
Team
TRUFFLE

000
팀 소개.

Team TRUFFLE

"땅 속의 다이아몬드"와 같은
데이터 속의 빛나는 인사이트,

컴퓨터공학과 김유철 | 산업정보시스템공학과 문민경
소프트웨어공학과 추은정 | 빅데이터학과 임승현
컴퓨터과학부 전유진 | 전자정보통신공학과 장서윤

떠오르는 취미 생□,

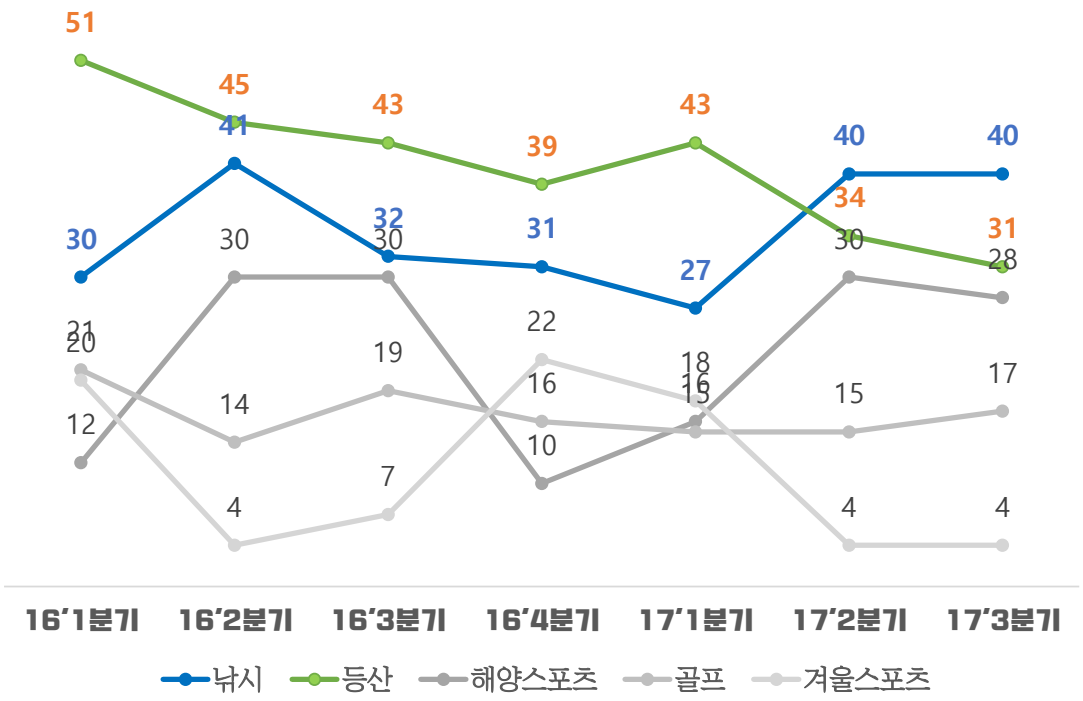
낚시

추1
추2

001
배경 / 시장 환경

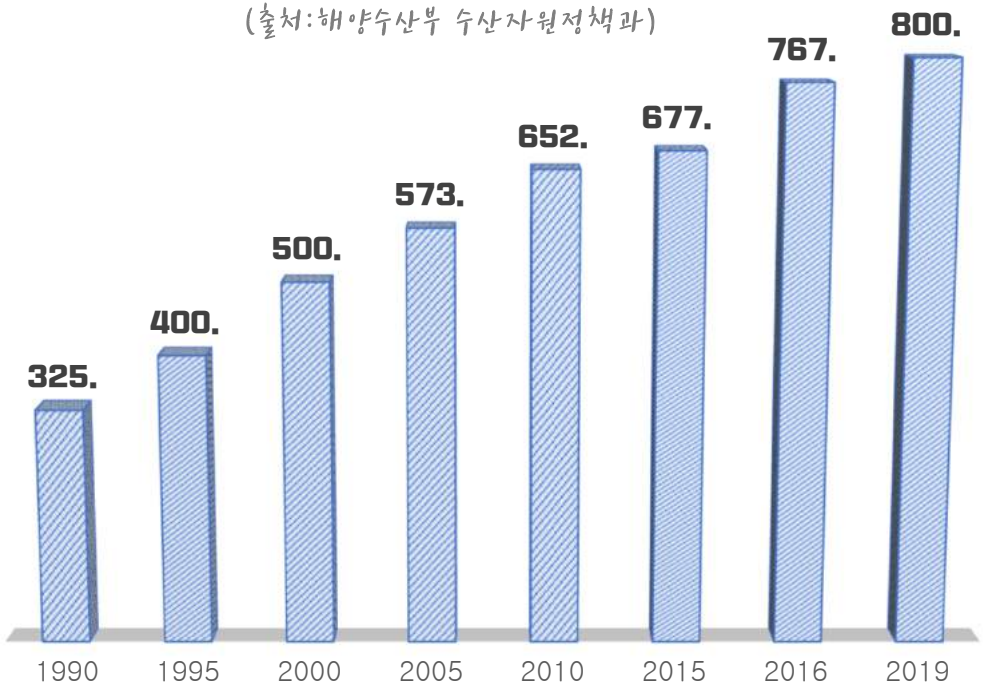
여행시 취미*운동 활동계획(%)

(출처:여행 행태 및 계획 조사-세종대학교 · 컨슈머인사이트 공동기획조사, 복수 응답)



낚시 산업현황-낚시인 현황(만명)

(출처:해양수산부 수산자원정책과)



슬라이드 4

추1 https://www.consumerinsight.co.kr/travel/report1_39.aspx
추은정, 07-24-수요일

추2 http://www.mof.go.kr/statPortal/search/searchList.do
추은정, 07-24-수요일

추1
추2

001
배경 / 시장 환경

월척이다!!!!!!!!

슬라이드 5

추1 https://www.consumerinsight.co.kr/travel/report1_39.aspx
추은정, 07-24-수요일

추2 http://www.mof.go.kr/statPortal/search/searchList.do
추은정, 07-24-수요일

추1
추2

001
배경 / 시장 환경

자랑 좀 하고 싶은데
이게 무슨 물고기
이더라,,,

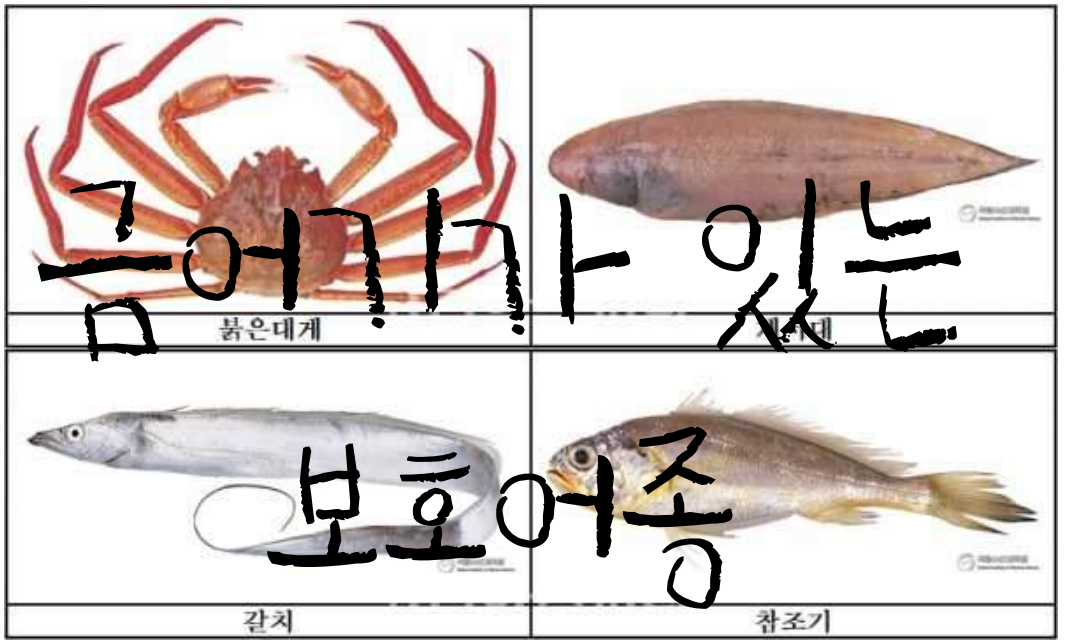
슬라이드 6

추1 https://www.consumerinsight.co.kr/travel/report1_39.aspx
추은정, 07-24-수요일

추2 http://www.mof.go.kr/statPortal/search/searchList.do
추은정, 07-24-수요일

추1
추2

001
배경 / 시장 환경



슬라이드 7

추1 https://www.consumerinsight.co.kr/travel/report1_39.aspx
추은정, 07-24-수요일

추2 http://www.mof.go.kr/statPortal/search/searchList.do
추은정, 07-24-수요일

추1
추2

001
배경 / 시장 환경

"Follow, follow me"

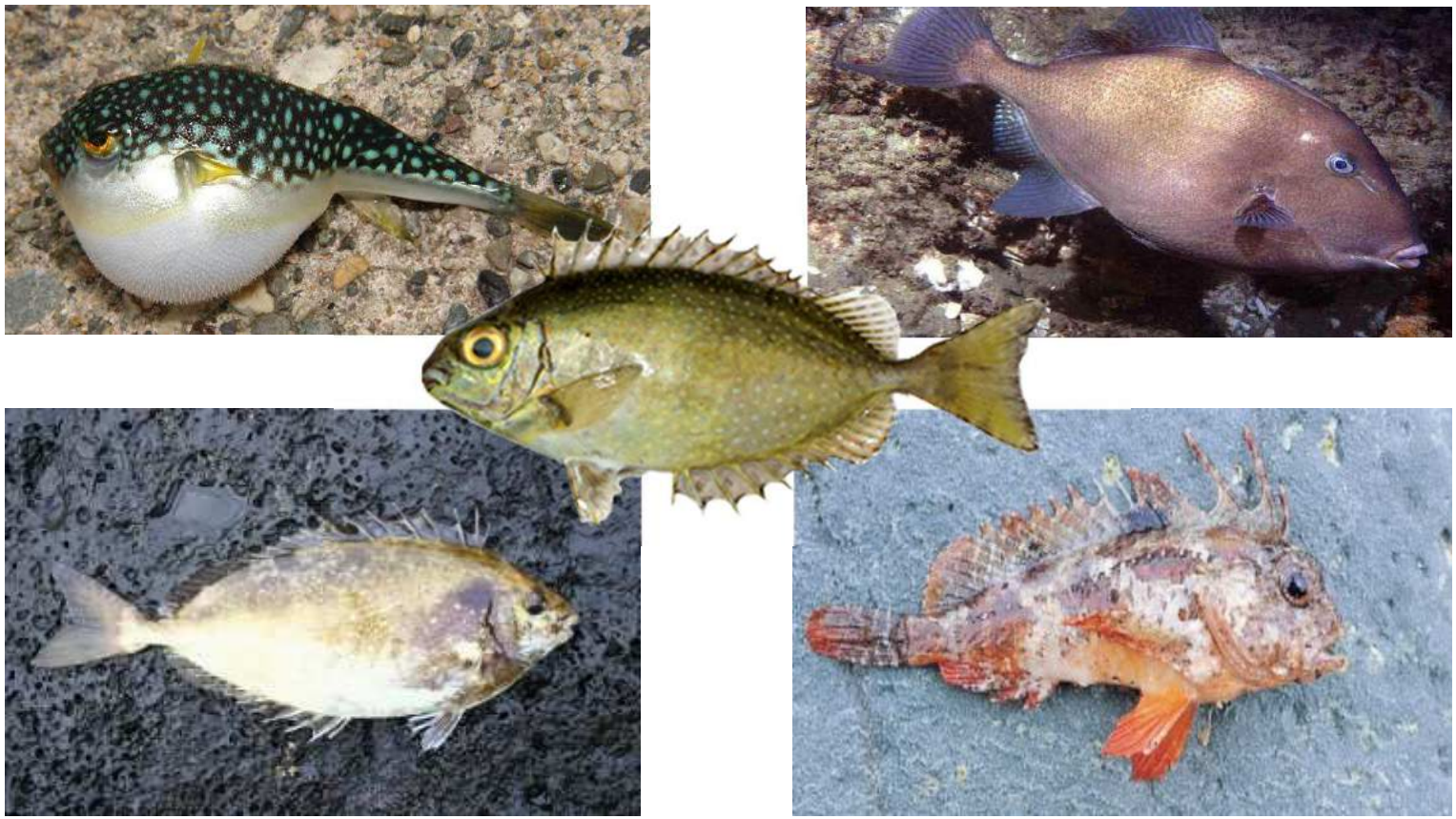
슬라이드 8

추1 https://www.consumerinsight.co.kr/travel/report1_39.aspx
추은정, 07-24-수요일

추2 http://www.mof.go.kr/statPortal/search/searchList.do
추은정, 07-24-수요일

추1
추2

001
배경 / 시장 환경



슬라이드 9

추1 https://www.consumerinsight.co.kr/travel/report1_39.aspx
추은정, 07-24-수요일

추2 http://www.mof.go.kr/statPortal/search/searchList.do
추은정, 07-24-수요일

추1
추2

001
배경 / 시장 환경

위기탈출 넘버원

독 있는 물고기는
무심코 만져서도 먹어서도
안됩니다.

슬라이드 10

추1 https://www.consumerinsight.co.kr/travel/report1_39.aspx
추은정, 07-24-수요일

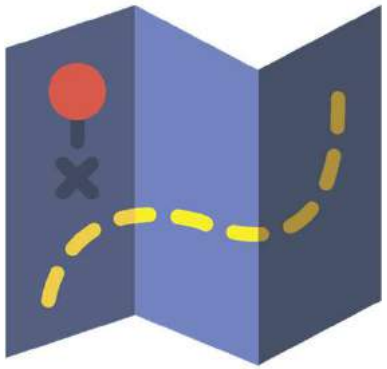
추2 http://www.mof.go.kr/statPortal/search/searchList.do
추은정, 07-24-수요일

002
구축모델 / 핵심 적용 사□

漁 夫 地理

어 부 지 리

고기 잡을 어 아비 부 땅 지 다스릴 리



002
구축모델 / 핵심 적용 사□

진 어 개

짜 부 이

득

어부지리'S 주요 기능

어종 분류

(보호 어종/위□ 어종)

지역정보 결합

(추천 어종)

사용자 흥미유발

(개인 어□/포인트 적립)

002
구축모델 / 핵심 적용 사□

이미지 크롤링, 정제

DeepLearning 설계

YOLO학습

Classification 학습

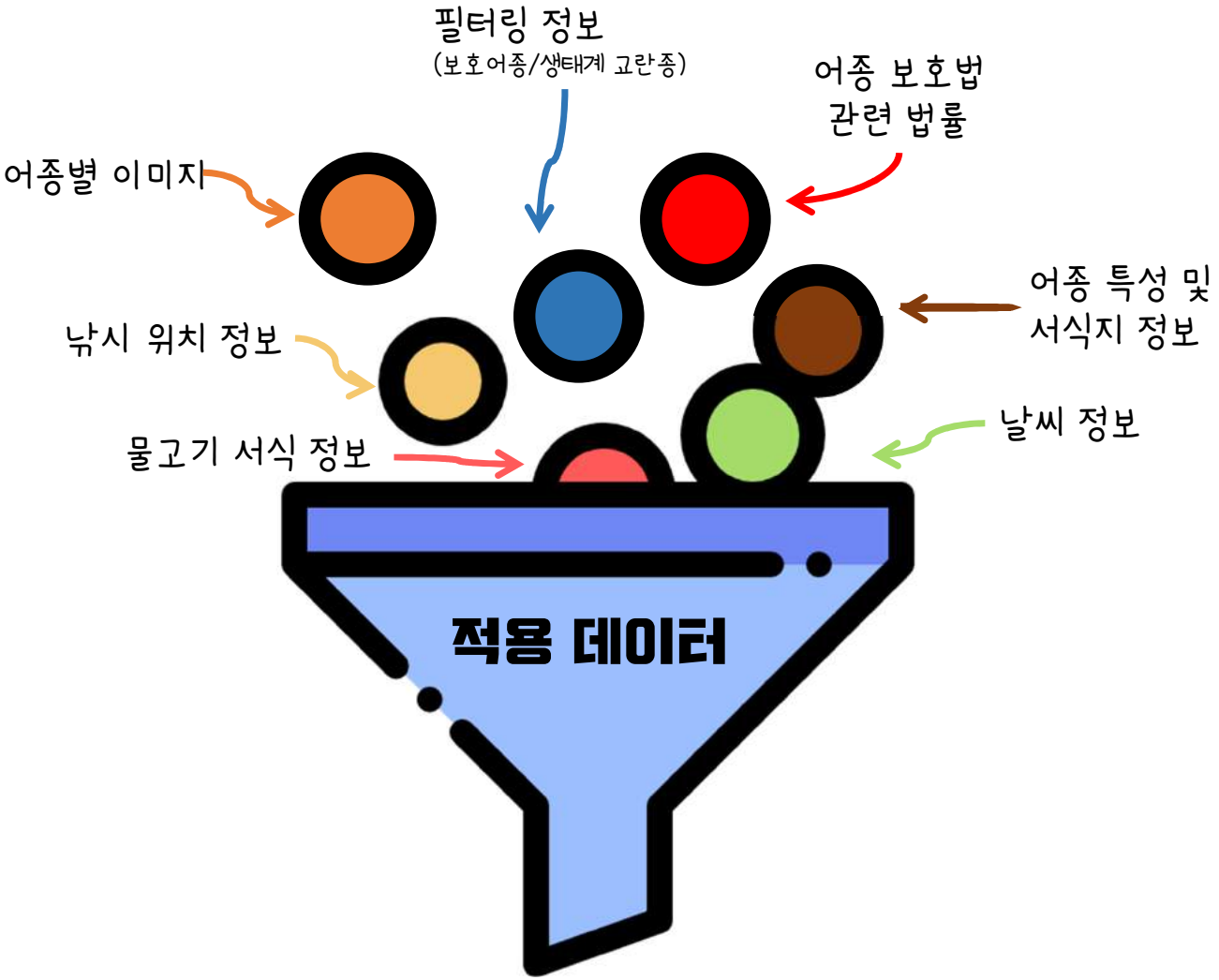
Spring MVC service 구성, DB설계

서비스 구현

화면 구성

TEAM TRUFFLE SCHEDULING

002
구축모델 / 핵심 적용 사□



002
구축모델 / 핵심 적용 사□

어부지리 사용 툴

★데이터 수

OpenCV

python™

★데이터 석-처리

colab

TensorFlow

real time object detection

YOLO

"You Only Look Once"

LabelImg

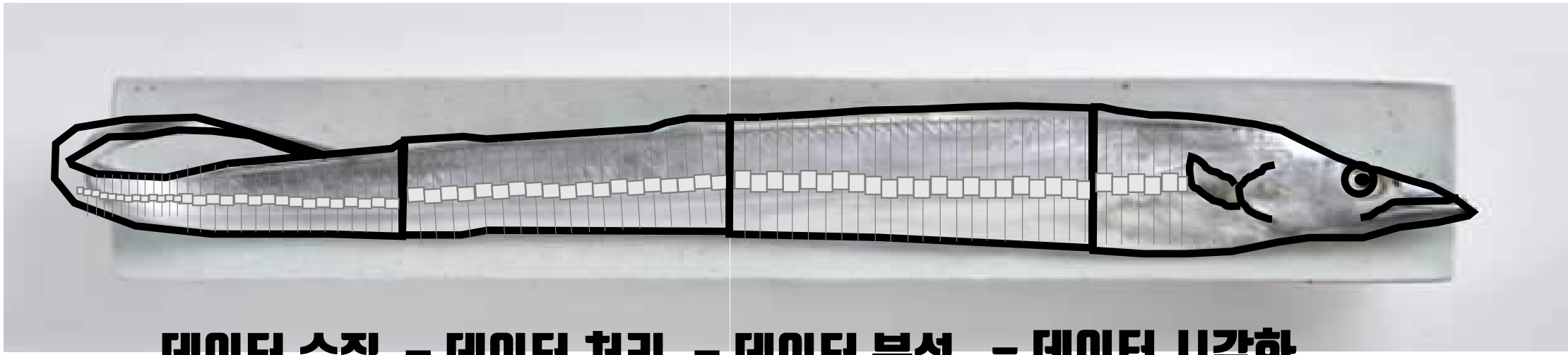
python™

★ +DB

spring boot

aws

MySQL™



데이터 수집 - 데이터 처리 - 데이터 분석 - 데이터 시각화

이미지 크롤링

크롤링/검색

이미지 트랙킹
(OpenCV)

Yolo detection

Data
Annotation

BBCrop

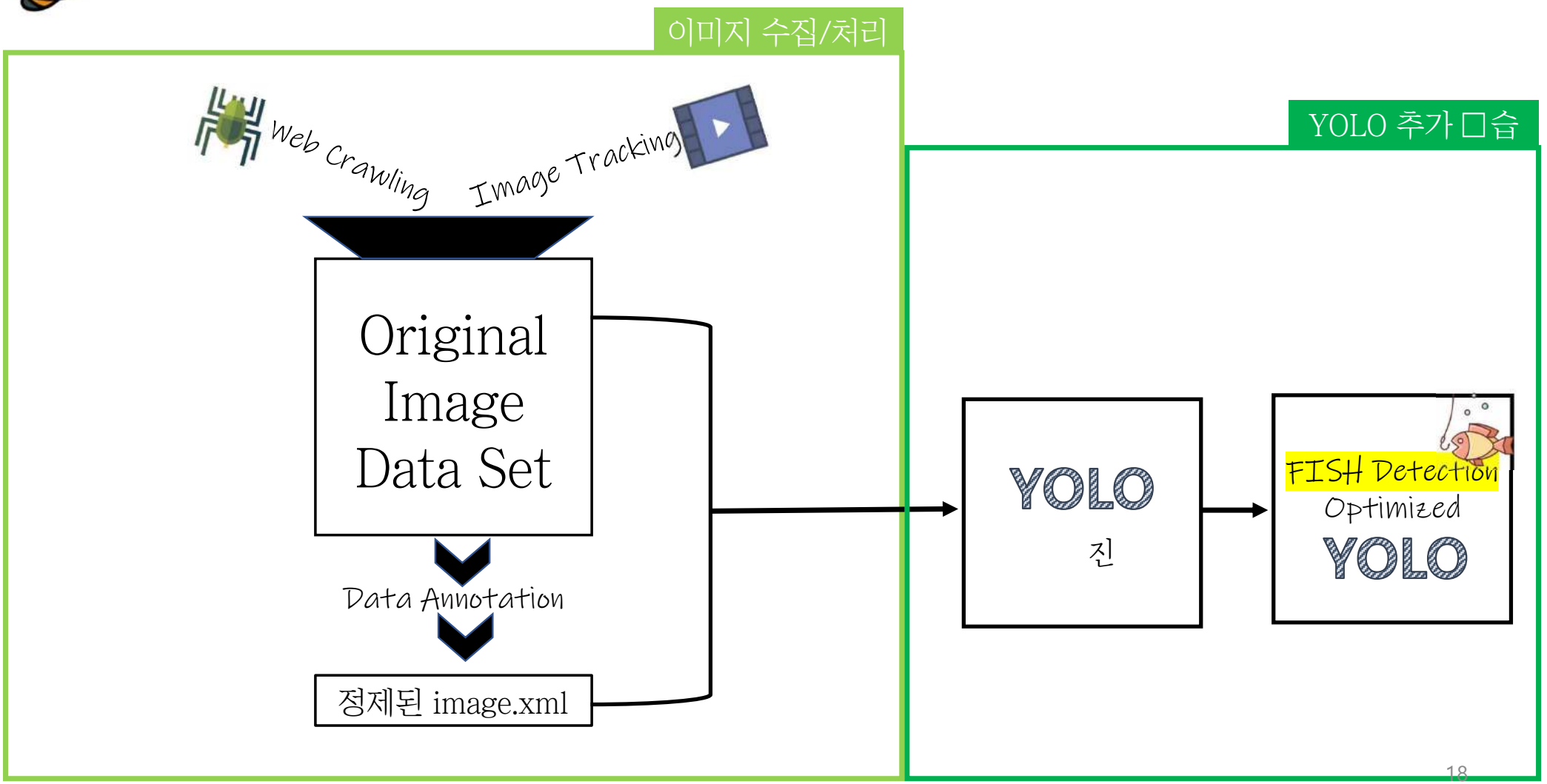
Data 2차 가공

이미지 분류
(CNN)

어종 분류 결과
▶어종 정보 제공

EXIF를 이용한
GPS 메타 데이터
시각화

구축모델 / 핵심 적용 사□


이미지 수집/처리
Web Crawling
Image Tracking
Original Image Data Set
Data Annotation
정제된 image.xml
YOLO 추가 □습
YOLO
진
FISH Detection
Optimized
YOLO

학습 어종 선정

금어기 어종

위험 어종

낚시 인기 어종

002
구축모델 / 핵심 적용 사□

이미지 분석

INPUT

Original Image Data Set

Data Augmentation

YOLO

<Fish Detection>

FISH

Image BBcrop

"무엇이 물고기인지" Detection

Fish 추출된 Jpg image

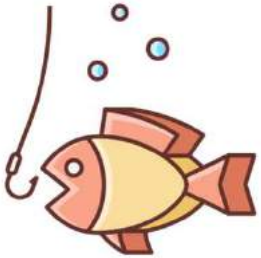
"어떤 물고기인지" Classification

CNN

OUTPUT

002
구축모델 / 핵심 적용 사□

Today's Pick
어종 추천 시스템



낚시 성공!

Classification
어종 분류 시스템

My Fishbowl
개인 어□ 시스템

Ranking
포인트 적립-랭킹 제도

#어부지리 실행화면

#실행영상


어부지로!
카카오 로그인
©COPYRIGHT DATA CAMPUS TEAM TRUFFLE
https://kauth.kakao.com/oauth/authorize?client_id=a3f9cdf6ff350244d4e4e19401521bfd&r...

#NOTICE

#MAIN

#recommendation

추은정 WELCOME! | LOGOUT

포인트

연동 방파제

서해
영종도방조제 비상주차공간주면 석축
삼목수문
삼목 선착장 방파제
연동 방파제

수온

19.6 (℃)

기상

날씨 | 오전 : 구름많음 / 오후 : 맑음
풍향 | 오전 : 서-북서 / 오후 : 남서-서
파고 | 0.5~0.6 (M)

물때

02:49 (802) ▲+609 15:08 (728) ▲+486
09:22 (242) ▼-560 21:24 (121) ▼-607

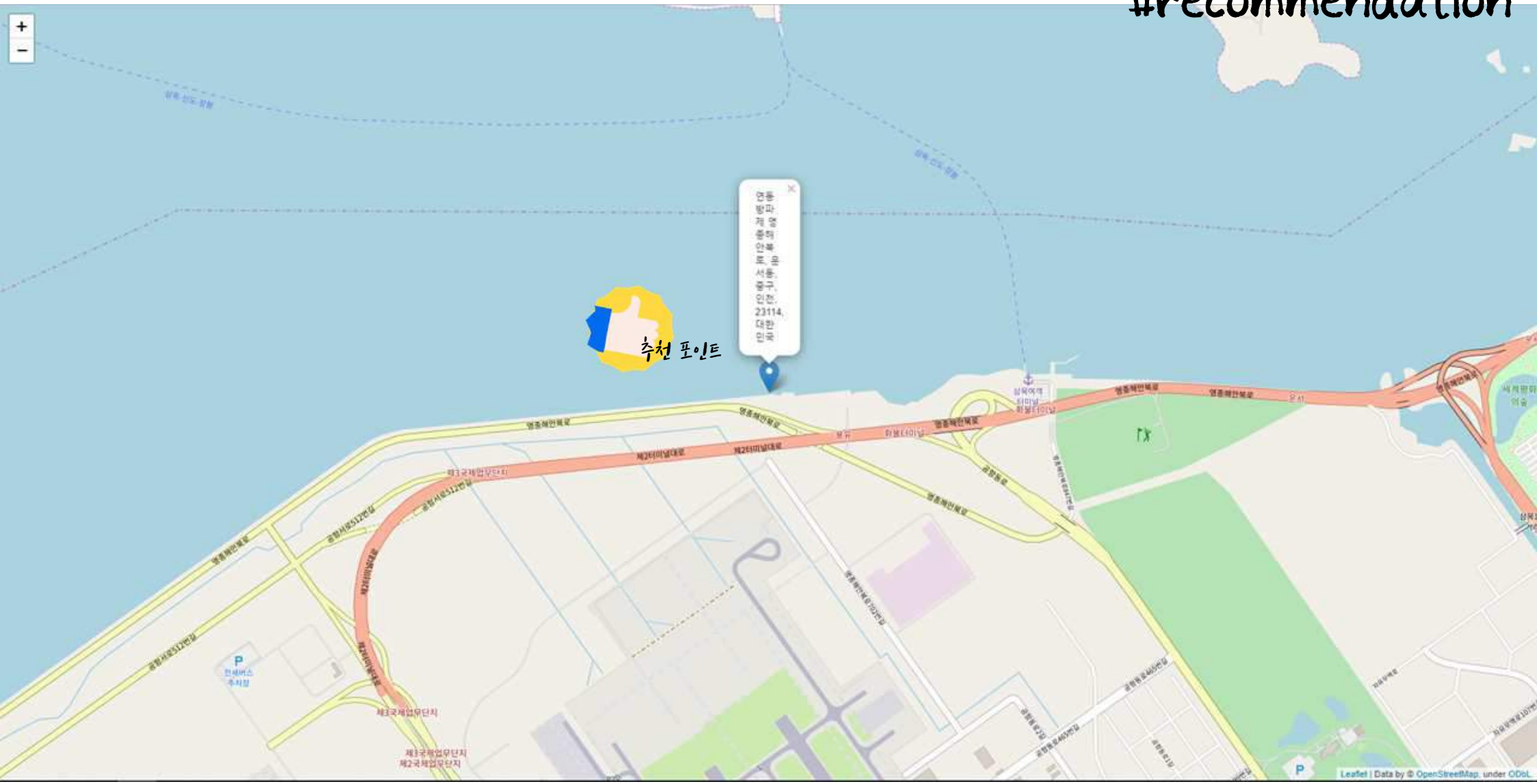
주의사항

고급도 남측에 천소수심 3.4M의 암반과 주변으로 이상체가 다수 존재하며, 섬 주변으로 양식장이 존재하므로 운항하는 선박은 주의하여야 한다.

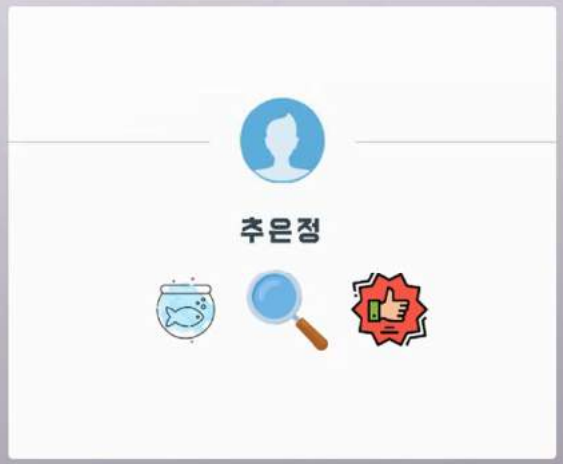
추천어종

감성돔, 붕장어, 숭어

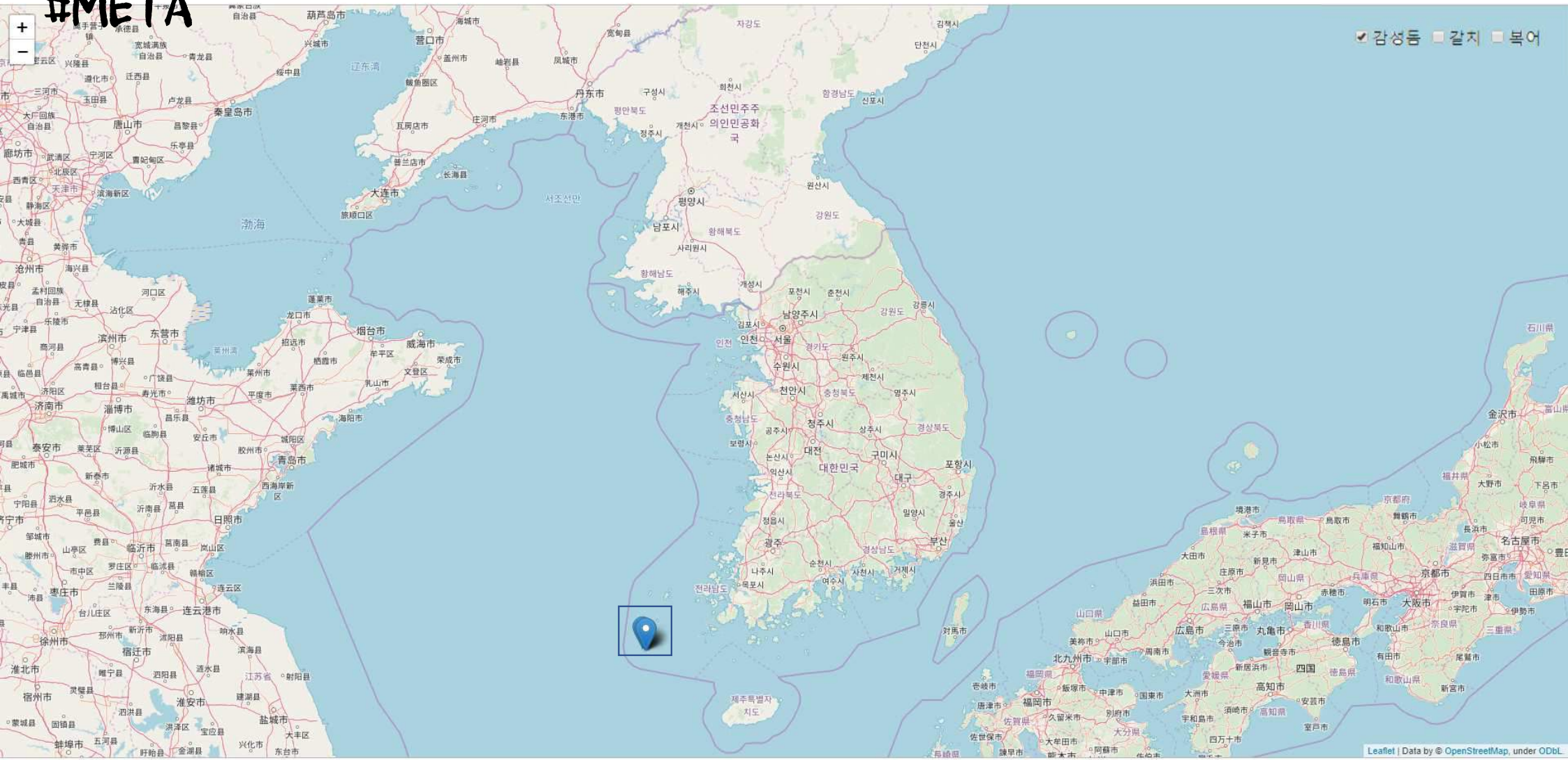
#recommendation



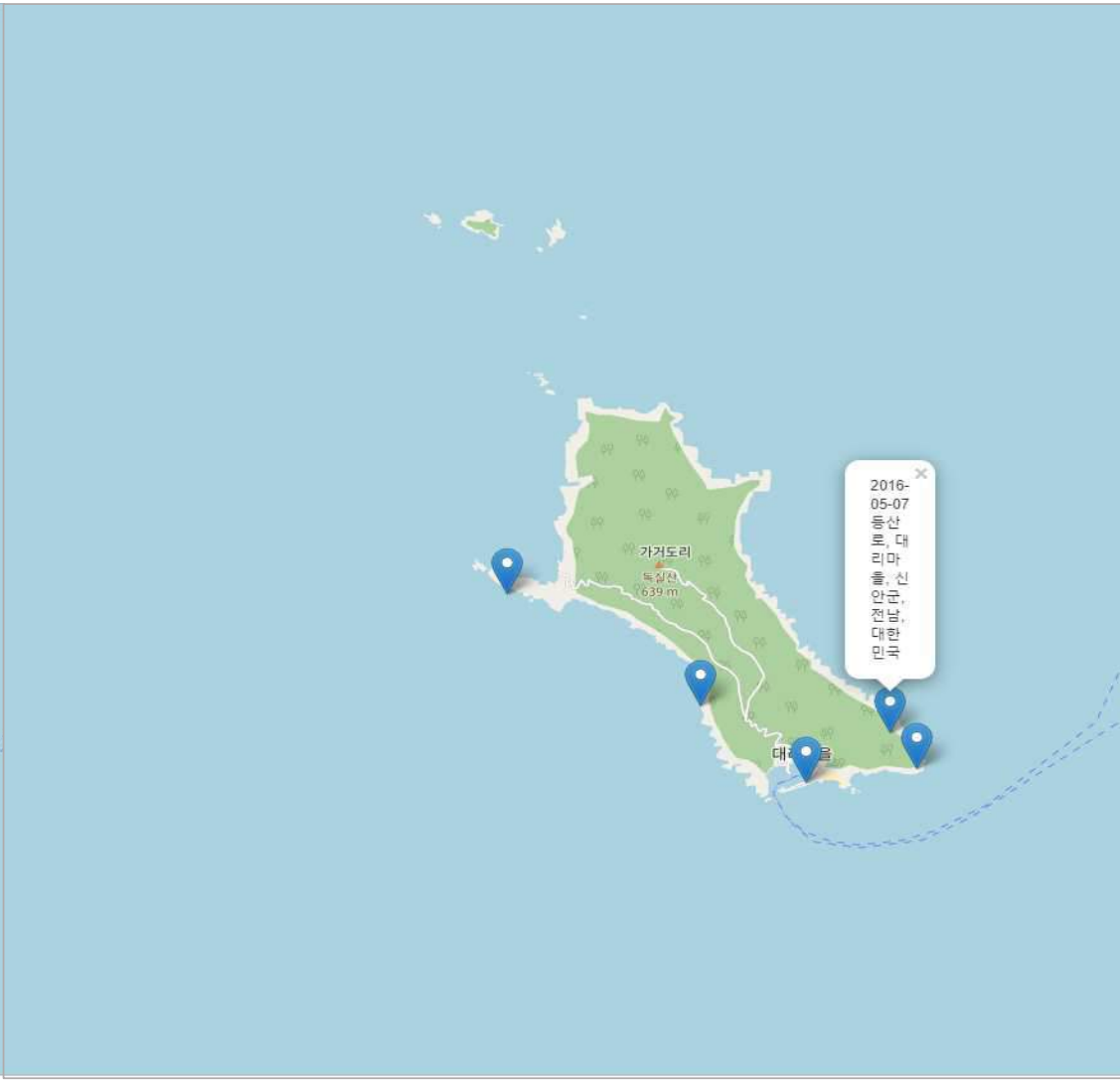
#실행영상



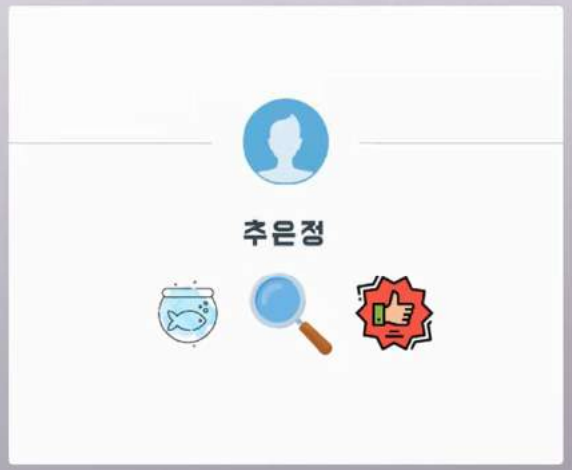
#META



#META



#실행영상



003
한계점 및 보완점

!한계점 및 해결 방안!

SERVICE

-추천 : 정보 제공 사이트 부족, 크롤링 어려움

-개인 어□ : 메타 데이터가 없는 사진에 대□ 서비스 방법

-낚시에 대□ 객관적인 도메인 정보 부족

CLASSIFICATION

-부적절□ □습 상태의 대량 데이터

Annotation, BBCrop사용하여 적절□ □태로 가공

-딥러닝 □습하기에 부족□ 데이터 수

Augmentation, 여러 사이트 크롤링, Tracking, Transfer learning등 이용

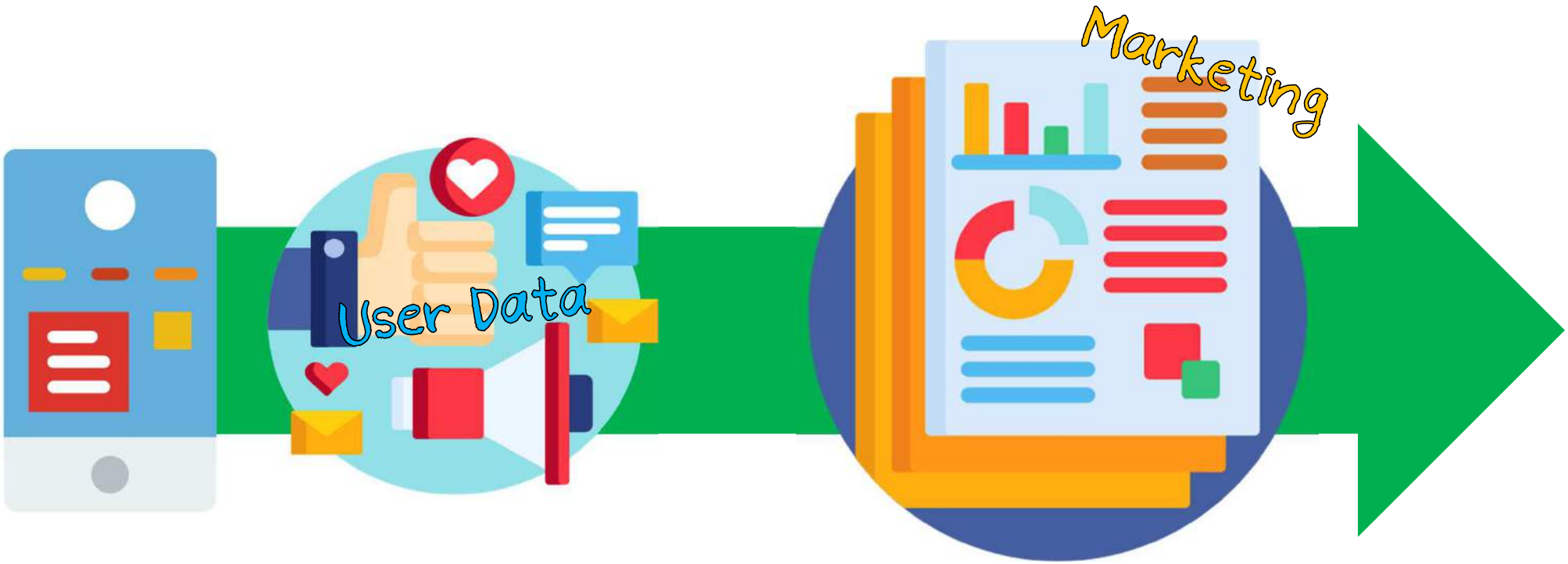
-데이터 분석의 정□도 저조

다양□ 모델, 조건 실□하여 비교

->최적의 결과를 찾아냄

낚시꾼에게
필요하거나 주의해야할
어종 정보를
앱 하나로 쉽게 알 수 있음

004
기대효과



어부지리를 통해 수집된 사용자 이용 정보
→ **낚시** 관련 빅데이터 분석 가능

들어주셔서
감사합니다!

자료 출처 

- 수집한 데이터 출처
  - 이미지 : Goolge Icrawler (구글,bing)
  - 어종 관련 정보 : 국립수산과학원 수산생명정보센터

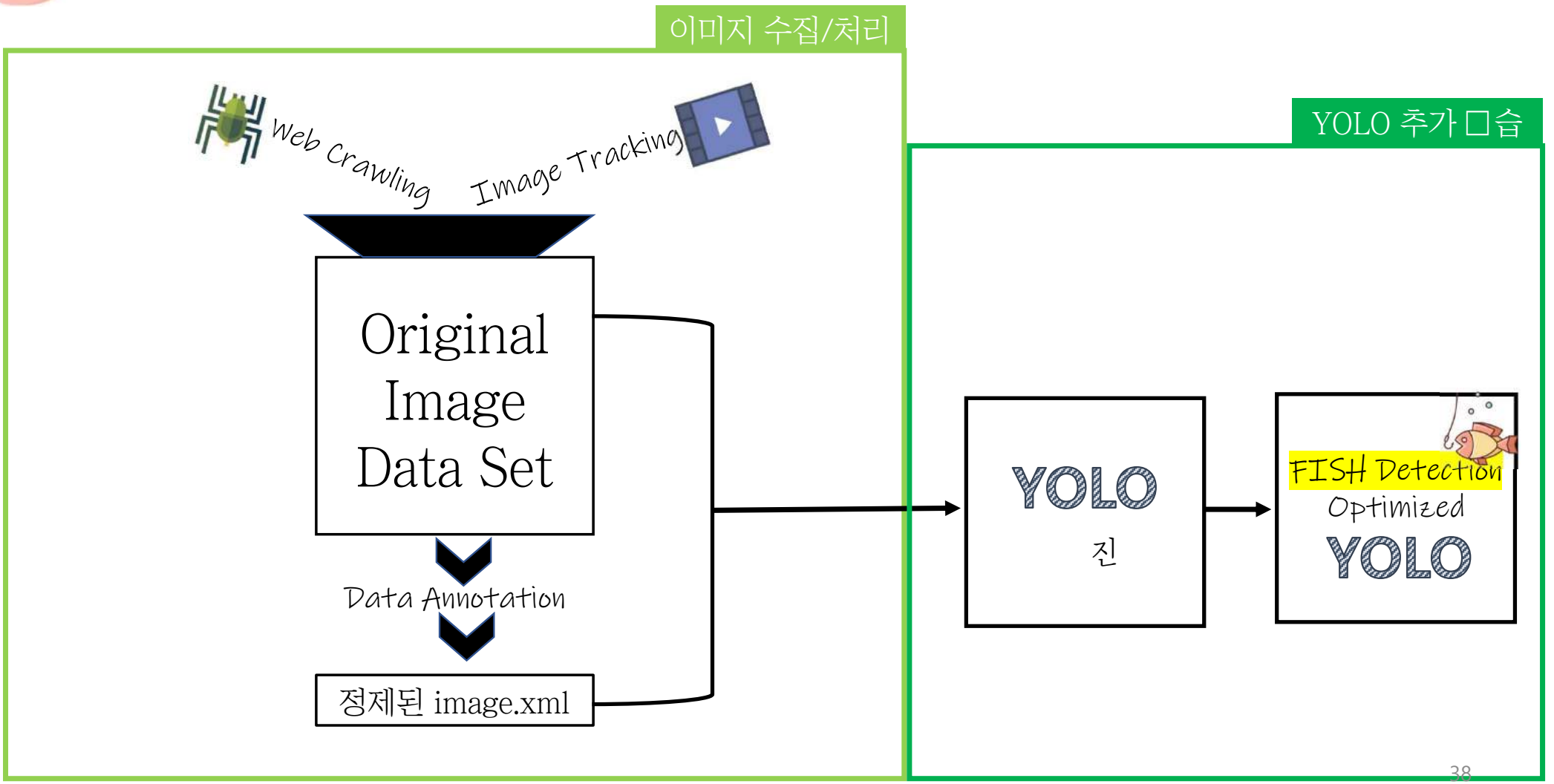
    http://www.nifs.go.kr/frcenter/?
  - 낚시 포인트 및 지역정보 : 바다타임닷컴

    https://www.badatime.com/?
  - 어종 관련 법률 : 수산자원관리법

    http://www.yeslaw.com/lims/front/page/fulltext.html?pAct=view&pLawId=4844

Team
Truffle의
별책 부록!

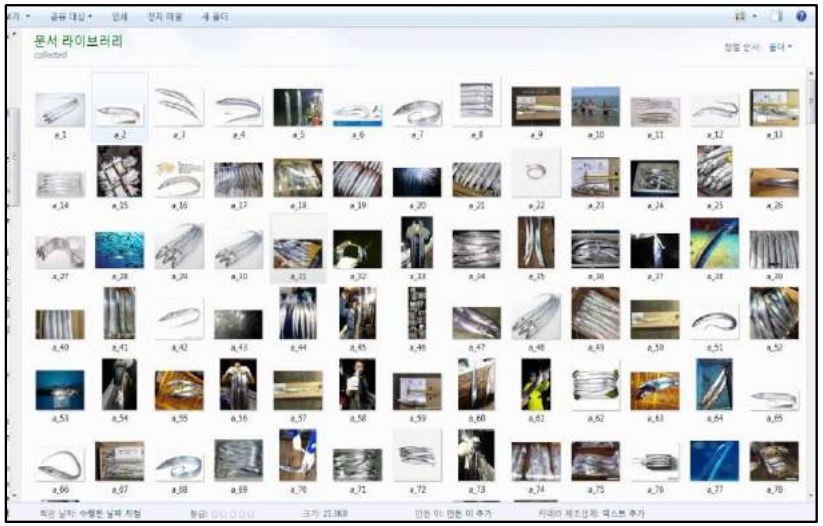
구축모델 / 핵심 적용 사□


이미지 수집/처리
Web Crawling
Image Tracking
Original Image Data Set
Data Annotation
정제된 image.xml
YOLO 추가 □습
YOLO
진
FISH Detection
Optimized
YOLO

데이터 상세 설명

ⓐ데이터 수집

이미지 크롤링 – 크롤링/검색 – 이미지 트랙킹(OpenCV)



□□지 크롤□

□□지 트랙□

▶Google, Bing 사이트에서 icrawler모듈을 통해서 각 □목별로 사진 1000장을 크롤링

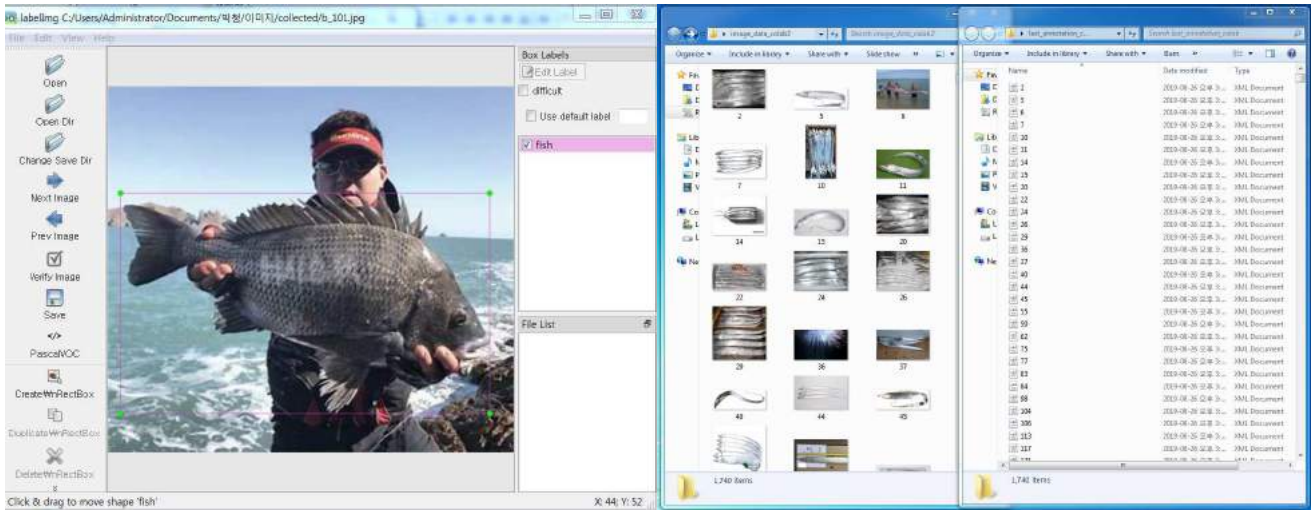
▶ 국립수산과□원-수산생명자원정보센터에서 어종들의 정보를 □목별로 크롤링

▶ OpenCV를 이용해서 동영상에서 프레임마다 물고기를 찾아서 **time step**별 이미지 추출

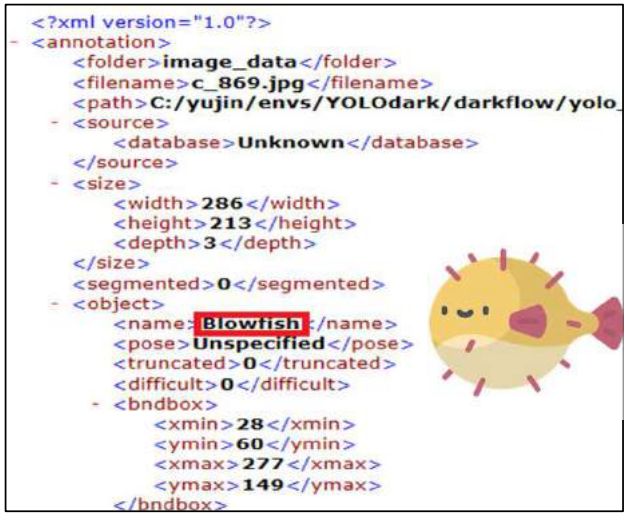
데이터 상세 설명

ⓑ데이터 처리_1차

YOLO + BBCrop + python코드로 정제



〈labelImg 실행 과정〉



Annotation(.xml)

▶ labelImg 툴을 사용하여 annotation 수□ 및 파일 생성

▶ Annotation 실□□ 결과 xml□식 파일+ 원본파일(.jpg)를 사용해 yolo□습 (FISH detecting에 적□□ yolo로 만들었음)

데이터 상세 설명

ⓑ데이터 처리_2차

Annotation 결과 file(.xml) 파일과

원본 파일을 python코드로

학습에 적 한 태로 2차 정제

□류□거 코드(python)

```
In [ ]: #-*- coding:utf-8 -*-
import xml.etree.ElementTree as ET
import os
from xml.etree.ElementTree import Element, SubElement, Comment, tostring, ElementTree, dump, parse
import requests

url = 'http://finance.naver.com/' #euc-kr (op949)
res = requests.get(url)
res.encoding = 'euc-kr' #한글 깨지지 않음
def find_error3(files_path_xml, files_path_jpg):   #파일 하나씩 가져오기
    file_list_1 = os.listdir(files_path_xml)
    file_list_2 = os.listdir(files_path_jpg)
    file_list_jpg = [file for file in file_list_2 if file.endswith(".jpg")]
    file_list_xml = [file for file in file_list_1 if file.endswith(".xml")]

    for i in range(len(file_list_xml)):#파일 하나씩 오픈한다
        doc = parse(files_path_xml +'/' + file_list_xml[i])#file parsing해오기
        #print(file_list_xml[i] + ',  ' + str(doc.find("filename")))
        size_tag = doc.find("size")
        width_tag = size_tag.find("width")
        height_tag = size_tag.find("height")
        depth_tag = size_tag.find("depth")

        image_xml_name = file_list_xml[i].split('.')
        image_jpg_name_1 = file_list_jpg[i].split('.')
        result_image_jpg_name = image_xml_name[0] + ".jpg"
        image_jpg_name = image_xml_name[0]
        image_jpg_name = doc.find("filename")

        if(image_xml_name[0]!=image_jpg_name_1[0]):#jpg랑 xml이 다른이름이 있으면 출력
            print(file_list_xml[i])
            print(file_list_jpg[i])

        if(width_tag.text=="0" or height_tag.text=="0"or depth_tag.text=="0"): #파일 삭제하던가/ 파일이름 print하기
            print("C:/yujin/envs/YOLOdark/darkflow/yolo_train_data_set/blowfish_annotation/" + file_list_xml[i])
            os.remove("C:/yujin/envs/YOLOdark/darkflow/yolo_train_data_set/blowfish_annotation/" + file_list_xml[i])
            os.remove("C:/yujin/envs/YOLOdark/darkflow/yolo_train_data_set/blowfish_image_data/" + result_image_jpg_name)

        if(result_image_jpg_name==image_jpg_name.text):
            m=0
        else:
            print(image_jpg_name.text)
            print(file_list_xml[i])

        root = doc.getroot()#xml 문서의 최상단 루트태그를 가리킨다
        bndbox_tag = doc.findall(".//bndbox")
        path_tag = doc.find("path")

        temp = len(bndbox_tag)
        for j in range(temp):
            xmin = bndbox_tag[j].find("xmin")
            xmax = bndbox_tag[j].find("xmax")
            ymin = bndbox_tag[j].find("ymin")
            ymax = bndbox_tag[j].find("ymax")
            if(int(xmin.text)>int(xmax.text) or int(ymin.text)>int(ymax.text)):
                print(len(bndbox_tag))
                print(file_list_xml[i])
    print("done")

find_error3("C:/yujin/envs/YOLOdark/darkflow/yolo_train_data_set/blowfish_annotation","C:/yujin/envs/YOLOdark/darkflow/yolo_train_data_s
```

데이터 상세 설명

ⓑ데이터 처리_2차

Img 파일의 tag_name 변경

학습 데이터에 적 하게 가공

```
1
c_13.jpg
C:/yujin/envs/YOLOdark/darkflow/yolo_train_data_set/blowfish_image_da
ta/c_13.jpg
--------------------
5
c_130.jpg
C:/yujin/envs/YOLOdark/darkflow/yolo_train_data_set/blowfish_image_da
ta/c_130.jpg
--------------------
1
c_131.jpg
C:/yujin/envs/YOLOdark/darkflow/yolo_train_data_set/blowfish_image_da
ta/c_131.jpg
--------------------
1
c_134.jpg
C:/yujin/envs/YOLOdark/darkflow/yolo_train_data_set/blowfish_image_da
ta/c_134.jpg
--------------------
```

□과□

```python
import xml.etree.ElementTree as ET
import os
from xml.etree.ElementTree import Element, SubElement, Comment, tostring, ElementTree, dump
import requests

url = 'http://finance.naver.com/' #euc-kr (cp949)
res = requests.get(url)
res.encoding = 'euc-kr' #한글 깨지지 않음

def ChangeTagName(files_path):    #파일 하나씩 가져오기
    file_list = os.listdir(files_path)
    file_list_xml = [file for file in file_list if file.endswith(".xml")]

    for i in range(len(file_list_xml)):#파일 하나씩 오픈한다
        name_tag = doc.findall(".//name")
        path_tag = doc.find("path")
        foldername_tag = doc.find("folder")
        filename_tag = doc.find("filename")        #dump(path_tag)
        image_jpg_name = file_list_xml[i].split('.')
        result_image_jpg_name = image_jpg_name[0] + ".jpg"
        print(len(name_tag))
        temp = len(name_tag)
        for k in range(temp):
            name_tag[k].text = "fish"

        path_tag.text = "C:/yujin/envs/YOLOdark/blowfish_data/" + result_image_jpg_name
        foldername_tag.text = "image_data"
        filename_tag.text = result_image_jpg_name
        print(filename_tag.text)

        doc.write(files_path + '/' + file_list_xml[i])
        print(path_tag.text)
        print("----------------------")

ChangeTagName("C:/yujin/envs/YOLOdark/darkflow/yolo_train_data_set/blowfish_annotation")
```

데이터 상세 설명

ⓑ데이터 처리_2차

□□Crop

정제 후 물고기 데이터만 잘라
사용하기 위해 이미지 가공

<BBCrop 예시>

```
import os
import json
import cv2
 def searchByLableName(param_json_files_path, param_label_name):
    def im_trim(img, t_x, t_y, b_x, b_y): #함수로 만든다
         if img is None:
            return
         img_trim=img[t_y:b_y, t_x:b_x]
            return img_trim #필요에 따라 결과물을 리턴
        json_jpg_files_path = param_json_files_path # path = "./out/"
        file_list = os.listdir(json_jpg_files_path)
        file_list_json = [file for file in file_list if file.endswith(".json")]
        searched_list=[]
        for i in range(len(file_list_json)):
            json_name_with_path=json_jpg_files_path+'/'+file_list_json[i]  # print(json_name_with_
            with open(json_name_with_path) as json_file:    # print("")
            json_data = json.load(json_file)        # print(json_data)

            search_label_dict = (item for item in json_data if item['label'] == param_label_name
            next_search_label_dict = next(search_label_dict, False)
            if next_search_label_dict is False:
                    continue
            image_jpg_name=file_list_json[i].split('.')
            next_search_label_dict["imagename"]=image_jpg_name[0]+".jpg"

            save_path=json_jpg_files_path+"/"+param_label_name
            print(json_jpg_files_path+"/"+param_label_name)
            if not os.path.isdir(save_path):
                  os.mkdir(save_path)

            for l in range(len(searched_list)):
                image_name_to_be_cropped=searched_list[l]["imagename"]
                image_name_to_be_cropped_t_x=searched_list[l]["topleft"]["x"]
                image_name_to_be_cropped_t_y=searched_list[l]["topleft"]["y"]
            print(json_jpg_files_path+"/"+image_name_to_be_cropped)
            org_image = cv2.imread(json_jpg_files_path+"/"+image_name_to_be_cropped)
            trim_image = im_trim(org_image,image_name_to_be_cropped_t_x, image_name_to_be_cropped_
            # ★★★★★★★★ param_label_name
            cv2.imwrite("{0}/{1}{2}.jpg".format(save_path, param_label_name, l), trim_image) #이미

searchByLableName("F:/TrufflePig/out", "bird")
```

데이터 상세 설명

ⓑ데이터 처리_2차

yolo로 fish를 찾아낸 결과가 저장된 json파일로
물고기가 있는 부분의 사진만을 따로 저장

<Json 저장 예시>

```
def searchByLableName(param_json_files_path, param_label_name):
    def im_trim(img, t_x, t_y, b_x, b_y):
        if img is None:
            return
        img_trim=img[t_y:b_y, t_x:b_x]
        return img_trim

    json_jpg_files_path = param_json_files_path
    # path = "./out/"
    file_list = os.listdir(json_jpg_files_path)
    file_list_json = [file for file in file_list if file.endswith(".json")]

    searched_list=[]
    for i in range(len(file_list_json)):
        json_name_with_path=json_jpg_files_path+'/'+file_list_json[i]

        with open(json_name_with_path) as json_file:
            json_data = json.load(json_file)
            search_label_dict = (item for item in json_data if item['label'] == param_label
            next_search_label_dict = next(search_label_dict, False)

            if next_search_label_dict is False:
                continue

            image_jpg_name=file_list_json[i].split('.')
            next_search_label_dict["imagename"]=image_jpg_name[0]+".jpg"
            searched_list.append(next_search_label_dict)

    save_path=json_jpg_files_path+"/"+param_label_name
    print(json_jpg_files_path+"/"+param_label_name)
    if not os.path.isdir(save_path):
       os.mkdir(save_path)

    for l in range(len(searched_list)):
        image_name_to_be_cropped=searched_list[l]["imagename"]
        image_name_to_be_cropped_t_x=searched_list[l]["topleft"]["x"]
        image_name_to_be_cropped_t_y=searched_list[l]["topleft"]["y"]
        image_name_to_be_cropped_b_x=searched_list[l]["bottomright"]["x"]
        image_name_to_be_cropped_b_y=searched_list[l]["bottomright"]["y"]

        print(json_jpg_files_path+"/"+image_name_to_be_cropped)
        org_image = cv2.imread(json_jpg_files_path+"/"+image_name_to_be_cropped)
        trim_image = im_trim(org_image,image_name_to_be_cropped_t_x, image_name_to_be_cropp
        cv2.imwrite("{0}/{1}{2}.jpg".format(save_path, param_label_name, l), trim_image)


searchByLableName("F:/TrufflePig/out", "knife")
```

데이터 상세 설명

ⓒ데이터 분석

□□**Crop**을 거친 이미지를 **yolo**로 분석

*output파일: 예상되는 물고기 종류(이름) + 그 종일 확률(%)

←output 이미지

▼ array [2]
  ▼ 0 {4}
    label : BlackPorgy
    confidence : 0.97
    ▼ topleft {2}
      x : 346
      y : 3
    ▼ bottomright {2}
      x : 479
      y : 377
  ▼ 1 {4}
    label : BlackPorgy
    confidence : 0.77
    ▼ topleft {2}
      x : 58
      y : 0
    ▼ bottomright {2}
      x : 305
      y : 448

yolo □□ json파□

데이터 상세 설명

@데이터 시각화

어종 분류 결과 → 어종 정보 제공

크롤링한 어종 정보 디비(rds_mysql)에 저장

분류 결과에 대한 어종 이름으로 데이터 가져와 화면에 출력

id	name	scientific_name	habitat	toxicity	distribution	prohibition_length	spawn	growth	fish_migration	feeding	color
1	감성돔	Acanthopagr...	수심 50...	0	우리나...	20센티미터 이하	산란...	부화...	내만성 어종...	잡식...	몸...

```
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://fisherman-geography.chkcdwxujprm.ap-northeast-2.rds.amazonaws.com/fisherman_geography
spring.datasource.username=[redacted]
spring.datasource.password=[redacted]
mybatis.type-aliases-package=team.truffle.VO

spring.servlet.multipart.max-file-size=6MB
spring.servlet.multipart.maxRequestSize=6MB

#server.port=8081
```

D□□□ □□

```
@RequestMapping(value = "/fishImageDisplay", method = RequestMethod.GET)
public void showImage(@RequestParam("imageId") String imageId,HttpServletResponse response,HttpServletRequest request)
        throws ServletException, IOException{


    Map<String,Object> userIMG =fileService.getFishByteImage(imageId);
    System.out.println("디비에서 불러오는 거 성공");
    byte[] userImg=(byte[])userIMG.get("img");
    System.out.println("바이트배열로 변경 성공");
    response.setContentType("image/jpeg, image/jpg, image/png, image/gif");
    InputStream inputStream = new ByteArrayInputStream(userImg);
    IOUtils.copy(inputStream, response.getOutputStream());
    System.out.print(userImg.length);

    response.getOutputStream().close();
}
```

```
@RequestMapping(value="/classification2")
public String classification2(Model model) throws Exception{

    DLModel dlm =new DLModel();
    double prediction=dlm.handle();
    model.addAttribute("prediction", prediction);

    return "classification2";

}
```

데이터 상세 설명

@데이터 시각화 EXIF를 이용한 GPS 메타 데이터 시각화

사용자 사진에 저장된 **EXIF태그**(gps 위치, 시간)를 파이썬으로 분리 JSON 태로 파싱

▶위도, 경도로 html 시각화 (folium 사용)

▶위도, 경도로 주소 구하기 (reverse geocoding)

```
class PythonUtils:
    def GetJsonFishInfo(path,saveFileName):
        image = Image.open(path+saveFileName+".jpg")

        metaDict={}
        info=image._getexif()
        for tag, value in info.items():
            decoded = TAGS.get(tag, tag)
            metaDict[decoded] = value
        lat_s = ((metaDict["GPSInfo"])[2])[2][0] / ((metaDict["GPSInfo"])[2])[2][1]
        lat_m = (((metaDict["GPSInfo"])[2])[1])[0]
        lat_h = (((metaDict["GPSInfo"])[2])[0])[0]
        s = round((lat_s/60),3)
        s_m = s + lat_m
        m = round((s_m/60),5)
        lat = lat_h + m
        lon_s = ((metaDict["GPSInfo"])[4])[2][0] / ((metaDict["GPSInfo"])[4])[2][1]
        lon_m = (((metaDict["GPSInfo"])[4])[1])[0]
        lon_h = (((metaDict["GPSInfo"])[4])[0])[0]
        s = round((lon_s/60),3)
        s_m = s + lon_m
        m = round((s_m/60),5)
        lon = lon_h + m

        point_map = folium.Map(location = [lat, lon], zoom_start = 30)
        folium.Marker([lat, lon], popup = "map_Catch Point").add_to(point_map)
        point_map.save(path+"map.html")

        soup = BeautifulSoup(open(path+"map.html"), "html.parser")
        filtered_soup = str(soup).replace("\n","")

        g = geocoder.osm([lat, lon], method="reverse")
        addr=g.address
        fish_date = metaDict["DateTime"]
        pic_metadata = OrderedDict()

        pic_metadata["lat"] = lat
        pic_metadata["lon"] = lon
        pic_metadata["addr"] = addr
        pic_metadata["date"] = fish_date
```