

농산물 이미지를 활용한

# 농산물 상품 등급 분류하기

2팀 (권보라, 박나은, 이나겸, 정성우)

# CONTENTS

**01** 프로젝트 개요

**02** 프로젝트 과정

**03** 프로젝트 결과

**04** 프로젝트 소감

CONTENTS

# 01 프로젝트 개요

---

권보라



- 학습 샘플 코드 작성
- Flask 웹 어플리케이션 구현
- 훈련된 모델을 활용하여 웹 페이지로 모델 서빙하는 코드 작성 및 기능 구현
- 포트폴리오 PPT 작성

박나은



- 학습 샘플 코드 작성
- 원본 데이터 학습 진행

이나겸



- 학습 샘플 코드 작성
- 학습 결과 시각화 작업
- 데이터 샘플링 작업
- 원본 데이터 학습 진행
- 포트폴리오 PPT 작성

정성우



- 학습 샘플 코드 작성
- 메타데이터 학습을 위한 linear regression 작업
- 원본 데이터 학습 진행
- 포트폴리오 PPT 작성

# 01 프로젝트 개요

농산물 품질을 나눌 때 크기와 무게에 따라 기계가 1차적으로 분류하고 나면  
사람이 농산물의 겉모양이나 색깔을 보고 2차적으로 분류한다.  
사람이 눈으로 보고 분류하는 2차 분류를 AI가 처리한다면 사람이 하는 일이 많이 줄어들 것으로 판단되어 주제로 선택.

목표 : 농산물 품질을 판단할 수 있는 AI 서비스를 구축하는 것

기능 구현



- 농산물 QC 데이터를 딥러닝으로 학습시키고, 웹에서 농산물 이미지 데이터를 입력했을 때 농산물 품질 예측 결과를 확인할 수 있다

활용 장비, 개발 환경



- Microsoft Azure, goorm IDE
- AI hub 농산물 QC 데이터 (<https://aihub.or.kr>)
- Visual Studio Code
- Pytorch

프로젝트 기대효과



농산물 품질을 분류할 때,  
Vision을 사용한다면  
1차 분류 뿐만 아니라 2차 분류까지  
자동화가 가능할 것으로 기대



## 02 프로젝트 과정

---

### 01

#### 데이터 샘플링

.....

- 원본 데이터 샘플링
- 파일 이름 매칭 확인
- 이미지 resize 작업
- 파일 공유를 위한 업로드 및 다운로드

### 02

#### 모델 확인, 선정

.....

- 샘플 데이터를 통해 모델 점수 확인
- 점수가 높은 모델을 선정하고 학습 진행

### 03

#### 메타데이터 학습

.....

- 선형회귀 학습진행
- 라벨이 입력되면 너비, 높이, 무게 예측

### 04

#### Flask 웹 구축

.....

- 학습된 모델을 통해 업로드 되는 사진의 종류, 등급, 특징을 보여주는 웹 구축

## 02 프로젝트 과정

학습 데이터 준비 및 샘플링

### 이미지 데이터 resize

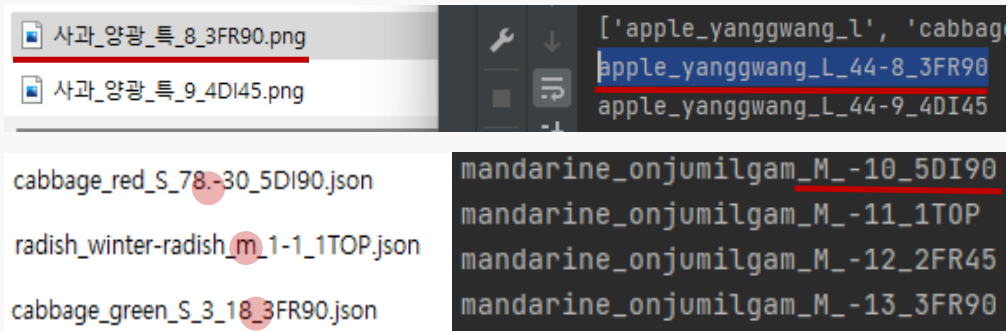
데이터 세트의 파일 선택 \*

이 파일은 선택한 데이터 저장소에 업로드되어 작업 영역에서 사용할 수 있습니다.

찾아보기 ▾ 95676개 파일을 선택했습니다. 전체 크기는 3.01 GB입니다. 28344/95676개 파일을 업로드했습니다.

- 전체 용량이 커서 이미지를 224, 224로 resize 후 업로드

### 파일 이름 매칭되게 수정



- png 파일과 json 파일명이 매칭되지 않는 부분 확인 (파일명이 한글, '.' 포함, '등급 소문자', '-'를 '\_'로 사용 등)
- png와 json 파일을 리스트로 불러와서 매칭되지 않는 파일이 있으면 이름을 추출하여 매칭되도록 파일 이름 변경

### 원본 데이터에서 20% 샘플링

```
idx = random.sample(range(data_length), int(data_length * 0.2))
sample = []
for i in idx:
    sample.append(rfile_list[i])
    # [168, 81, 32, 29, 262, 308, 76, 271, 299, 106, 265, 72, 19

remove_file = [x for x in rfile_list if x not in sample]

for file in remove_file:
    file = file.split('.')[0]
    a = os.path.join(json_path, file+'.json')
    b = os.path.join(png_path, file+'.png')
    os.remove(a)
    os.remove(b)
```

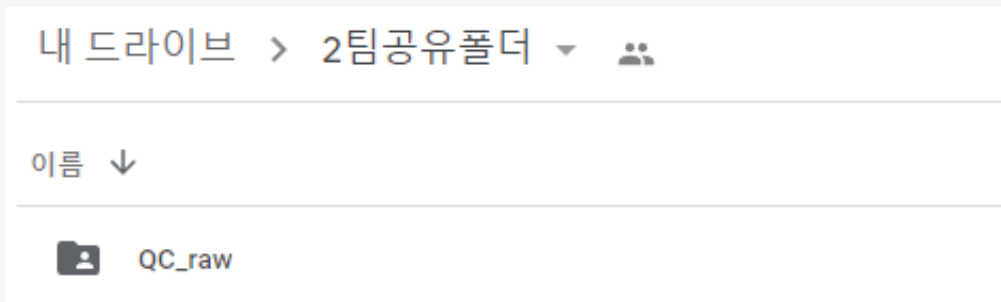
apple_fuji_L_1-1_1top.png	apple_fuji_L_1-2_2FR45.png	apple_fuji_L_1-2_2FR45.json
apple_fuji_L_1-2_2fr45.png	apple_fuji_L_1-7_2FR45.png	apple_fuji_L_1-7_2FR45.json
apple_fuji_L_1-3_3fr90.png	apple_fuji_L_1-11_1TOP.png	apple_fuji_L_1-11_1TOP.json
apple_fuji_L_1-4_4di45.png	apple_fuji_L_1-12_2FR45.png	apple_fuji_L_1-12_2FR45.json
apple_fuji_L_1-5_5di90.png	apple_fuji_L_1-16_1TOP.png	apple_fuji_L_1-16_1TOP.json


- 원본 데이터의 개수가 많아서 Azure에 모든 데이터를 올리기가 힘들다고 판단되어 전체 파일에서 20%를 제외한 나머지 80%의 데이터는 원본에서 os.remove로 삭제

## 02 프로젝트 과정

학습 데이터 Azure 업로드

### 학습 데이터 업로드



	QC_raw
종류:	파일 폴더
위치:	E:\Portfolio\Classification
크기:	3.00GB (3,231,794,892 바이트)
디스크 할당 크기:	3.23GB (3,478,822,912 바이트)
내용:	파일 95,676, 폴더 222

- 이미지 크기를 줄여서 용량을 작게 만들었으나 파일 개수가 약 9만개로 많음
- Azure의 데이터 탭에 7시간 동안 약 4만개 파일이 업로드 되다가 사라짐.
- 구글 드라이브에서 다운로드 받는 방법으로 변경

### python코드로 다운로드 후 압축해제 진행

```
parser = argparse.ArgumentParser(description='data loader ... ')
parser.add_argument('--data', type=str, help='key for selecting data!!!')
args = parser.parse_args()

download_file_from_google_drive(
    id=file_id_dic[args.data], destination=file_destinations[args.data]
)

test_file_name = "./QC_raw.zip"

with ZipFile(test_file_name, 'r') as zip:
    zip.printdir()
    zip.extractall()
```



- .zip파일 다운로드와 압축 해제가 잘된 것을 확인



# 02 프로젝트 과정

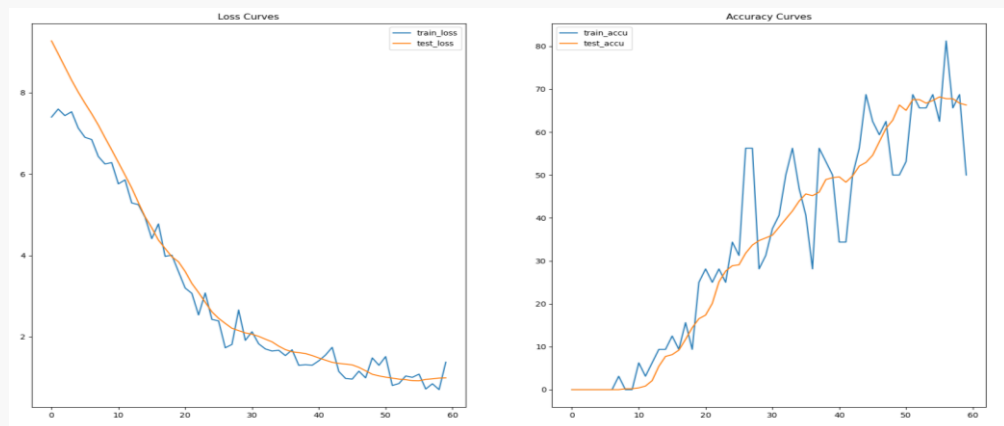
데이터 학습 기록, 모델 선정

샘플 데이터 학습으로 비교해서 가장 좋은 모델 선정

GhostNet

Epoch / Valid	3 / 3	Loss Function	CrossEntropy
Batch size	32	Optimizer	SGD
Momentum	0.9	Learning rate	0.025

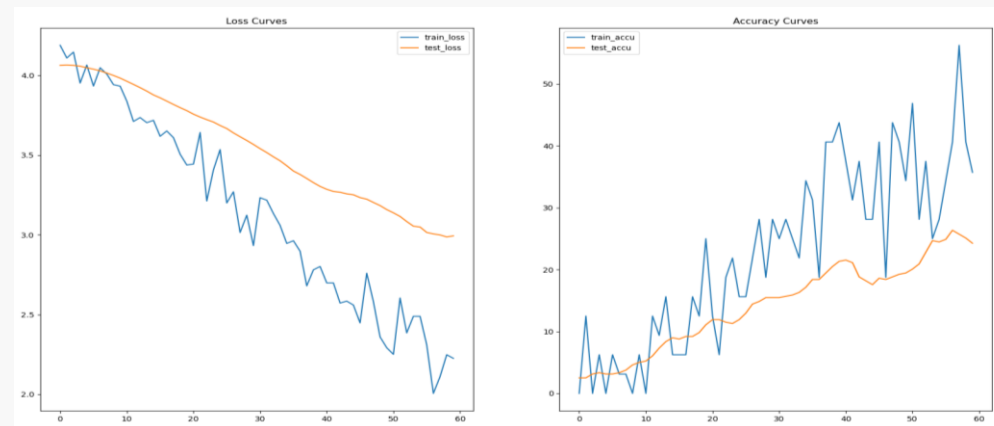
```
Starting evaluation
15it [00:02, 6.73it/s]
Test acc for image : 478 ACC : 42.89
End test..
```



MobileNet

Epoch	3 / 3	Loss Function	CrossEntropy
Batch size	32	Optimizer	SGD
Momentum	0.9	Learning rate	0.001

```
Starting evaluation
15it [00:02, 6.64it/s]
Test acc for image : 478 ACC : 32.01
End test..
```



# 02 프로젝트 과정

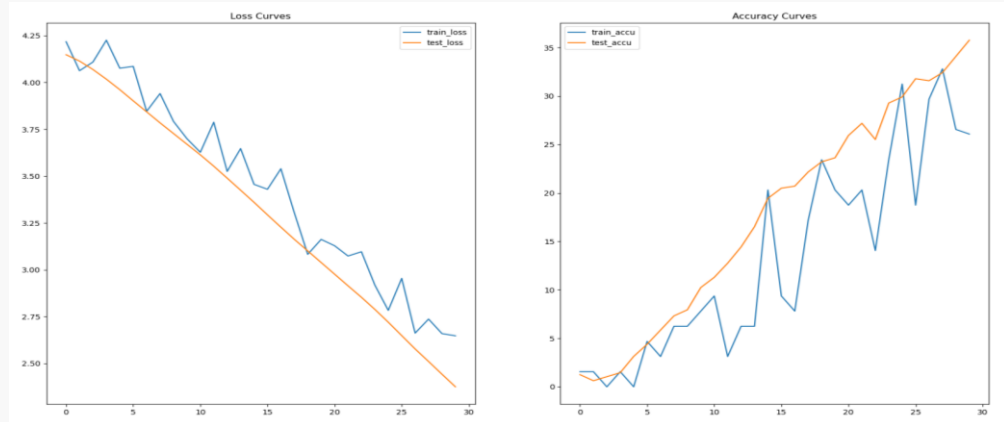
데이터 학습 기록, 모델 선정

샘플 데이터 학습으로 비교해서 가장 좋은 모델 선정

AlexNet

Epoch / Valid	3 / 3	Loss Function	CrossEntropy
Batch size	32	Optimizer	SGD
Momentum	0.9	Learning rate	0.025

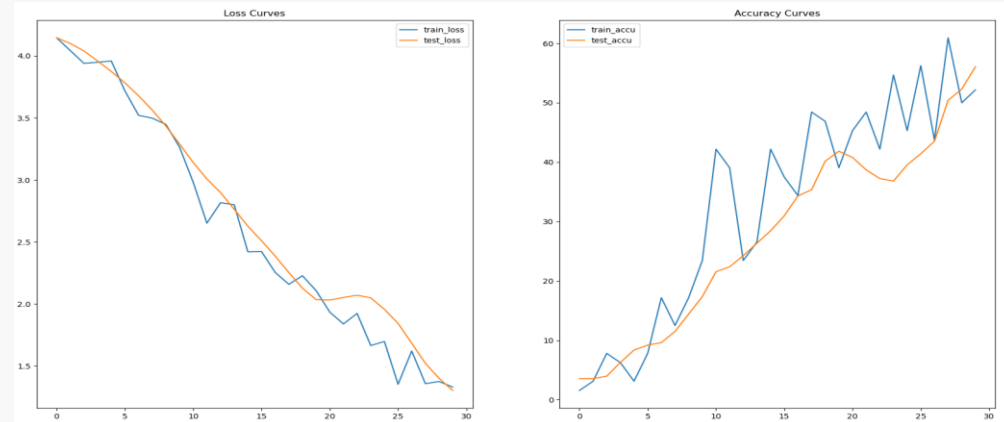
```
Starting evaluation
15it [00:02, 7.28it/s]
Test acc for image : 478 ACC : 11.72
End test..
```



ResNet18

Epoch / Valid	3 / 3	Loss Function	CrossEntropy
Batch size	32	Optimizer	SGD
Momentum	0.9	Learning rate	0.005

```
0it [00:00, ?it/s]Starting evaluation
8it [00:02, 3.60it/s]
Test acc for image : 478 ACC : 56.07
End test..
```

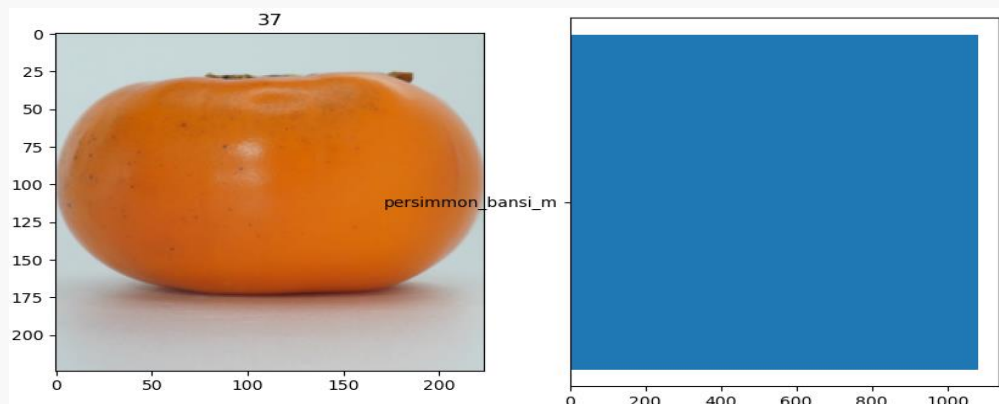


# 02 프로젝트 과정

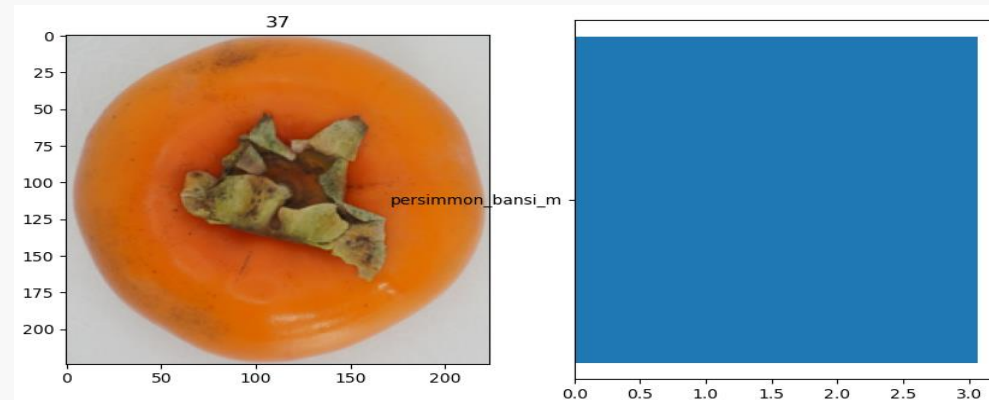
데이터 학습 기록, 모델 선정

샘플 데이터 학습으로 비교해서 가장 좋은 모델 선정

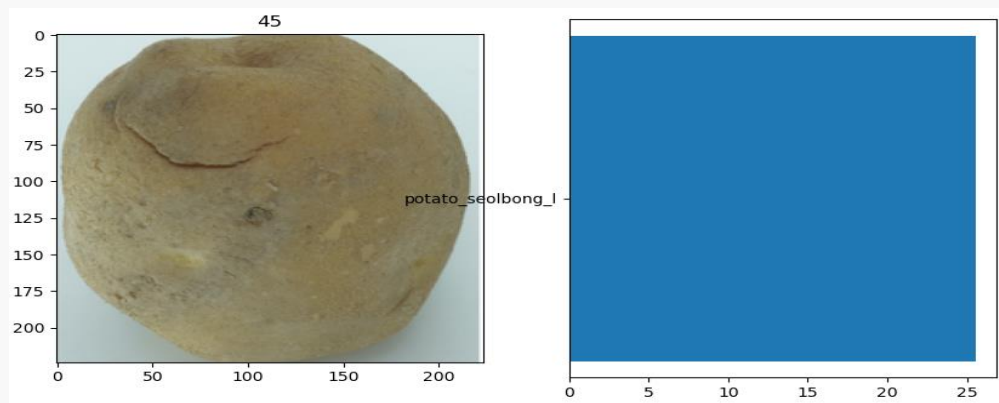
GhostNet



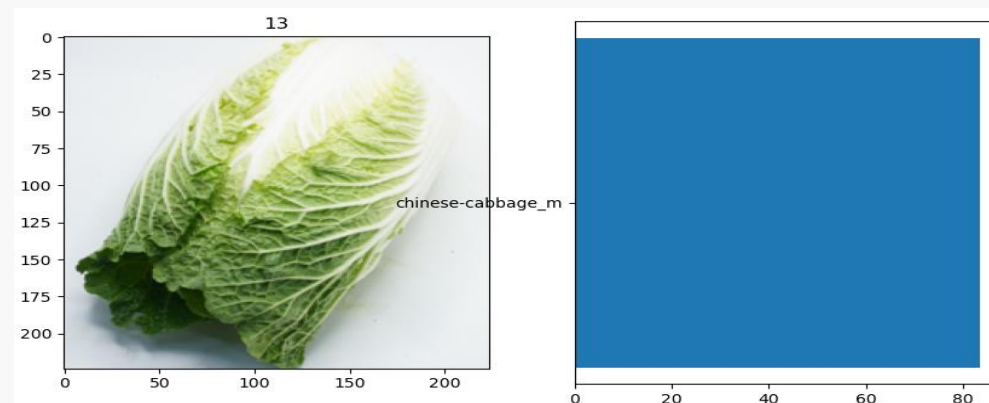
MobileNet



AlexNet



ResNet18



## 02 프로젝트 과정

데이터 학습 기록, 모델 선정

샘플링 한 원본데이터로 학습한 것 중 결과가 괜찮은 모델

MobileNet

Epoch / Valid	10 / 10	Loss Function	CrossEntropy
Batch size	32	Optimizer	SGD
Momentum	0.9	Learning rate	0.001

```
Starting evaluation
627it [01:08, 9.18it/s]
Test acc for image : 20052 ACC : 77.41
End test..
```

GhostNet

Epoch / Valid	10 / 10	Loss Function	CrossEntropy
Batch size	64	Optimizer	SGD
Momentum	0.9	Learning rate	0.0025

```
0it [00:00, ?it/s]Starting evaluation
627it [01:08, 9.13it/s]
Test acc for image : 20052 ACC : 79.62
End test..
```

ResNet18

Epoch / Valid	10 / 10	Loss Function	CrossEntropy
Batch size	32	Optimizer	SGD
Momentum	0.9	Learning rate	0.005

```
Starting evaluation
627it [01:08, 9.17it/s]
Test acc for image : 20052 ACC : 81.94
End test..
```



VGG16

Epoch / Valid	10 / 10	Loss Function	CrossEntropy
Batch size	32	Optimizer	SGD
Momentum	0.9	Learning rate	0.0025

## 메타데이터 학습 (선형회귀)

## Json 파일에서 필요한 변수 선택 후 CSV 추출

	group_no	cate1	cate2	cate3	width	height	weight
0	601031001000	사과	부사	특	9.7	9.0	450.0
1	601031001000	사과	부사	특	9.7	9.0	450.0
2	601031001000	사과	부사	특	9.7	9.0	450.0
3	601031001000	사과	부사	특	9.7	9.0	450.0
4	601031001000	사과	부사	특	9.7	9.0	450.0
	group_no	cate1	cate2	cate3	width	height	weight
0	601031001000	사과	부사	특	9.7	9.0	450.0
40	601031010000	사과	부사	특	9.6	8.9	385.0
80	601031011000	사과	부사	특	9.4	8.7	415.0
120	601031012000	사과	부사	특	9.7	8.7	390.0
160	601031013000	사과	부사	특	9.7	9.4	430.0

[illegible]

- 중복이 있는 전체 json 파일을 읽어서 csv 파일을 생성
- csv를 불러와서 중복을 제거한 csv파일을 생성

## linear regression을 사용하여 메타데이터 학습

[illegible]

- 품목\_품종\_등급에 해당하는 54개의 라벨은 순서가 없는 명목변수 이므로 one hot encoding 처리  
x에 범주형 변수, y는 width, height, weight

```
Epoch 30000/30000 Cost: 13116.337891
예측값 : tensor([ 9.5631,  8.8576, 411.1672], grad_fn=<AddBackward0>)
tensor([ 9.6000,  8.9000, 385.0000])
```

- $(\text{예측 값} - \text{실제 값})^2$ 의 차이가 적도록 학습
- 예측 값은 원 데이터의 라벨 별 평균값과 유사
- 위의 이미지는 '사과부사특'일 경우 너비, 높이, 무게

## 02 프로젝트 과정

### 홈페이지 첫 화면

#### ⚙️ 농산물 등급 판정

cabbage\_red\_1\_1-8\_3fr90.png   
판정에 사용할 사진을 업로드해주세요.

**1**   **2**  
파일이 선택되지 않았습니다.  
판정에 사용할 사진을 업로드해주세요.

- Flask로 훈련시킨 모델을 호출하고 서빙하는 웹 페이지를 app.py와 index.html로 구현

### 이미지 학습이 완료된 모델 호출

```
# 모델 불러오기
model, size = models.initialize_model("resnet18", 54, use_pretrained=False)
model.load_state_dict(torch.load("./resnet18_best.pt"))

# 이미지 open, 전처리
image = Image.open(filename).convert("RGB")
image = test_transform(image)
image = image.unsqueeze(0)

# 모델에 넣어 결과 가져오기
with torch.no_grad():
    model.eval()
    # Model outputs log probabilities
    output = model(image)
    output = torch.exp(output)
    _, index = torch.max(output, 1)
    label = classes[index]

# print("#####", label, "##", output, "##", index, ">>>>")

width, height, weight = predictvalue.predict(index)

return label, str(output[0][index].item())[:5], width.item(), \
    height.item(), weight.item()
```

- 이미 학습이 완료된 resnet18\_best.pt를 호출하여 업로드 된 이미지를 open, 전처리하고 결과를 return

## 02 프로젝트 과정

### 이미지 업로드한 화면

찾아보기... 파일이 선택되지 않았습니다. upload



품목 : 양파

품종 : 적색

등급 : 특

가로 : 10.633 cm

세로 : 10.108 cm

무게 : 440.04 g

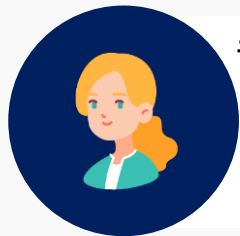
- 업로드 된 이미지를 미리 훈련이 완료된 resnet18 모델로 어떤 품목, 품종, 등급을 가진 농산물인지 확인 가능

### 품목, 품종, 등급 예측 모델 호출

```
def predict(label):  
    data=pd.get_dummies(classes)  
    # label=torch.Tensor([0])  
    label=int(torch.Tensor.numpy(label)[0])  
    for i in range(0,54):  
        if i==classes[label]-1:  
            x=data.values[i]  
    x=np.array(x)  
    x=torch.Tensor(x)  
  
    model = nn.Linear(54,3)  
    model.load_state_dict([loaded_model])  
    model.eval()  
    with torch.no_grad():  
        outputs = model(x)  
        width=outputs[0]  
        height=outputs[1]  
        weight=outputs[2]  
  
    print("STAT", width, height, weight)  
    return width, height, weight
```

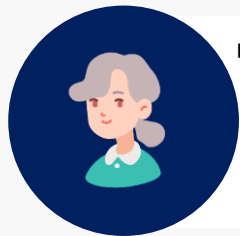
- 등급에 따라 가로, 세로, 무게가 어느 정도 인지 linear regression 학습을 통한 결과 모델로 예측 가능

## 04 프로젝트 소감



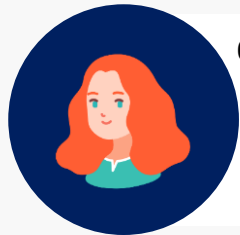
권보라

수업 때는 이해가 되지 않던 부분을 그냥 넘어가버리곤 해서 딥러닝에 대한 이해가 부족한 상태였는데, 팀원들과 함께 프로젝트를 진행해보니 빈틈이 메꿔지는 느낌이 들었고 분류와 회귀까지 전체적인 프로세스를 훑어보는 경험이 되어 유익했습니다.



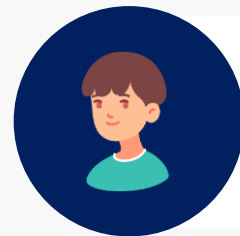
박나은

확실히 수업으로 듣는것보다 내가 직접 해보니까 습득력과 이해도가 좀 더 채워지는 느낌을 받았다. 아직 엉성한 부분도 있고 완벽하진 않지만 팀원들과 함께 프로젝트 진행한 것 만으로도 충분한 배움을 얻을 수 있었다.



이나겸

AI hub에서 받은 파일이지만 이미지와 annotation 파일 이름이 매칭되지 않아서 파일 이름이 맞는지 확인하는 작업이 제일 오래 걸렸다. 학습하기 전에 데이터를 준비하고 확인하는 부분부터 중요하다고 생각이 되어 앞으로는 이 부분을 소홀히 하지 않고 진행할 수 있을 것 같다.



정성우

잘 정제된 이미지를 가지고 모델학습을 해서 일반적인 사진에 대해서는 정확도가 떨어질 것 같아 아쉬움이 남지만 처음으로 이미지를 분류하면서 재미있었고 flask로 웹에 연동되는 부분이 신기했습니다.



Q & A

Q&A

THANK YOU

경청해주셔서 감사합니다.