

## 포트폴리오



## 신상정보

김현수/Kim Hyun Soo

HP:010-3044-1578

EMAIL:hyunsooklm@naver.com

거주지:서울시 관악구 봉천동

## graduation

2012.03 제물포고등학교 입학

2015.02 제물포고등학교 졸업

2015.03 중앙대학교 입학

2022.02 중앙대학교 졸업(예정)

## AWARDS

2020.08 한국CDE학회상(Edison 전산설계 자율주행 로봇제작 경진대회 특별상)

## 경력

2021.08.02.~2021.08.27 신영증권 IT부서 업무지원팀(인턴)

수행업무: 증권상품 권리 데이터 SQL쿼리업무(데이터 가공업무)

2021.08.30.~2021.09.29. NH농협은행 IT기획부 신기술융합팀(인턴)

수행업무: 농업데이터 기반 AI 김장비용 예측서비스 기획/개발(단독프로젝트)

## 주요 프로젝트

2021.08~2021.09 LSTM네트워크를 활용한 AI김장비용 예측서비스 기획/개발(단독 프로젝트)

2021.04~2021.06 이미지 분석 기반 유기견 입양추천 서비스-유기견 이미지 검색기능 개발

2020.06~2020.08 자율주행 로봇제작 경진대회 - 카메라 영상처리 SW개발(Python-OpenCV)

### ※수행했던 개발분야

데이터 분석 기반 미래가격 예측(LSTM네트워크 활용)

이미지 데이터 분석 프로젝트

라즈베리파이(Linux) 기반 카메라 영상처리SW 개발

## 대학교 주요 수강과목

### 머신러닝

- Numpy를 이용하여 기초 머신러닝 핵심기법 직접 구현
- polynomial regression 직접 구현(Numpy기반)
- Compute loss/gradient 직접 구현(Numpy기반)
- Logistic Regression 직접 구현(Numpy기반)
- Binary classification based on Logistic Regression using nonlinear regression function 직접 구현(Numpy기반)
- K-means clustering 직접 구현(Numpy기반)

### 네트워크 응용 및 설계

- OSI 계층(3~4,7) 계층 집중 학습(Network layer/TransPort layer/Application layer)
- UDP/TCP 소켓프로그래밍 1:1 채팅 프로젝트 구현
- Multi-User 소켓프로그래밍 based on MultiThreadTCPServer / NonblockingTCPServer 프로젝트 구현
- UDP Doubly Linked P2P Chatting 프로젝트 구현

### 캡스톤 프로젝트 1

- JAVA 기반 Android Studio에서 Mobile application
- 사교모임 어플리케이션 개발
- 모임의 지각자를 체크해 Firebase에 기록하는 기능 개발
- TIMER 설정 시간의 모임 장소 GPS와 User의 GPS좌표 비교를 통한 지각자 색출 기능 개발

### 캡스톤 프로젝트 2

- 이미지 분석 기반 유기견 입양 웹 서비스
- 유기견 이미지 검색기능 구현
- R-CNN기반 Object Detection기능을 제공하는 Python ImageAI라이브러리 활용
- Dog\_breed identification(강아지 품종분류) 모델 개발(Google Teacable Machine 툴 활용)

### 알고리즘

- Devide & Conquer / Dynamic Programming / Greedy / BackTracking / Branch & Bound 설계기법 공부 & 예제풀이

### 기타

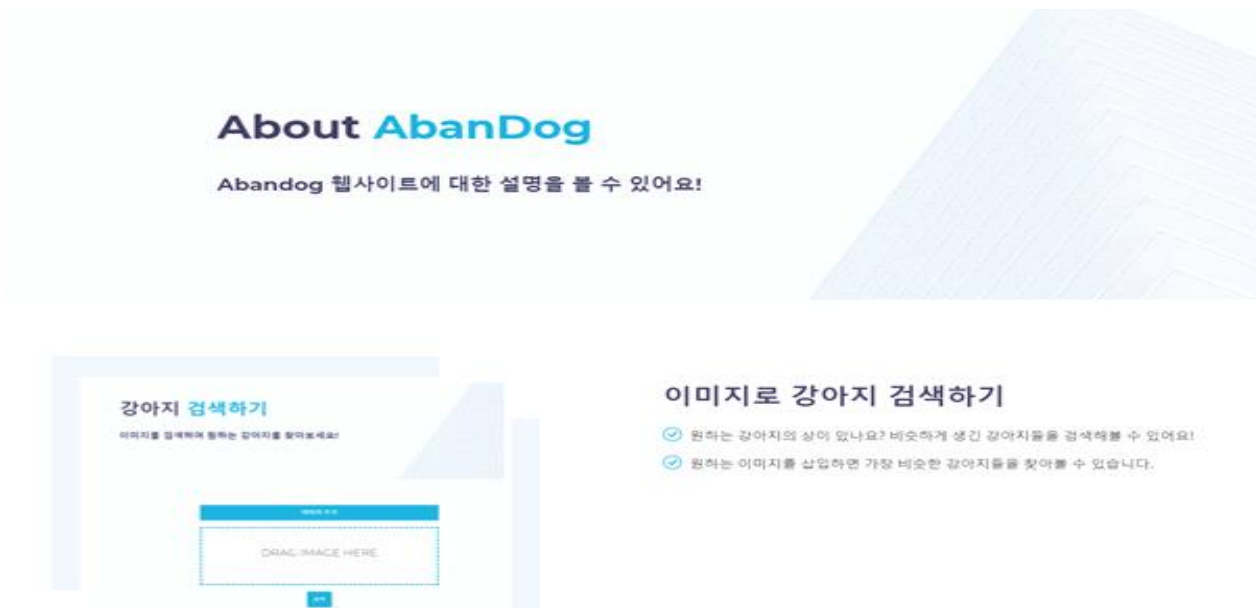
이 외 기본적인 Database / OS / 자료구조 등 기초 CS과목은 모두 이수하였습니다.

## 1. 이미지 분석 기반 유기견 입양 추천 서비스

언어:Python

주요 기술

- Python ImageAI라이브러리
- Object Detection
- Kaggle dog\_breed\_classification
- Google Teachable Machine
- 실패한 이유: 비지도학습 / 클러스tring 기법 / 데이터 수 부족



<유기견 입양권장 프로젝트 웹사이트>

### Abandog 목표

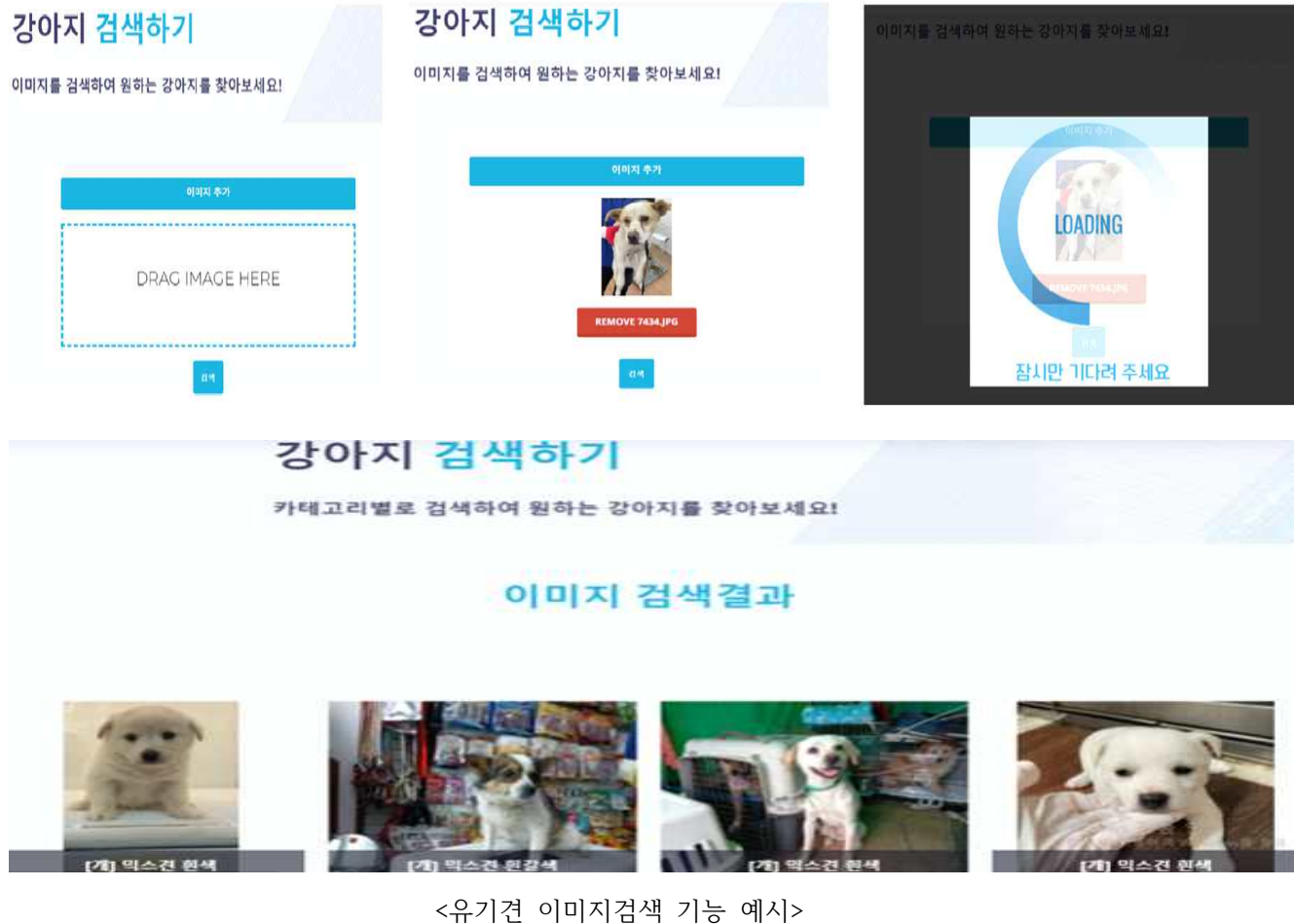
유기견 보호소의 보호기간(통상 10~20일 간 보호하는 기간, 이 때 입양받지 못하면 안락사)에 있는 유기견들을 보여주고 선택이 용이하도록 하여 입양되는 유기견 수의 증가 및 안락사되는 유기견의 감소를 목표로 한다.

사용자가 희망하는 강아지 이미지로, 혹은 자신이 원하는 카테고리별(품종, 색깔, 나이, 성별 등)로 검색할 수 있는 서비스를 통해 유기견의 새로운 가족을 찾아주는 것이 우리 서비스의 목적이다.

※사용자가 희망하는 외형의 유기견을 검색하는 기능을 통해 유기견 입양을 활성화하는 것을 목표로 한다.

## 본인이 개발한 기능

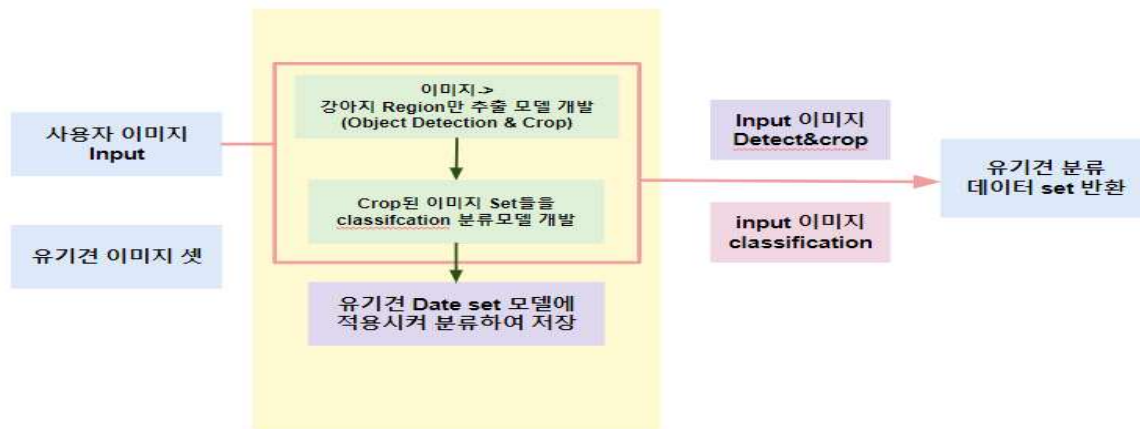
- 마음에 드는 강아지 이미지로 유기견 검색 기능 == 이미지 검색기능 개발



## 이미지검색 기능 개발절차

### 1. 유기견 데이터 수집

공공데이터 포털 OPENAPI를 통해 총 2500여개의 유기견 정보(이미지,출생정보,보호소 위치)를 수집



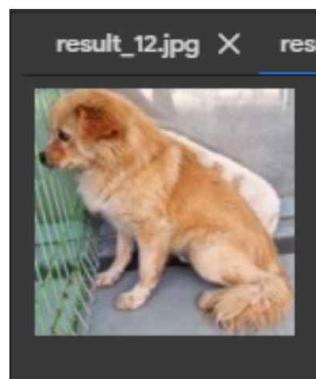
<이미지 검색 모델 아키텍처>

## 2. 이미지에서 강아지영역 Detect and crop

### 머신러닝 - Object Detection & Detection Region Extraction

특징벡터 추출전에 배경을 최대한 배제하기 위해 진행

ImageAI 라이브러리 활용 Resnet 모델 기반 Object Detection 후, 바운딩 박스 내 이미지영역 추출



이름	유형	입출력 크기	입출력 사용	결과
result_2434	이미지	1560	1510	1560
result_2435	이미지	4000	1510	4000
result_2436	이미지	4000	1510	4000
result_2437	이미지	1000	1510	1000
result_2438	이미지	3000	1510	3000
result_2439	이미지	1000	1510	1000
result_2440	이미지	1500	1510	1500
result_2441	이미지	1500	1510	1500
result_2442	이미지	1500	1510	1500
result_2443	이미지	1500	1510	1500
result_2444	이미지	1500	1510	1500
result_2445	이미지	1500	1510	1500
result_2446	이미지	1500	1510	1500
result_2447	이미지	1500	1510	1500
result_2448	이미지	1500	1510	1500
result_2449	이미지	1500	1510	1500
result_2450	이미지	1500	1510	1500
result_2451	이미지	1500	1510	1500
result_2452	이미지	1500	1510	1500
result_2453	이미지	1500	1510	1500
result_2454	이미지	1500	1510	1500
result_2455	이미지	1500	1510	1500
result_2456	이미지	1500	1510	1500
result_2457	이미지	1500	1510	1500
result_2458	이미지	1500	1510	1500
result_2459	이미지	1500	1510	1500
result_2460	이미지	1500	1510	1500

Windows 점진 인증  
총 2500여개의 Detection & Extraction Result 확보

이미지 상 강아지의 유무 판별과, 강아지 이미지 분석 시 정확도 향상을 위해. 사전 데이터 강아지 영역 Detection과 bounding box영역을 crop하는기능을 구현했다.

해당 기능은 R-CNN기반 Object detection기능을 제공하는 Python imageAI라이브러리를 활용하였고, 해당 라이브러리의 detectCustomObjectsFromImage함수를 사용했다. 이 기능은 유기견 데이터 전처리와, 사용자 입력이미지에 대한 강아지 유무를 확인하여 예외처리에 사용되었다.

```
def DetectCrop(input_image,detect_path,detector,custom_objects):
    #print('here1')
    detections = detector.detectCustomObjectsFromImage(custom_objects=custom_objects,
                                                         input_image=input_image,
                                                         output_image_path=detect_path,
                                                         minimum_percentage_probability=30)
    #애가 crop and 바운딩박스그려놓은걸 detected crop 저장 !!쓸데없음

    #print('here2')
    # 강아지 영역 detect
    box_point = tuple(detections[0]['box_points'])
    #print('here3')
    src = cv2.imread(input_image)
    crop_image = src[box_point[1]:box_point[3], box_point[0]:box_point[2]]
    #cv2.imwrite(crop_path, crop_image)
    pil_Image = Image.fromarray(crop_image)
    return pil_Image
```

<Detect & Crop 핵심코드>

### 3. 견종 classification

#### classification 모델 기반 유기견 데이터셋 나누기

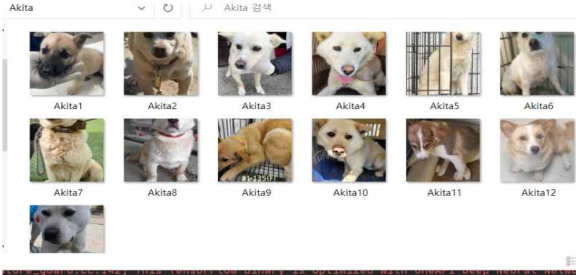
Detect and Crop 모델을 통해 유기견 데이터 셋의 강아지 영역만 추출

Classification 분류모델을 통해 유기견 데이터셋을 분류하여 폴더에 저장

기존 Dog Detection+Crop모델 + classification 분류모델 활용 정확도 향상

폴더명	생성일	파일명
cropped_data	2021-05-22 오후 10:04	파일폴더
detect_result	2021-05-22 오후 10:04	파일폴더
original_data	2021-05-22 오후 6:19	파일폴더

**DATA SET 분류**



Windows 정품 인증  
[설정]으로 이동하여 Windows를 정품 인증함

**모델을 통해 나눈 유기견데이터**

키우고 싶은 강아지 이미지로 유기견을 검색하는 원리는 간단하다.  
'같은 종의 강아지는 비슷하게 생겼다.'

dog-breed classification 모델을 구현해 우리가 가진 데이터에 적용했다. kaggle dog-breed dataset을 구한 후, 구글 티처블머신을 활용하여, 데이터 셋 라벨링을 해준 뒤, 훈련시켜 강아지 종 분류 학습모델을 얻었다.

Detect and Crop모델에 처리된 유기견 데이터셋을 강아지 종분류 학습모델을 통과시켜 나온

결과값(유기견의 품종 데이터)을 유기견 정보가 저장된 데이터베이스에 저장한다.

이 후 사용자 입력 이미지를 두 모델에 통과한 결과값을 데이터베이스에 query하여 나온 결과값을 추천결과로 돌려준다.

※dog-breed classification모델

강아지 이미지로부터 강아지 종을 추출하는 모델

```

model = tensorflow.keras.models.load_model('/home/wndvlf96/abandog/keras_model.h5')
np.set_printoptions(suppress=True)
new_updt_list = []

for i in range(len(rows)):
    url = rows[i]['img']
    cid = rows[i]['cid']
    try:
        image = Image.open(requests.get(url, stream=True).raw)
        image.save('/home/wndvlf96/abandog/Test/test.jpg')
        input_image = '/home/wndvlf96/abandog/Test/test.jpg' #test1.jpg로 저장.
        detect_path = '/home/wndvlf96/abandog/Testdetect/detect.jpg' #./Testdetect폴더 하위에 detect.jpg로 저장됨.
        try:
            print(i)
            cropped_img = DetectCrop(input_image, detect_path, detector, custom_objects)
            # classification 진행
            data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
            size = (224, 224)
            cropped_img = ImageOps.fit(cropped_img, size, Image.ANTIALIAS)
            image_array = np.asarray(cropped_img)
            normalized_image_array = (image_array.astype(np.float32) / 127.0) - 1
            data[0] = normalized_image_array
            prediction = model.predict(data)
            pred_label = np.argmax(prediction)
            # pred_label 값을 specials컬럼에 저장하기 위한 준비
            # new_updt_list.append([pred_label, cid])
            sql = "UPDATE `dog` SET `specials`=%s WHERE `cid` = %s;" %(pred_label, cid)
            cur.execute(sql)
            conn.commit()
        except:
            pass

```

구글 Teachable머신을 통해 얻은  
dog\_breed classification 모델

<dog\_breed classification 핵심코드>

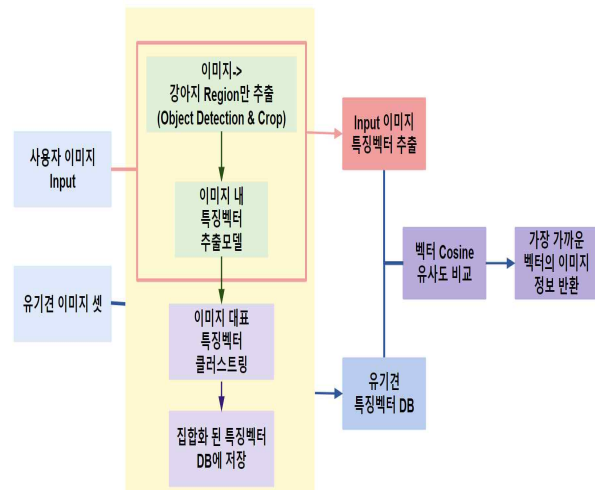
## 개발도중 어려웠던 점

### - 이미지 검색모델의 설계 오류

처음엔 구글/네이버 등에서 사용되는 이미지검색 기능을 적용시키려고 노력했다.



<참고했던 이미지 검색 시스템 예시>



<초기 이미지검색 모델 아키텍처>

강아지 영역을 추출하는 Object Detection & Crop 모델의 사용은 동일하다. 차이점은 강아지 종 Classification이 아닌, 이미지 특징벡터 클러스터링을 통해 실제 이미지검색을 구현하려고 시도한 점이다.

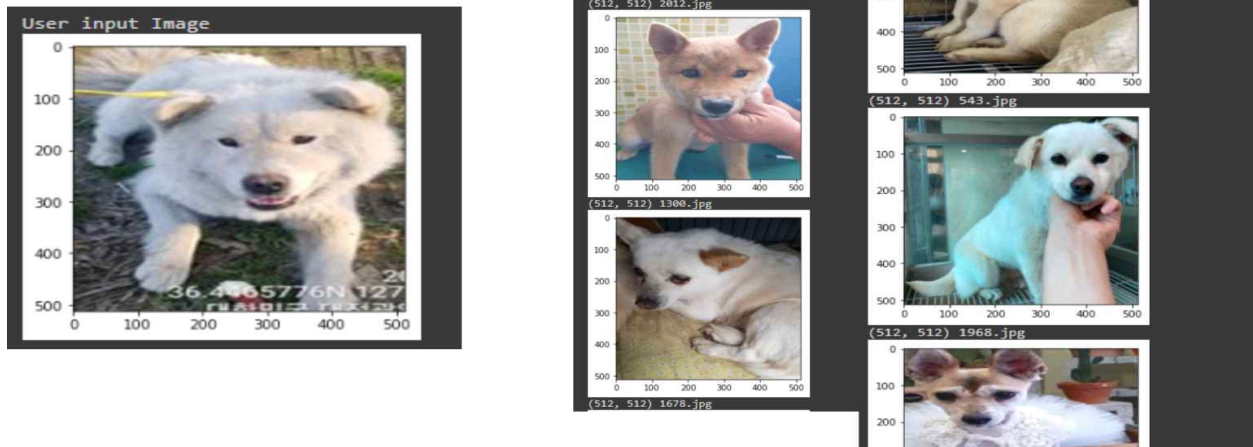
비지도학습 기반 k-means 군집화 알고리즘을 통해 이미지 특징벡터 클러스터링을 시도했다.



## - 알고리즘 설계방식

1. Object Detection & Crop모델을 통해 강아지영역만 추출한 이미지 데이터 셋 준비
2. ResNet모델을 활용해 각 이미지 데이터의 고유한 특징을 벡터화시킨 특징벡터 추출
3. k-means 군집화 알고리즘을 통해 특징벡터 클러스터링
4. 사용자 입력 이미지로부터 특징벡터를 추출해 최단거리 내 군집화된 이미지 데이터셋 반환

### 1. 이미지 검색 기능



<초기 이미지검색 기능의 낮은 성능 예시>

### ※해당 기능 개발 실패한 이유

- 비지도 학습 기반 클러스터링 기법을 사용하기엔 보유한 데이터셋의 크기가 작았다.

비지도 학습을 통해 이미지 데이터의 특징벡터끼리 군집을 이루기 위해선 대규모의 데이터셋이 필요하다. 그러나 우리가 확보한 유기견 dataset의 크기는 약 2500개로, 데이터셋이 비지도학습을 진행하기에 적합하지 않았다.

- 이미지 데이터셋 부풀리기를 통해 dataset 규모 확장을 시도했지만, 유기견들이 비슷하게 생긴 경우가 많아, 실제 성능향상으로 이어지지 않았다.

- 기능구현을 위해 실제적인 이미지 검색보단 유기견데이터의 종을 최대한 세분화해, 사용자로부터 입력된 강아지와 같은 종의 유기견을 추천해주는 방식으로 방법을 선회했다.

입력된 강아지 이미지와 비슷한 외형의 강아지를 추천해주는 기능이 목표.  
한참을 이미지 검색을 시도하다.. 결국 기한 내 개발완수를 위해 개발방법 선회...



## 2. 자율주행 로봇제작 경진대회-주행용 카메라 영상처리 sw개발

언어:Python

주요 기술: OpenCV 라이브러리

개발환경: 라즈베리파이 기반 Linux 개발환경



<자율주행 로봇 제작 경진대회 출품작>

### 대회 목표

자율주행 로봇을 제작하여 위 사진과 같은 맵을 가장 빠르게 완주하는 것이 목표이다.

프로젝트 구성: 로봇 하드웨어 제작 | 자율주행을 위한 카메라 영상처리 SW개발

로봇 하드웨어를 제작하여 소형 카메라 1개와 라즈베리파이를 장착한 후, 라즈베리파이 내 코딩을 통해 주행기능 개발

언어:Python 활용 라이브러리:OpenCV 라이브러리

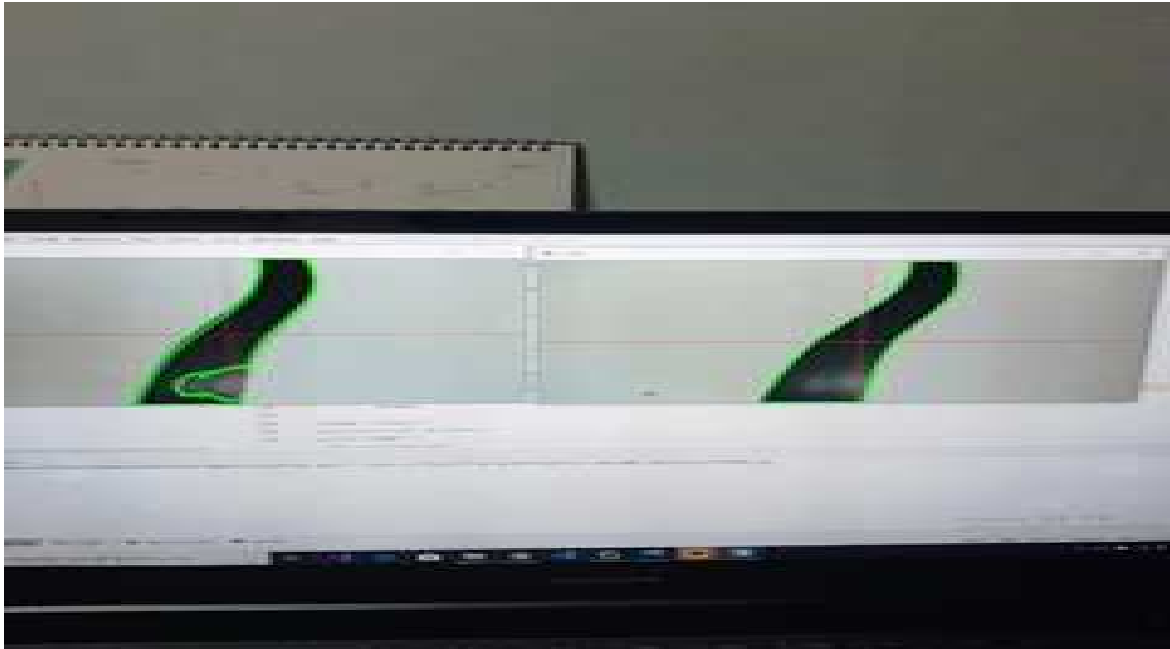
개발환경:라즈베리 파이 내 Linux 환경

본인이 기능한 개발

1. 카메라 영상을 통해 주행도로 인식 및 직진,좌/우회전 판단 알고리즘
2. 카메라 영상을 통해 위 사진의 좌상단 정지 표지판을 인식하면 10초간 정지 후 출발하는 알고리즘 설계/개발

## 자율주행용 영상처리 SW 개발절차

### 1. 주행도로 인식 및 직진,좌/우회전 판단 알고리즘



<주행도로 인식 개발화면>

#### 도로인식 후 방향전환 알고리즘 설계방식

1. 영상 속 이미지 흑백전환 후 OpenCV의 findContours함수를 통해 도로 인식(위 이미지의 초록색 선)
2. 인식된 도로 내 무게중심의 좌표 찾기(위 이미지의 빨간 십자가 선이 바로 도로의 무게중심)
3. 무게중심 좌표를 이미지 정중앙 y좌표와 비교하여 좌/우/직진 판단 후 주행  
(하단 우측 코드의 if  $cx \leq 365$ 를 통해 left/right/straight를 결정)

```
def direct_detection(frame,STOP_COUNT):
```

```
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray,(5,5),0)
    #canny=cv2.Canny(blur,100,200)

    _,thresh1 = cv2.threshold(blur,200,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    thresh2=cv2.bitwise_not(thresh1)    ##흰색 검출
    #_,thresh2 = cv2.threshold(blur,200,255,cv2.THRESH_BINARY_INV)    ##검은색 검출

    # cv2.imshow('thresh2',thresh2)

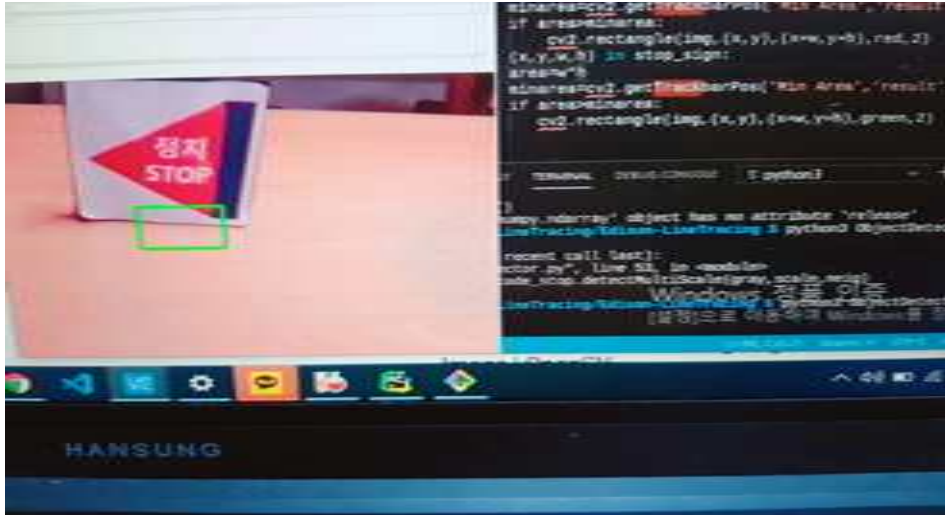
    blk,_ = cv2.findContours(thresh2,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
```

<도로인식 핵심코드>

```
    contours1,hierarchy = cv2.findContours(white.copy(), 1, cv2.CHAIN_APPROX_NONE)
    contours2,_=cv2.findContours(black.copy(), 1, cv2.CHAIN_APPROX_NONE)
    if len(contours1)>0 and len(contours2)>0: #흰 검 둘다잡힐때
        c=max(contours1, key=cv2.contourArea)
        M=cv2.moments(c)
        try:
            cx=int(M['m10']/M['m00'])
            cy=int(M['m01']/M['m00'])
            #무게중심코드
        except ZeroDivisionError as e:
            # print("what?")
            straight(100)
            continue
        if cx<=365:
            right(100)
        else:
            left(100)
    elif len(contours1)<=0 and len(contours2)>0: #검은색만 잡힐때
        straight(100)
```

<방향전환 판단 핵심코드>

## 2. 정지신호판 인식 후 10초간 정지 기능



<정지 신호판 인식 개발화면>

### 정지 신호판 인식 알고리즘 설계방식

1. 카메라 영상 속 붉은 색의 도형 찾기
2. 붉은 색의 도형이 삼각형일 경우 정지신호판으로 판단
3. stop\_detection 함수 내 count 매개변수의 의미는, 미션 수행 중 정지판은 하나뿐이고, 그에따라 미션 수행 과정에서 신호판을 다시 인식하지 않도록 0/1을 통해 조정(일종의 flag 효과)

```
lower_red = np.array([145,100,100])
upper_red = np.array([255,255,255])

def stop_detection(frame,count):
    if count>0:
        return False
    else:
        hsv = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
        red_range = cv2.inRange(hsv, lower_red, upper_red)
        red_range = cv2.erode(red_range,None, iterations=1)
        red_range = cv2.dilate(red_range,None,iterations=1)
        cnts,_ = cv2.findContours(red_range.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

        try:
            if len(cnts)>0: #컨투어잡고
                cnt2=max(cnts,key=cv2.contourArea)
                # cv2.drawContours(frame,cnt2,-1,(0,255,0),3)
                # ### contour 몇 개인지
                epsilon = 0.03*cv2.arcLength(cnt2,True)
                approx = cv2.approxPolyDP(cnt2,epsilon,True)
                size = len(approx)
                if size<6: #삼각형잡았을경우
                    # print(size)
                    # for q in range(size-1):
                    #     cv2.line(frame,tuple(approx[q][0]),tuple(approx[q+1][0]),(255,0,0),3)
                    return True
                else:
                    return False
            else:
                return False
        except:
            return False
```

<정지 신호판 인식 핵심 코드>

### 3. 농업데이터 기반 AI 김장비용 예측서비스 기획/개발



농산물가격예측

예측하기

향 후 4주간의 김장비용 예측 결과

기준일자:2019-10-15

DOWNLOAD ALL



<농업데이터 기반 AI 김장비용 예측 서비스 웹사이트>

#### AI김장비용 예측서비스 목표

농산물 가격변동이 심한 김장철, 향 후 4주간의 농산물 소매가격을 예측해 김장비용을 산출한 것을 시각화함으로써 소비자의 김장비용 물가부담을 줄여주는 것을 목표로 했다.

언어:Python

주요 기술

- Pandas 라이브러리를 활용한 데이터 수집/정제
- 미래 데이터 예측을 위해 RNN계열의 LSTM네트워크 활용
- Keras 기반 LSTM 네트워크 구성
- Flask 프레임워크 기반 AI 모델 api서비스화

## 아이디어 기획

김장철이 되면 수요가 높아져 채소수급이 불안정하여, 농산물 가격변동이 심하다.  
변동이 심한 농산물의 가격을 정확히 예측해 예상 김장비용을 시각화함으로써 소비자의 김장물가부담을 줄여준다.



<뉴스기사>

<김장비용 세부내역>

구분 김장재료		가격 (단위:원)			동락률 (단위:%)	
		금일(A) 12/2(목)	지난주(B) 11/25(목)	전년도(C) (20.12.2)	지난주 대비 (A/B)	전년 대비 (A/C)
배추	20포기	90,840	91,920	59,140	-1.2	53.6
무	10개	17,790	17,950	19,240	-0.9	-7.5
고춧가루	1.86kg	56,445	57,082	65,303	-1.1	-13.6
마늘	1.2kg	14,837	13,753	12,048	7.9	23.1
대파	2kg	6,168	6,226	8,172	-0.9	-24.5
쪽파	2.4kg	14,040	16,320	17,074	-14.0	-17.8
상장	0.12kg	838	793	1,069	5.7	-21.6
미나리	2kg	19,520	19,040	20,360	2.5	-4.1
갯	2.6kg	10,039	10,301	15,179	-2.5	-33.9
굴	2kg	56,986	58,192	49,096	-2.1	16.1
멸치액젓	1.2kg	5,612	5,622	5,467	-0.2	2.7
새우젓	1kg	21,855	21,533	20,461	1.5	6.8
소금	8kg	16,386	16,147	12,757	1.5	28.4
합계금액		331,356	334,879	305,365	-1.1	8.5

인공지능 예측대상 농산물

비예측대상 농산물  
(실제가격)

<한국농수산물유통공사 출처 4인가구 김장재료>

## 제한사항

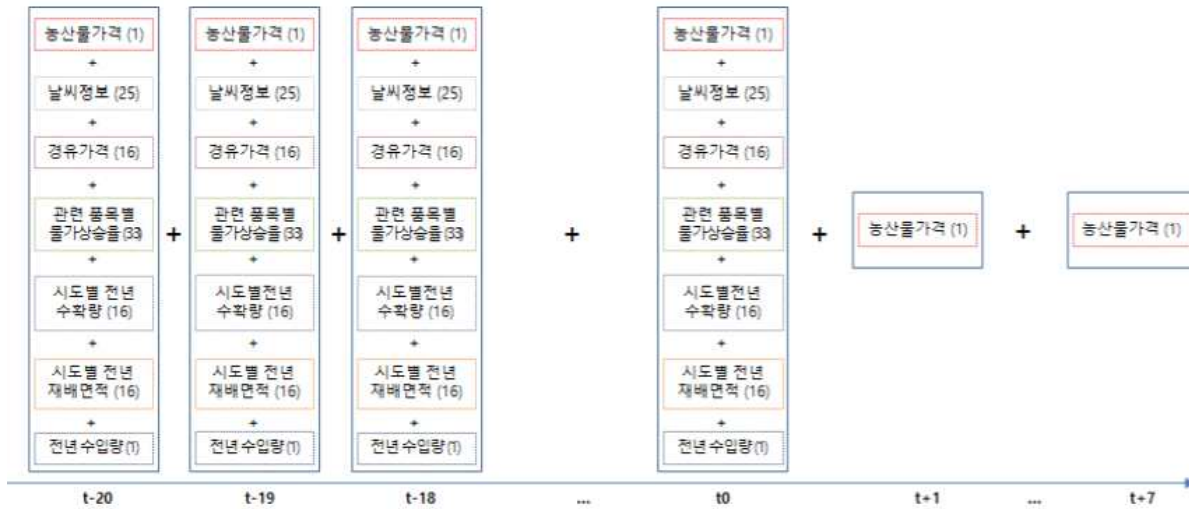
우측 한국농수산물유통공사 4인가구 김장재료 표준재료를 기반으로 김장비용을 예측했다.  
김장비용은 상위 6가지 항목의 인공지능 기반 예측가격과 항목 하위 7가지 항목의 동일날짜 실제  
소매가격을 합산하여 산출했다.  
한달이란 기한 상, 김장재료 항목 13가지 항목 모두에 대한 가격예측모델을 개발하는 것이 불가능했다.  
이에 따라 김장재료중 주요 농산물만 가격예측 대상으로 삼았다.

참고 논문은 ‘LSTM 네트워크를 활용한 농산물 가격 예측 모델’이다.





## 가격예측 진행방식

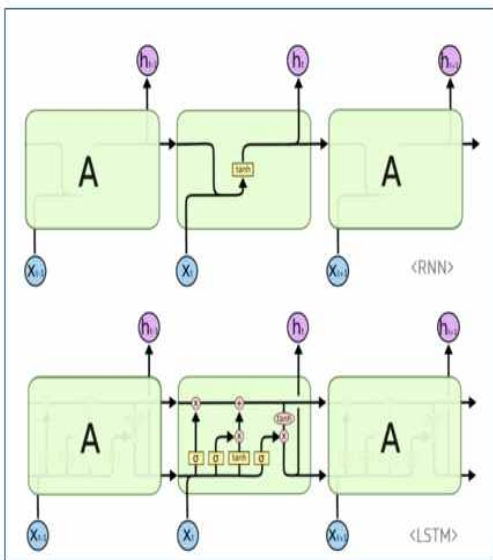


<참고한 논문의 학습데이터>

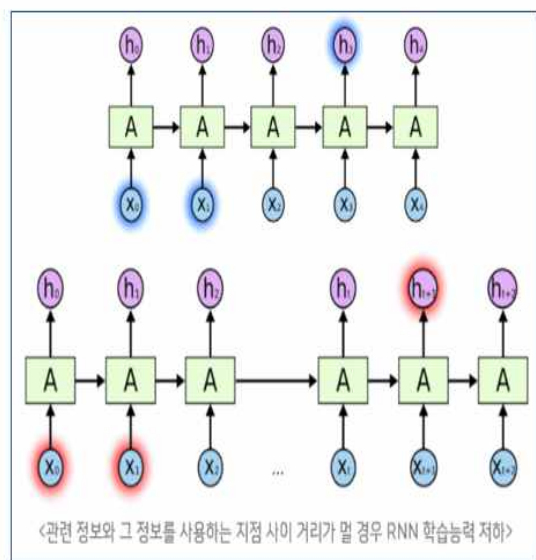
개발시 참고한 논문은 LSTM네트워크를 활용해 과거 21일간의 농업데이터를 기반으로 미래 7일간의 농산물 가격을 예측하는 방식으로 진행된다. 본 프로젝트에선 해당 논문을 참고해 과거 50일간의 데이터를 기반으로 미래 28일간의 농산물가격을 예측하는 네트워크를 구성하였다.

## Q. RNN과 LSTM이란?

RNN은 히든 노드가 방향을 가진 엣지로 연결돼 순환구조를 이루는(directed cycle) 인공신경망의 한 종류이다. RNN은 관련 정보와 그 정보를 사용하는 지점 사이 거리가 멀 경우 역전파시 그래디언트가 점차 줄어 학습능력이 크게 저하된다. 이를 vanishing gradient problem이라고 한다. ->(장기간의 과거데이터를 기반으로 미래데이터를 예측하는데 부적합한 이유)



<RNN과 LSTM네트워크 구조>



<RNN 학습능력 저하>

LSTM네트워크는 RNN의 한계를 극복한 모델로 내부 RNN의 Hidden State에 cell-state를 추가함으로써 그래디언트 전파가 잘되게한다. 이를 통해 기존 RNN의 한계를 극복해 장기간의



과거데이터를 기반으로 미래데이터를 예측하는데 용이하게 된다.

## 개발코드

과거 50일간의 데이터를 기반으로 미래 28일간의 가격을 예측하므로, LSTM Many-to-many 네트워크를 구성하였으며 해당 네트워크의 코드는 github 검색을 통해 완성했다.

```
# Import Libraries and packages from Keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from tensorflow.keras.optimizers import Adam # - Works
from keras.callbacks import ModelCheckpoint
from keras.callbacks import EarlyStopping

model = Sequential()
model.add(LSTM(units=64,activation='relu', input_shape=(X_train.shape[1],X_train.shape[2]),return_sequences=True ))
model.add(LSTM(units=32, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(Y_train.shape[1]))
model.compile(optimizer='adam', loss='mse')
model.summary()

MODEL_SAVE_FOLDER_PATH = '/content/gdrive/MyDrive/Nonghyup/model/baechu.h5'
early_stop = EarlyStopping(monitor='val_loss', min_delta=0.0001, patience=15, verbose=1, mode='auto')

checkpoint_filepath = MODEL_SAVE_FOLDER_PATH
model_checkpoint_callback = ModelCheckpoint(
    filepath=checkpoint_filepath,
    monitor='val_loss', verbose=1, save_best_only=True,mode='auto')

history=model.fit(X_train,Y_train,epochs=30,batch_size=16,validation_split=0.2,callbacks=[model_checkpoint_callback,early_stop])
```

<lstm네트워크 개발 코드>

학습데이터 수집 과정에서 공휴일의 경우 조사기관의 데이터 중 결측값이 존재하는 경우가 있었다. 학습데이터들 중 농산물 도매가격, 기상 데이터의 경우 과거 며칠간의 데이터로부터 크게 벗어나지 않는 경우가 많아 선형보간법을 통해 보간하는 방식을 선택했다. Pandas라이브러리의 interpolate함수를 통해 결측데이터를 보간했다.

## Pandas 라이브러리 활용 데이터 결측치 보간(공휴일 등 데이터 값이 존재하지 않는 경우 데이터 흐름(선형)에 따라 채워줌(LINEAR))

```
import pandas as pd
from pandas import DataFrame, Series

address='/content/gdrive/MyDrive/Nonghyup/data_preprocessing/sub_price/'
name_list=['gat.xlsx','seonggang.xlsx','myulchi.xlsx','seu.xlsx','sogim.xlsx']

for name in name_list:
    address_target=address+name
    df=pd.read_excel(address_target)

    df=df.astype({'거래일자':'int'}) #gochu 날짜 데이터타입 int로 바꿈

    df['거래일자']=pd.to_datetime(df['거래일자'],format='%Y%m%d')
    df=df.loc[df['거래일자']<='2019-11-12']
    df=df.loc[df['거래일자']>='2006-03-14']

    df['price']=df['price'].astype('float')

    df['price']=df['price'].interpolate(method='linear',limit_direction='forward', axis=0)
    df.to_excel(address_target, index=False)
```

<데이터 전처리 과정에서 통한 결측값 보간>

## 김장비용 산출

한국농수산식품유통공사의 표준 재료기준의 맞춰 농산물 예측가격을 환산한 뒤, 비예측 김장재료의 실제가격을 합산하여 4인가구 김장비용을 산출한다.

```
def Predict_Gimjang():
    # -----original-----
    original_baechu['gimjang'] = original_baechu['some'] * 20 # 배추 x 20포기
    original_muu['gimjang'] = original_muu['some'] * 10 # 무 x 10개
    original_gochu['gimjang'] = original_gochu['some'] * 3.1 # 고춧가루 x 3.1
    original_manl['gimjang'] = original_manl['some'] * 1.2 # 마늘 x 1.2
    original_defa['gimjang'] = original_defa['some'] * 2 # 대파 x 2
    original_jjokpa['gimjang'] = original_jjokpa['some'] * 2.4 # 쪽파 x 2.4

    # -----Predict-----
    PREDICT_baechu['gimjang'] = PREDICT_baechu['price'] * 20 # 배추 x 20포기
    PREDICT_muu['gimjang'] = PREDICT_muu['price'] * 10 # 무 x 10개
    PREDICT_gochu['gimjang'] = PREDICT_gochu['price'] * 3.1 # 고춧가루 x 3.1
    PREDICT_manl['gimjang'] = PREDICT_manl['price'] * 1.2 # 마늘 x 1.2
    PREDICT_defa['gimjang'] = PREDICT_defa['price'] * 2 # 대파 x 2
    PREDICT_jjokpa['gimjang'] = PREDICT_jjokpa['price'] * 2.4 # 쪽파 x 2.4

    predict_price = []
    original_price = []
    for i in range(len(predict_date)):
        original_sum_price = 0
        predict_sum_price = 0

        original_sum_price += original_baechu['gimjang'].tolist()[i]
        original_sum_price += original_muu['gimjang'].tolist()[i]
        original_sum_price += original_gochu['gimjang'].tolist()[i]
        original_sum_price += original_manl['gimjang'].tolist()[i]
        original_sum_price += original_defa['gimjang'].tolist()[i]
        original_sum_price += original_jjokpa['gimjang'].tolist()[i]

        predict_sum_price += PREDICT_baechu['gimjang'][i]
        predict_sum_price += PREDICT_muu['gimjang'][i]
        predict_sum_price += PREDICT_gochu['gimjang'][i]
        predict_sum_price += PREDICT_manl['gimjang'][i]
        predict_sum_price += PREDICT_defa['gimjang'][i]
        predict_sum_price += PREDICT_jjokpa['gimjang'][i]

    # -----original/predict price에 각각 실제 농산물가격 / 예측 농산물가격 넣기
    # -----original/predict price에 각각 비예측농산물 실제가격 넣기
    predict_sum_price += gat_df['gimjang'].tolist()[i]
    predict_sum_price += gool_df['gimjang'].tolist()[i]
    predict_sum_price += minari_df['gimjang'].tolist()[i]
    predict_sum_price += myulchi_df['gimjang'].tolist()[i]
    predict_sum_price += senggang_df['gimjang'].tolist()[i]
    predict_sum_price += seu_df['gimjang'].tolist()[i]
    predict_sum_price += sogm_df['gimjang'].tolist()[i]
    original_price.append(original_sum_price)
    predict_price.append(predict_sum_price)
```

<4인가구 김장비용 산출 개발코드>

## Flask 기반 모델 api화 웹서비스

Python Flask라이브러리와 goormide를 통해 개발한 AI모델을 API화시켰다.

```
# 메인 페이지 라우팅
@app.route("/")
@app.route("/index")
def home():
    return flask.render_template('index.html')

@app.route('/result', methods=['POST', 'GET'])
def result():
    if request.method == 'POST':
        last_Day = '2019-10-15'
        print(f'last_Day:{last_Day}')

        make_df()
        make_Predict_parameter(last_Day)
        predict()
        make_predict_df()
        make_nonPredict_price()
        PREDICT_GIMJANG = Predict_Gimjang()
        Gimjang_Total()

        # PREDICT_GIMJANG
        labels = PREDICT_GIMJANG['Date'].astype('str')
        Origin_price = PREDICT_GIMJANG['Origin_Sum'].astype('int')
        Predict_price = PREDICT_GIMJANG['Predict_Sum'].astype('int')
        labels = labels.tolist()
        Origin_price = Origin_price.tolist()
        Predict_price = Predict_price.tolist()

    return flask.render_template('index.html', labels=labels, Origin_price=Origin_price,
                                Predict_price=Predict_price, status_day=last_Day)
```

<Flask기반 모델 api화>