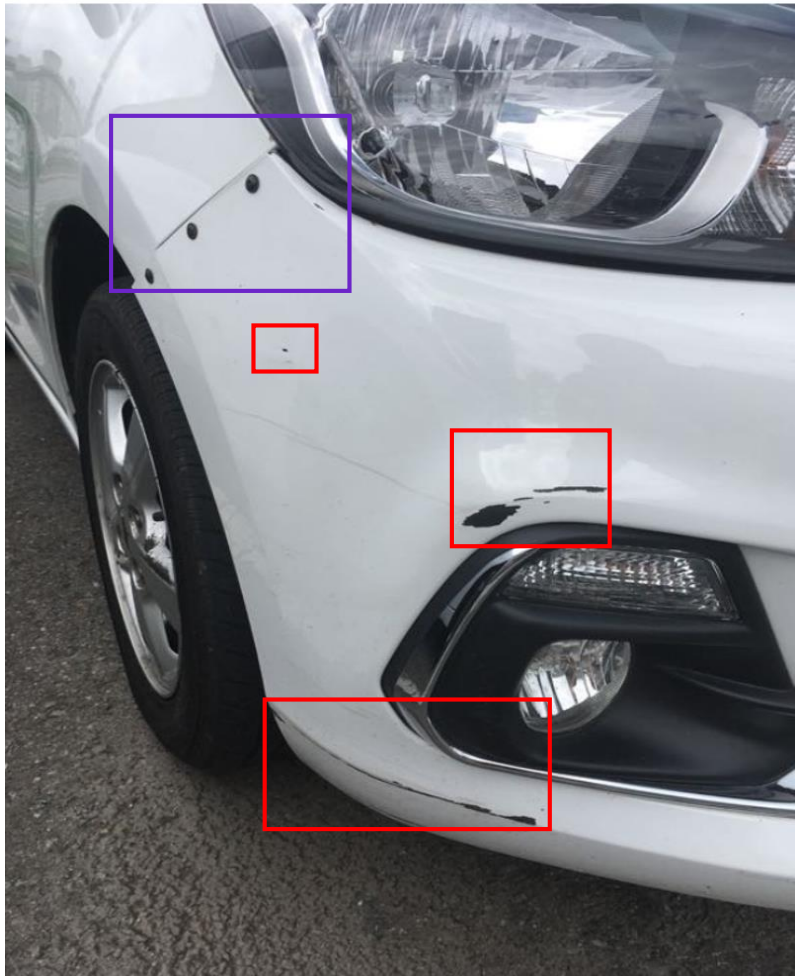
```
3 0.26375 0.2150735294117647 0.235 0.1819529411764705
1 0.6775 0.5193014705882353 0.2325 0.07077205882352941
1 0.35625 0.3612132352941176 0.0225 0.021139705882352942
1 0.39375 0.2113970588235294 0.03875 0.024816176470588234
```

YAML 파일 저장

```
train: /content/damage_detection/train/images
val: /content/damage_detection/valid/images

nc: 4
names: ["Crushed", "Scratched","Breakage","Separated"]
```

데이터의 수

Train : img, txt : 33713개
Valid : img, txt : 5828개



[0001061_sc_123724.jpg]

YOLO

[2] 파손 데미지 판별 모델

YOLO모델을 활용하여 데미지 판별

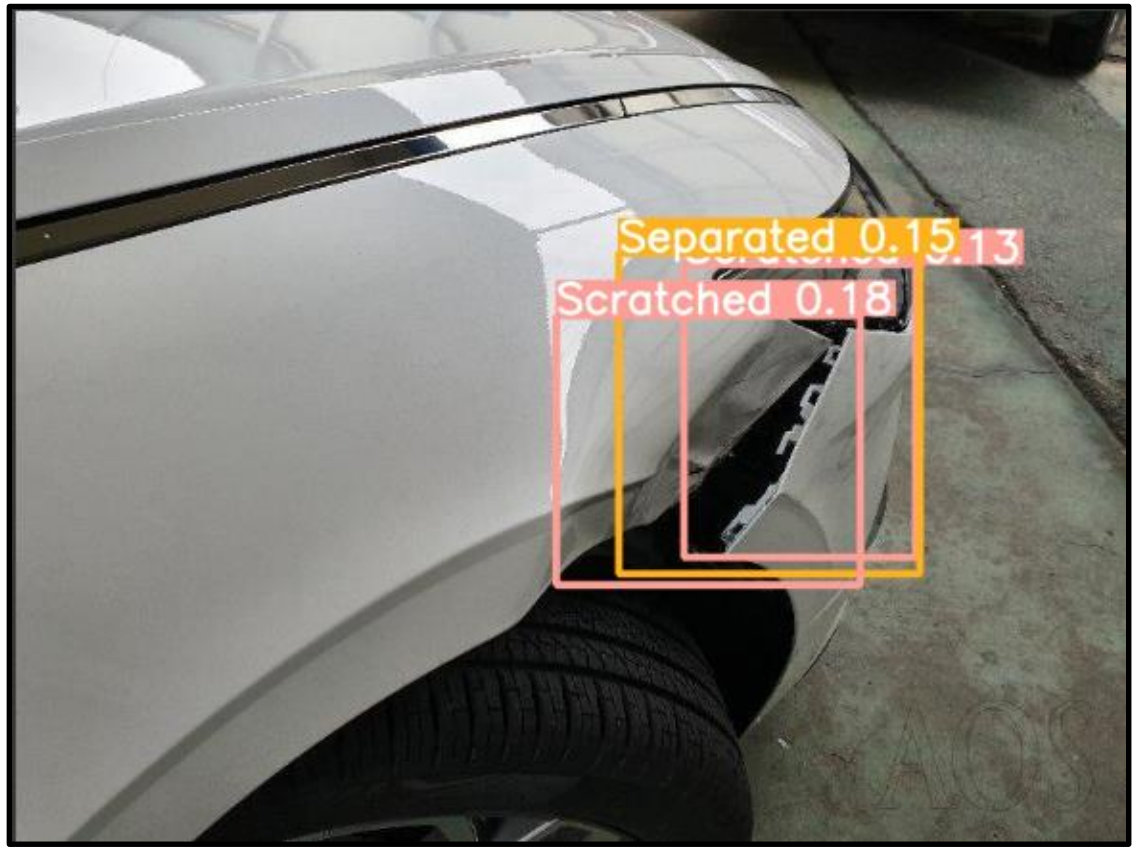
목차


```
img_size = 576
batch_size = 32
epochs = 30

data_path = os.path.join(DATA_ROOT_DIR, "data.yaml")
yaml_path = os.path.join(YOLOv5_ROOT_DIR, "models/yolov5s.yaml")
weights_path = os.path.join(YOLOv5_ROOT_DIR, "yolov5m.pt")

!python3 {os.path.join(YOLOv5_ROOT_DIR, "train.py")} \
     --img {img_size}     \
     --batch {batch_size} \
     --epochs {epochs}    \
     --data {data_path}   \
     --cfg {yaml_path}    \
     --weights {weights_path}
```

YOLO

[2] 파손 데미지 판별 모델

YOLO모델을 활용하여 데미지 판별 예시



YOLOv5s summary: 157 layers, 7020913 parameters, 0 gradients, 15.8 GFLOPs

Class	Images	Instances	P	R	mAP50	mAP50-95
all	6000	24212	0.355	0.277	0.229	0.0883
Crushed	6000	3648	0.342	0.248	0.203	0.0694
Scratched	6000	12246	0.362	0.29	0.231	0.0913
Breakage	6000	3282	0.344	0.265	0.227	0.0843
Separated	6000	5036	0.37	0.304	0.253	0.108

Results saved to **yolov5/runs/train/exp**

Map50 값이 0.229로 성능이 좋지 않음.

YOLO

[2] 파손 데미지 판별 모델

YOLO모델을 활용하여 데미지 판별

성능 개선을 위한 노력

BBOX의 크기 분포를 확인 후,
극도로 작은 값 삭제(w * h < 0.004)

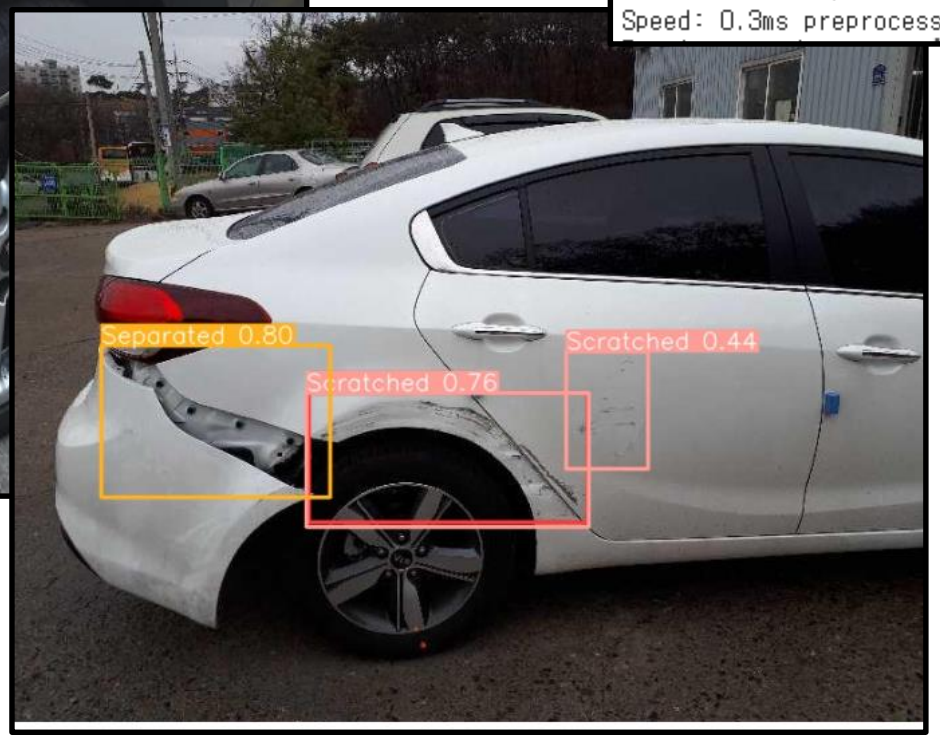
```
bbox['multiple'].describe()
```

```
count    126482.000000
mean          0.073464
std           0.129468
min           0.000006
25%           0.004010
50%           0.019350
75%           0.079231
max           1.000000
Name: multiple, dtype: float64
```

```
directory = "C:/Users/acorn/Desktop/damage_valid_labels_30000/labels"
threshold = 0.004

# 디렉토리 내의 모든 txt 파일 가져오기
txt_files = [file for file in os.listdir(directory) if file.endswith(".txt")]

for txt_file in tqdm(txt_files):
    file_path = os.path.join(directory, txt_file)

    with open(file_path, "r") as file:
        lines = file.readlines()

    # w * h 값이 threshold보다 작은 줄 삭제하기
    lines = [line for line in lines if float(line.split()[3]) * float(line.split()[4]) >= threshold]

    with open(file_path, "w") as file:
        file.writelines(lines)

    # 줄이 줄어든 경우 마지막 공백 제거
    with open(file_path, "r") as file:
        lines = file.readlines()

    lines = [line.strip() for line in lines]

    with open(file_path, "w") as file:
        file.write("\n".join(lines))
```

YOLO 성능 개선을 위한 노력

YOLO모델을 활용하여 데미지 판별



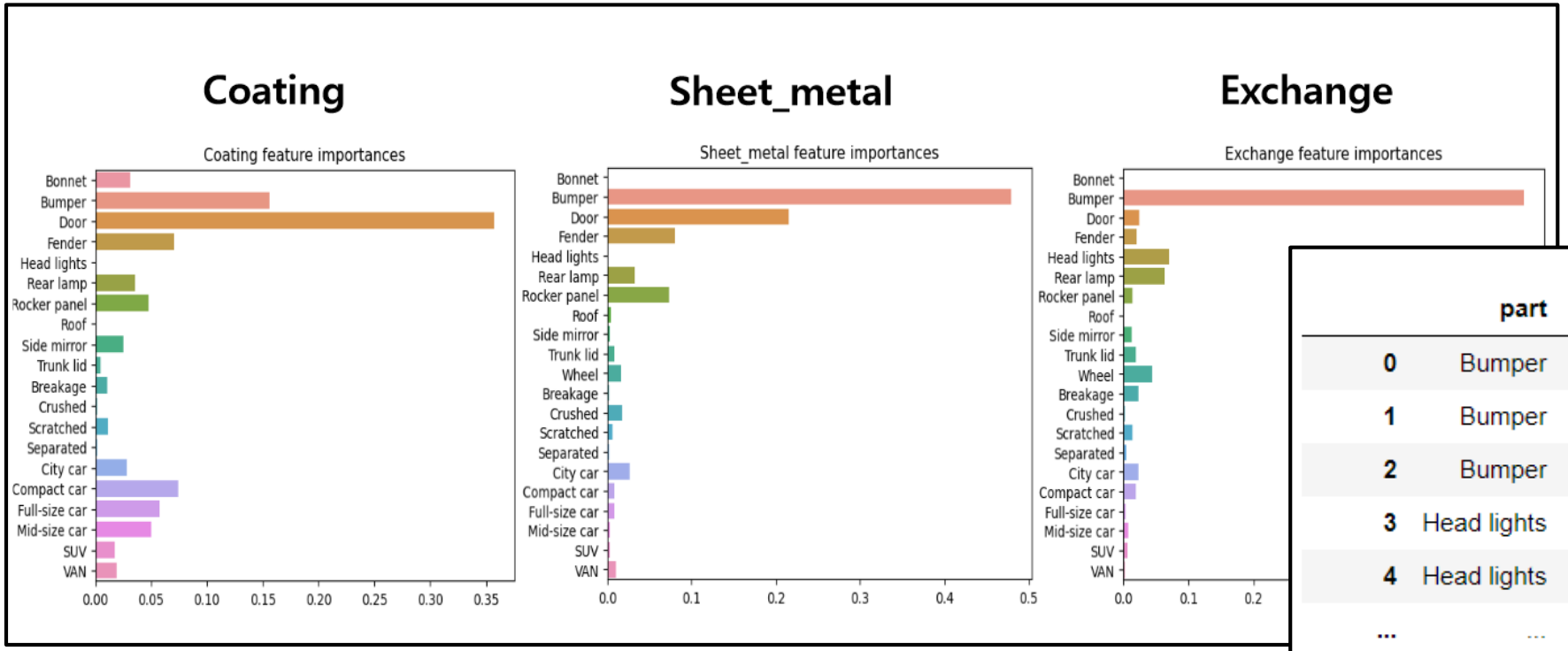

```
Ultralytics YOLOv8.0.131 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11127132 parameters, 0 gradients
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all       5828      17919      0.449      0.382      0.361      0.185
               Crushed       5828       3178      0.439      0.284      0.287      0.122
             Scratched       5828       7999      0.451      0.392       0.37      0.193
              Breakage       5828       2345      0.457      0.409      0.384      0.194
             Separated       5828       4397      0.448      0.443      0.404       0.23
Speed: 0.3ms preprocess, 2.3ms inference, 0.0ms loss, 2.1ms postprocess per image
```

Map50 값이 0.22 -> 0.36으로 개선

D. 모델 학습

[3] HQ 판별 모델

수리방법별 공임시간 예측을 위하여 Randomforest를 이용
모델 구축 과정에서 데이터 상 회귀를 사용하는 것이 적합하지 않아 분류로 구축



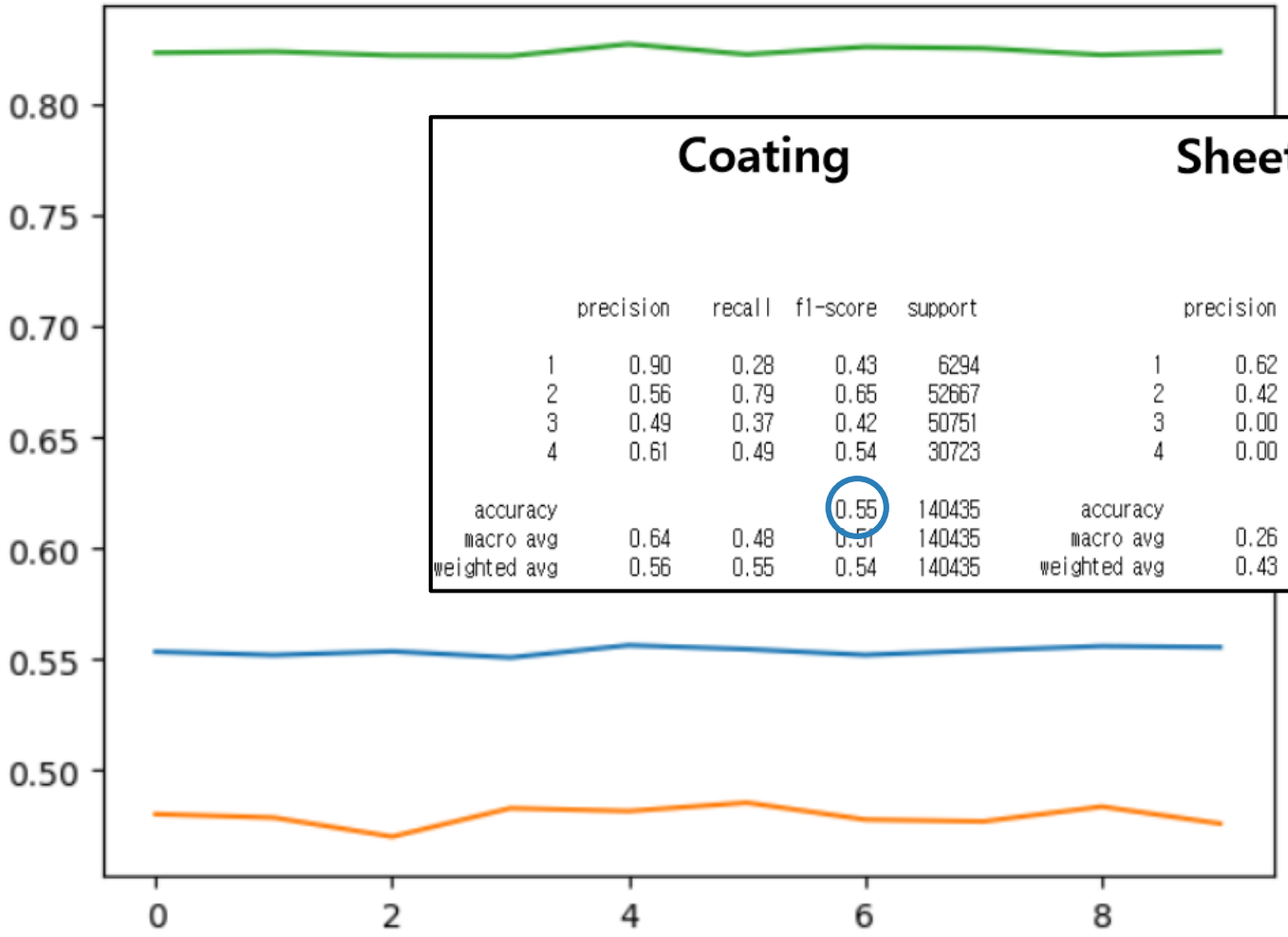
	part	damage	supercategory_name	HQ
0	Bumper	Crushed	VAN	1.79
1	Bumper	Separated	VAN	1.79
2	Bumper	Scratched	VAN	1.79
3	Head lights	Crushed	VAN	0.44
4	Head lights	Separated	VAN	0.44
...	...	...	...	...
563045	Bumper	Crushed	Compact car	2.11
563046	Bumper	Separated	Compact car	2.11
563047	Bumper	Scratched	Compact car	2.11
563048	Wheel	Breakage	Mid-size car	0.55
563049	Wheel	Scratched	Mid-size car	0.55

D. 모델 학습

[3] HQ 판별 모델

수리방법별 공임시간 예측을 위하여 Randomforest를 이용
모델 구축 과정에서 데이터 상 회귀를 사용하는 것이 적합하지 않아 분류로 구축

HQ model accuracy (cv=10)



Coating

	precision	recall	f1-score	support
1	0.90	0.28	0.43	6294
2	0.56	0.79	0.65	52667
3	0.49	0.37	0.42	50751
4	0.61	0.49	0.54	30723
accuracy			0.55	140435
macro avg	0.64	0.48	0.51	140435
weighted avg	0.56	0.55	0.54	140435

Sheet_metal

	precision	recall	f1-score	support
1	0.62	0.44	0.52	30859
2	0.42	0.74	0.53	28515
3	0.00	0.00	0.00	10516
4	0.00	0.00	0.00	2667
accuracy			0.48	72557
macro avg	0.26	0.29	0.26	72557
weighted avg	0.43	0.48	0.43	72557

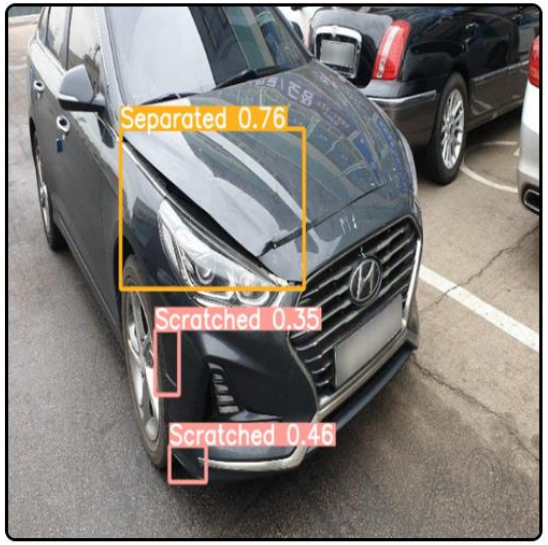
Exchange

	precision	recall	f1-score	support
1	0.88	0.95	0.91	118480
2	0.50	0.29	0.37	17014
3	0.46	0.41	0.43	12793
4	0.00	0.00	0.00	850
accuracy			0.82	149137
macro avg	0.46	0.41	0.43	149137
weighted avg	0.79	0.82	0.80	149137

결과물

5. 결과물

Red Hands Home 회원 가입 회원 전용 서비스 로그인



YOLO

CNN

사이트 이용법

주의사항

※ 사고난 부위를 판별 하기 위해 차량에서 1M 가량 떨어진 후 사진을 1장 찍어 주시기 바랍니다.

1. YOLO 이용방법

- Step1 : YOLO는 여러개의 파손이 있더라도 모두 인식하기 때문에, 전체 부위가 드러나도록 1 장만 입력해주시면 됩니다.

2. CNN 이용방법

- Step1 : 사고난 차량 파손 유형을 판별 하기 위해 사진이 1장 이상 필요 합니다.
- Step2 : 차량에 발생한 파손이 여러개라면, 파손 1개당 1장씩 올려주세요.

아쉬운 점

[1] DATA

A. 기존 데이터셋(JSON)에 있는 모든정보 활용X

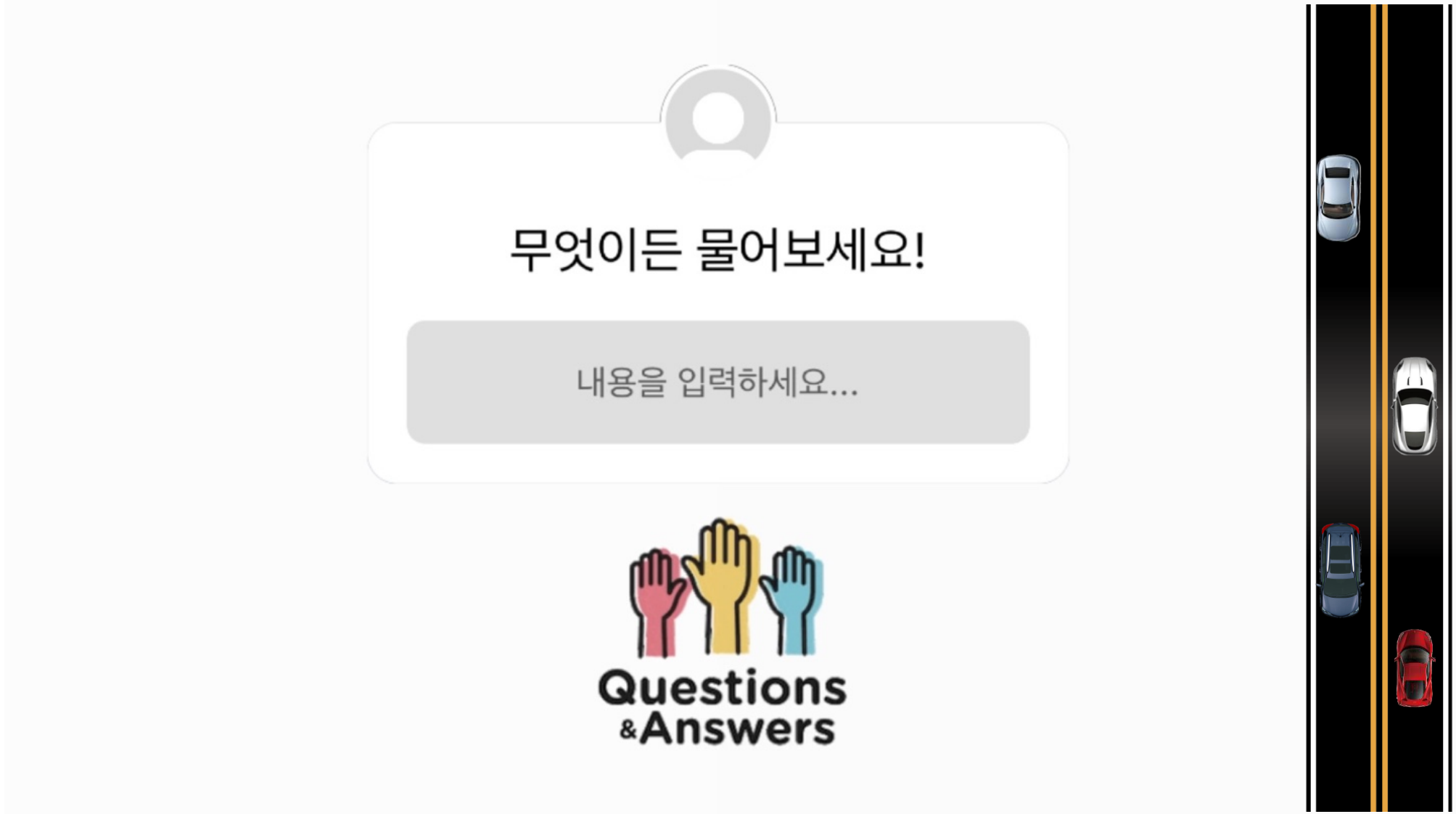
B. 비용 편차

- ▶ 수리공에 따라 발생하는 공임비용 편차가 커 정확한 비용 예측 불가
- ▶ 외제차의 경우 가격 편차로 인한 활용 제한
- ▶ 일부 부품에 대하여 디테일한 가격설정 불가, 프레임 가격만 산출(소모성 부품 등)

[2] MODEL

A. DL/ML

- ▶ 파손부위 분류 모델에서 디테일한 파트 구분 불가(프론트 도어, 리어 도어)
- ▶ 장비의 성능제한으로 인해 20만개 이상의 데이터를 전부 활용하지 못 함


무엇이든 물어보세요!
내용을 입력하세요...
Questions
&Answers