INDEX

토마토 상태분석관리 웹 어플리케이션

설계 및 프로젝트 심화 I 8조

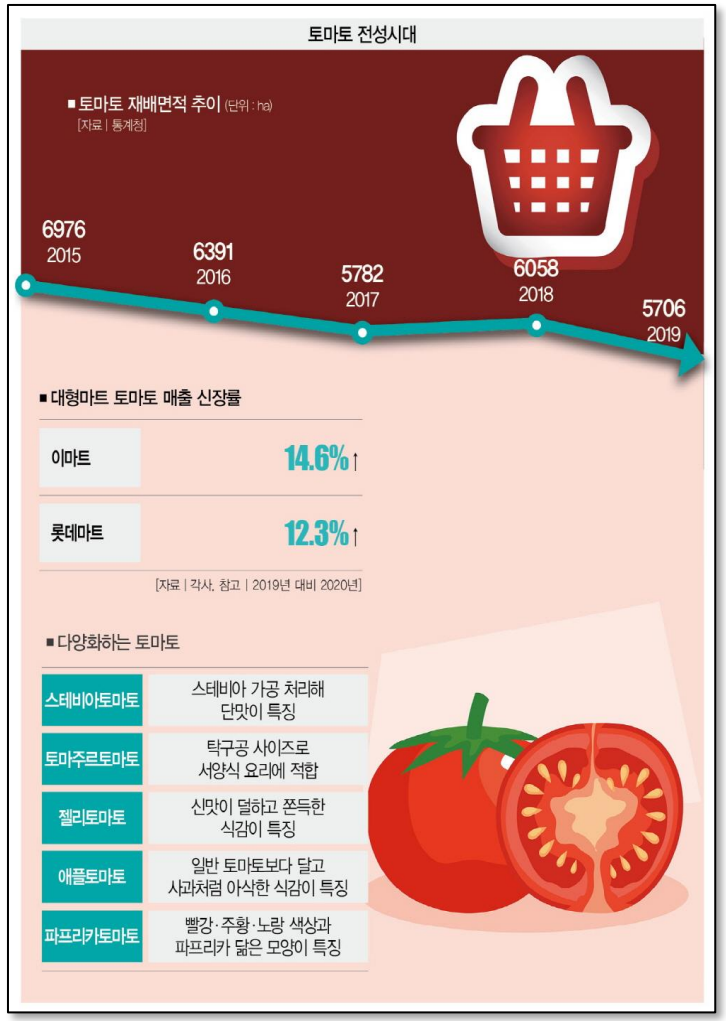
Project Manager
17615025 신준호

Project Engineer
17615024 송전영

Project Engineer
19615004 김미래

INDEX

01 프로젝트 소개



Data 탐색

인공지능을 학습시킬 Dataset을 찾던 중
최근 농산물 관리에도 AI가 사용됨을 알게 되었고
토마토 질병에 관한 Image Dataset을 발견함.

토마토의 시장성

많은 기사와 통계에서 토마토 소비의 증가를 나타내며 토마토
생산량 또한 증가할 것으로 예상
따라서 토마토 AI는 소비자에게 도움을 줄 수 있을 것.

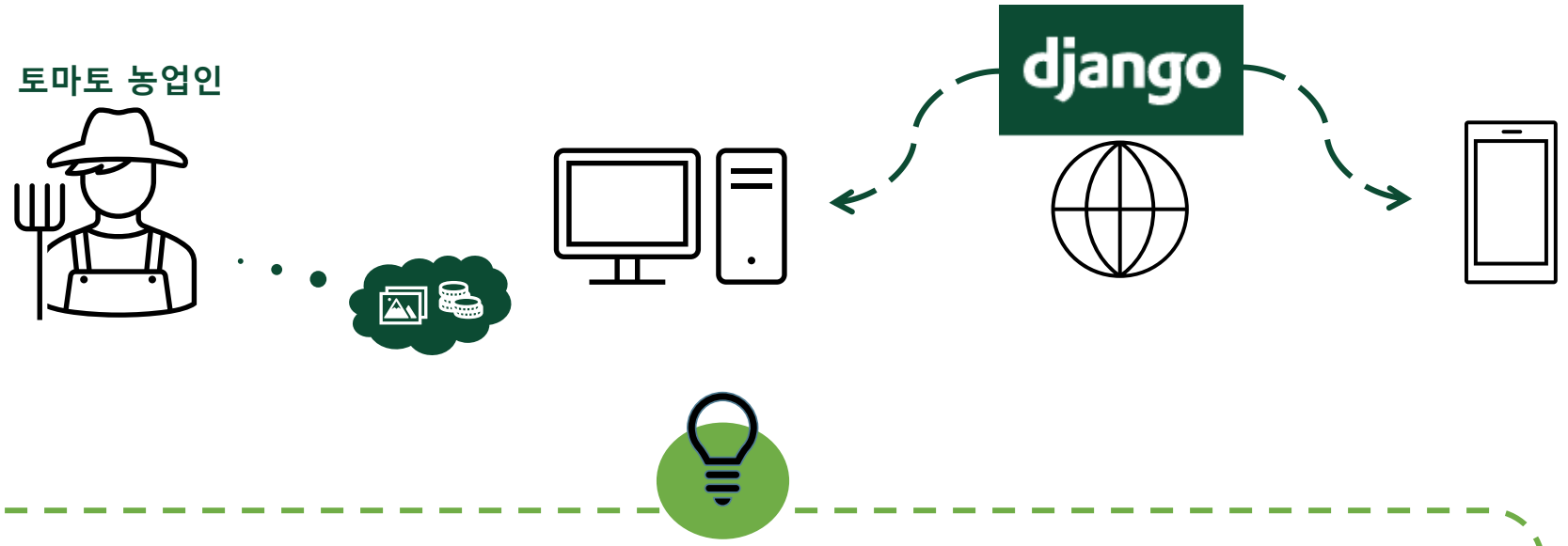
최근 대형마트에선 토마토 매출이 껑충 뛰었다. 이마트와 롯데마트에선 2020년 토마토 매출액이 전년 대비 각각 14.6%, 12.3% 증가했다. 홈플러스 역시 최근 한달 간(2020년 12월 12일~2021년 1월 11일) 토마토 매출 신장률이 전년 동기 대비 20.0%를 기록했다.

토마토가 인기를 끄는 이유는 별다른 게 아니다. 코로나19 사태가 확산하면서 '집밥족'이 증가한 게 영향을 미쳤다. 대형마트의 한 관계자는 "외식이 줄고 '집밥'을 먹는 횟수가 증가하면서 샐러드나 파스타에 쓰이는 토마토 수요가 늘었다"면서 "소비자의 니즈에 맞춰 다양한 종류의 토마토를 선보이고 있다"고 설명했다.
출처 : 더스쿠프(http://www.thescoop.co.kr)

농촌진흥청에서 토마토의 소비 트렌드를 조사한결과 과일->채소->간식 및 식사대용으로 변화 하였음

프로젝트 주제 선정

토마토 상태분석관리 웹 어플리케이션



AI 학습을 통해 토마토 병해충 감염 판독을 주요기능으로 하는

토마토 상태 분석 관리 “웹 어플리케이션”을 주제로 선정

프로젝트 진행 일정 토마토 상태분석관리 웹 어플리케이션

세부 개발 내용	추진 일정												비고
	4주차	5주차	6주차	7주차	8주차	9주차	10주차	11주차	12주차	13주차	14주차	15주차	
주제선정													
데이터 셋 관리													5,6주차 : 데이터셋 확보 7,8주차 : 데이터셋 확장 및 분류 9,10주차 : 데이터셋 정제 및 레이블링
딥러닝 모델 개발													6-12 주차 : 병충해 이미지 딥러닝 모델 9-12 주차 : 가격 예측 선형 모델
웹 개발													11,12주차 : 프론트엔드 13,14주차 : 백엔드(서버&데이터베이스)
프로그램 테스트 후 배포 (개발 완료)													

INDEX

02 프로젝트 주요 기능

주요 기능

토마토 상태분석관리 웹 어플리케이션

off

01 토마토 질병 인식

토마토의 잎 사진을 업로드하면 병충해 질병을 AI로 판별해 사진과 그래프를 보여준다.

02 토마토 가격 예측

기존의 토마토 가격 데이터를 활용해 AI로 다음날의 가격을 예측해서 알려준다.

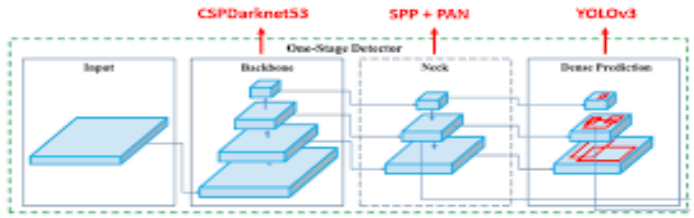
03 토마토 질병 도감

사용자에게 토마토 병충해 별 더 자세한 정보를 보여준다.

01

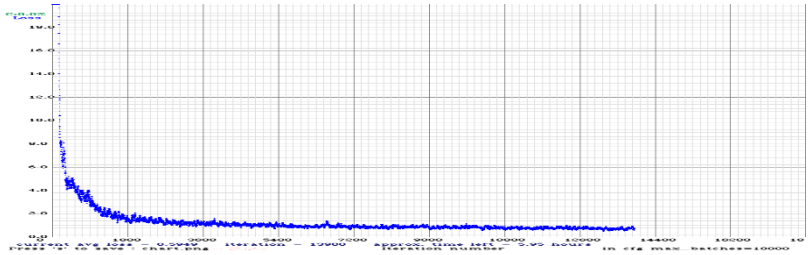
토마토 질병 인식

토마토의 잎 사진을 업로드하면 질병을 판별하여 사용자에게 사진과 그래프를 보여준다



YOLOv4 = YOLOv3 + CSPDarknet53 + SPP + PAN + BoF + BoS

Path Aggregation Network

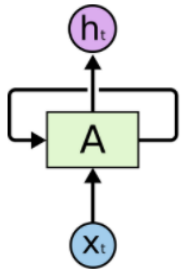


02

토마토 가격 예측

기존의 토마토 가격데이터를 활용해 다음날의 가격을 예측해서 보여준다.

	A	B	C	D
1	날짜	도매-상	도매-중	소매-상
2	20200207	36400	31600	5325
3	20200210	37000	32400	5352
4	20200211	37000	32400	5352
5	20200212	38600	33800	5352
6	20200213	35400	30800	5480
7	20200214	38400	33600	5546
8	20200217	41200	36400	5584
9	20200218	35200	30600	5584
10	20200219	39200	35000	5588
11	20200220	37400	33200	5892
12	20200221	38000	33500	5959
13	20200224	38000	33500	5988
14	20200225	38000	33500	5992
15	20200226	38000	33500	5996
16	20200227	38000	33500	5964
17	20200228	38000	33500	5999
18	20200302	39750	35250	5969
19	20200303	36250	31750	5950
20	20200304	40000	35500	5909
21	20200305	39000	34500	5976
22	20200306	42500	37750	5969

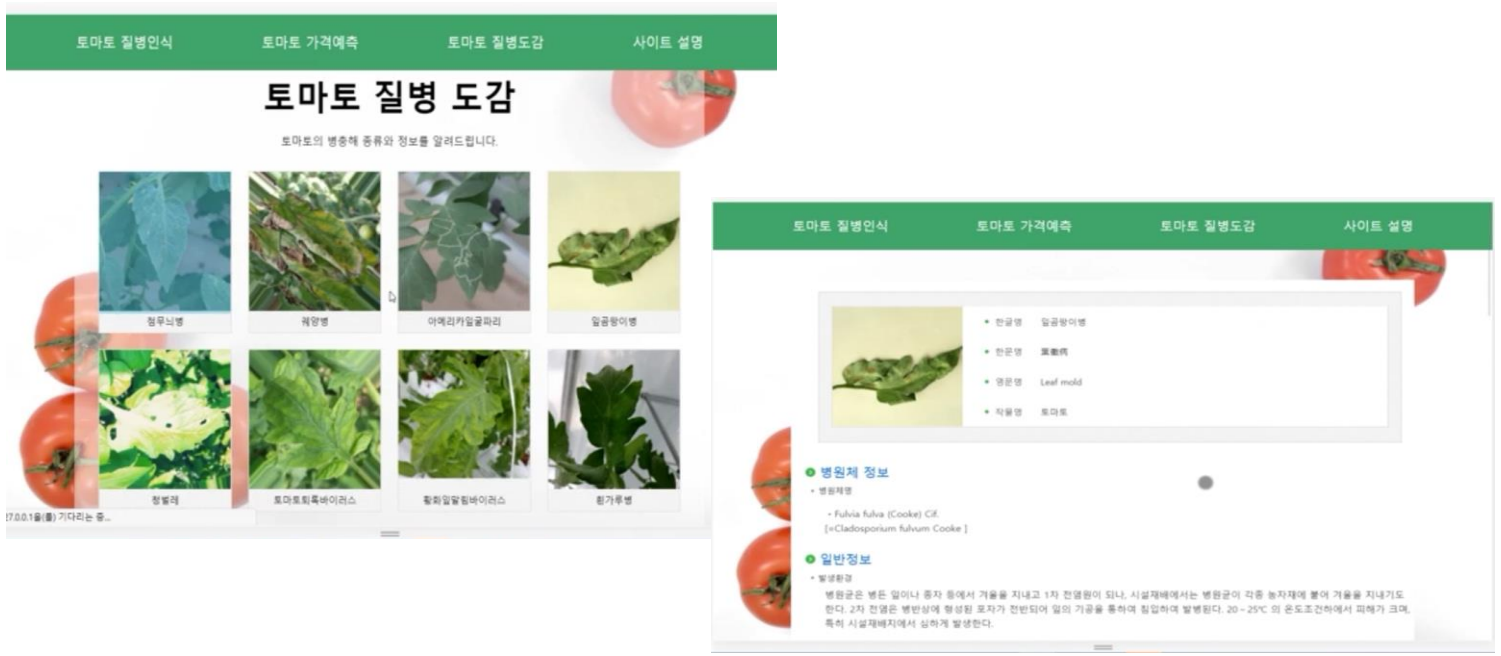



```
116/116 [==============================] - 4s 31ms/step - loss: 0.0010
Epoch 42/100
116/116 [==============================] - 4s 34ms/step - loss: 0.0013
Epoch 43/100
116/116 [==============================] - 4s 34ms/step - loss: 0.0012
Epoch 44/100
116/116 [==============================] - 4s 36ms/step - loss: 0.0012
Epoch 45/100
116/116 [==============================] - 4s 36ms/step - loss: 0.0013
Epoch 46/100
116/116 [==============================] - 4s 31ms/step - loss: 0.0014
Epoch 47/100
116/116 [==============================] - 4s 31ms/step - loss: 0.0012
Epoch 48/100
116/116 [==============================] - 4s 32ms/step - loss: 0.0013
Epoch 49/100
116/116 [==============================] - 4s 32ms/step - loss: 0.0011
Epoch 50/100
116/116 [==============================] - 4s 34ms/step - loss: 0.0012
```

03

토마토 질병 도감

토마토 병충해 별로 더 자세한 정보를 사용자에게 보여준다.

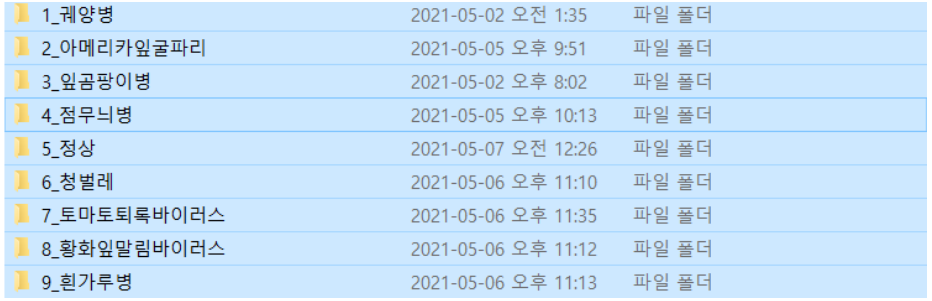


INDEX

03 딥러닝 학습 과정

1. 토마토 질병 객체 인식 모델

병충해명	데이터 사이즈
정상	약 170GB
점무늬병	27GB
아메리카잎굴파리	34.5GB
잎곰팡이병	8.6GB
청벌레	1.5GB
토마토퇴록바이러스	1.5GB
황화잎말림바이러스	5.1GB
흰가루병	634MB
궤양병	1.64GB



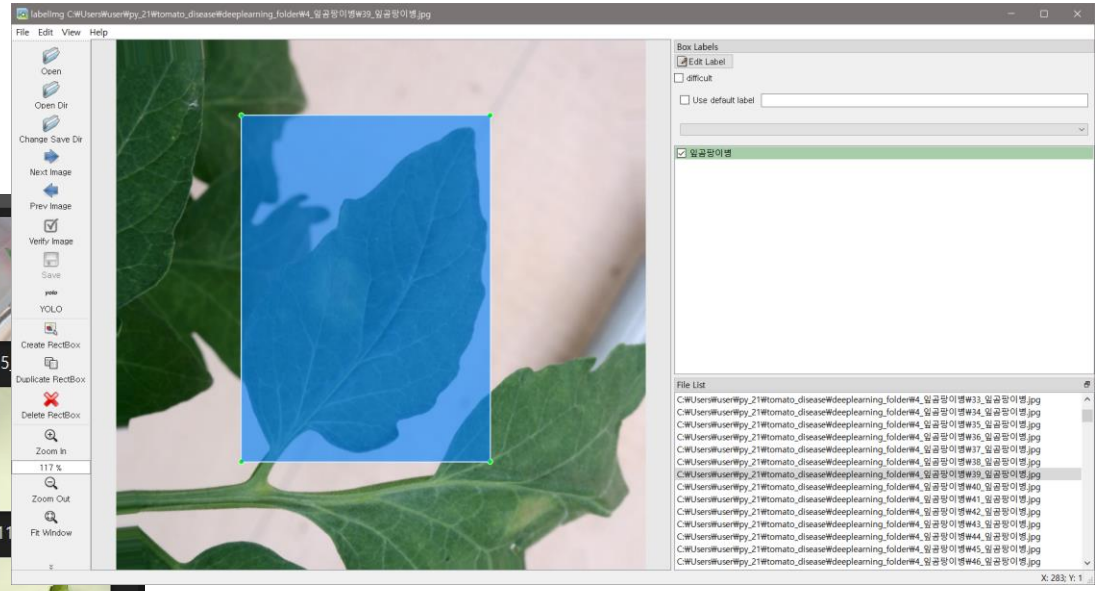
데이터셋의 이미지 개수의 차이로 1000개로 설정
: 부족한 데이터셋에서는 이미지 확장 코드를 이용해 늘리도록 하고,
많은 데이터셋에서는 특정한 이미지만 선별해 사용

```
data_datagen = ImageDataGenerator(rescale=1./255)
data_datagen = ImageDataGenerator(rescale=1./255,
                                  rotation_range=15,
                                  shear_range=0.5,
                                  # width_shift_range=0.1,
                                  # height_shift_range=0.1,
                                  horizontal_flip=True,
                                  vertical_flip=True,
                                  fill_mode='nearest')
```

△ ImageDataGenerator()사용 : 이미지 개수 확장

(테스트셋)1347 : (학습셋)7637 총 8984장

1. 토마토 질병 객체 인식 모델



각 병충해 분류에 따라 1000개씩 선별된 데이터에 병충해가 포함된 나뭇잎 부분만 따로 레이블링 후 최종 데이터셋 정리 완료

토마토 병충해 이미지 학습 방법 토마토 상태분석관리 웹 어플리케이션

off

1. 토마토 질병 객체 인식 모델

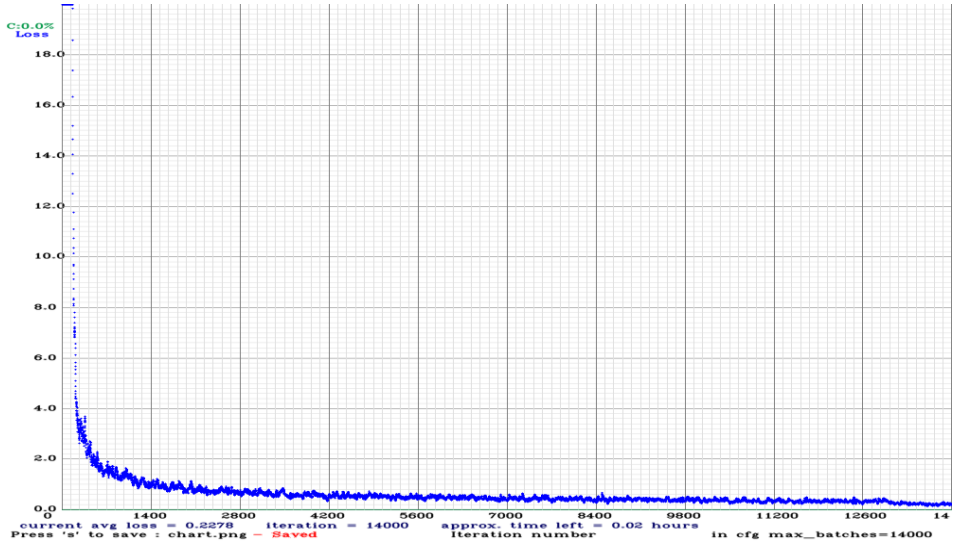
```
!darknet/darknet detector train custom_data/labelled_data.data darknet/cfg/yolov3_custom.cfg custom_weight/darknet53.conv.74 -dont_show
```

다크넷으로 yolo v3와 v4의 커스터마이징 이후 코랩의 gpu를 사용해 20시간 동안 학습

```
1480: 0.208085, 0.226324 avg loss, 0.001000 rate, 10.579943 seconds, 94720 images, 57.397319 hours left Resizing, random_coef =
1.40 576 x 576 try to allocate additional workspace_size = 0.04 MB CUDA allocate done! Loaded: 0.000052 seconds v3 (mse loss,
Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.807994), count: 4, class_loss = 0.265819, iou_loss = 0.181560,
total_loss = 0.447379 v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.000000), count: 1,
class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000 v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00)
Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000 total_bbox = 94580,
rewritten_bbox = 0.000000 % v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.709207), count: 4,
class_loss = 0.827464, iou_loss = 0.171634, total_loss = 0.999098 v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00)
Region 94 Avg (IOU: 0.000000), count: 1, class_loss = 0.000002, iou_loss = 0.000000, total_loss = 0.000002 v3 (mse loss,
Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss =
0.000000, total_loss = 0.000000 total_bbox = 94584, rewritten_bbox = 0.000000 %
```

학습 중 일부 텍스트

학습된 모델 생성



1. 토마토 질병 객체 인식 모델

Darknet이란?

- 오픈 소스 신경망 프레임워크

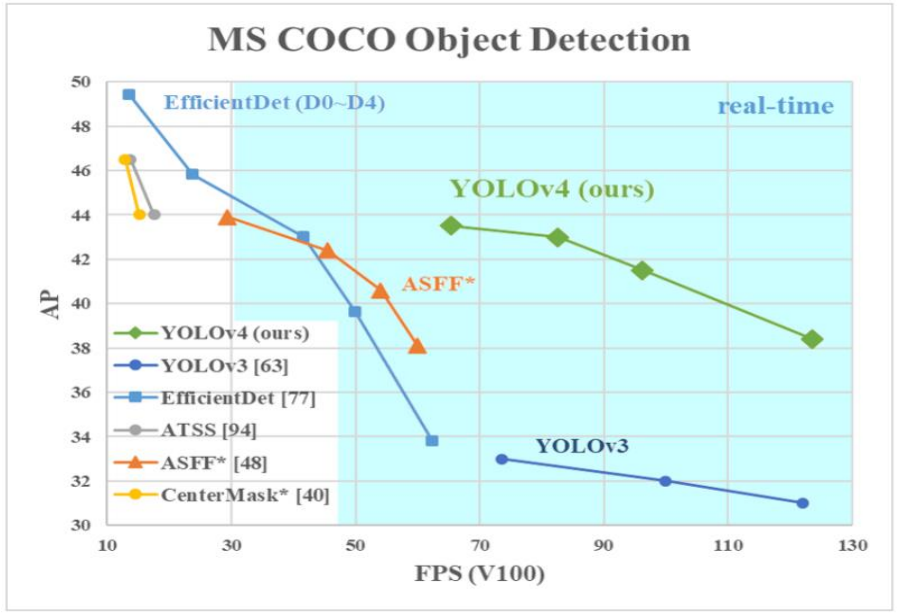
- C와 cuda로 작성되었기 때문에 빠르다
  Cuda : many core dependent 연산

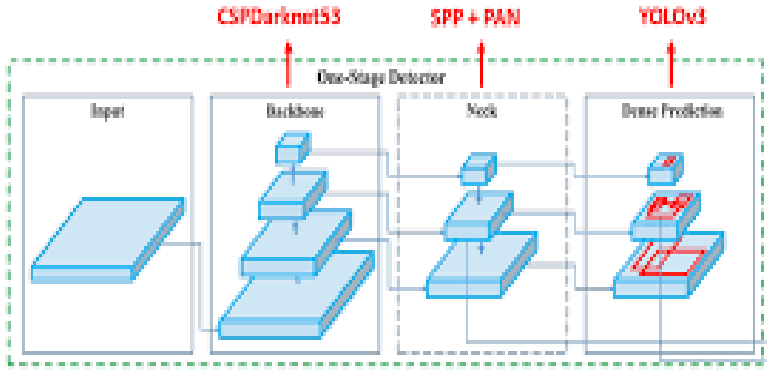
- 실시간 객체감지 및 이미지에서 객체감지를 위한 프레임워크

YOLO V4란

토마토 상태분석관리 웹 어플리케이션

YOLO v4는 YOLO v3이후에 나온 딥러닝의 정확도를 개선하는 다양한 방법을 적용해 YOLO의 성능을 극대화 하였다.





YOLOv4 = YOLOv3 + CSPDarknet53 + SPP + PAN + BoF + BoS

Path Aggregation Network

학습된 모델 테스트

토마토 상태분석관리 웹 어플리케이션

off

1. 토마토 질병 객체 인식 모델

테스트는 각 질병별로 거의 같은 비율의 총 1347개의 이미지를 사용

질병 명	정확도(X는 미 실시)
궤양병	X
아메리카잎굴파리	X
잎곰팡이병	X
점무늬병	X
정상	X
청벌레	X
토마토퇴록바이러스	X
황화잎말림바이러스	X
흰가루병	X
전체	59.99%
전체 평균	X

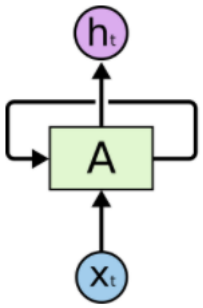
약 37% 상승

질병 명	정확도
궤양병	86.5%
아메리카잎굴파리	91.95%
잎곰팡이병	97.93%
점무늬병	98.64%
정상	100%
청벌레	97.5%
토마토퇴록바이러스	99.31%
황화잎말림바이러스	98.67%
흰가루병	100%
전체	96.59%
전체 평균	96.72%

전처리 및 토마토 가격 예측 모델 학습방법 토마토 상태분석관리 웹 어플리케이션

2. 토마토 가격 예측 모델

1) 데이터 셋 확보 및 데이터 정제



LSTM 이란?

LSTM은 RNN의 변형 버전.

RNN은 순환신경망으로 다른 NN과 다르게 **자신의 아웃풋을 다시 인풋으로 사용**하며, 이 과정에서 시계열처럼 긴 데이터도 사용할 수 있도록 개선한 것이 LSTM이다.

* 토마토의 가격 변동 – 시계열 데이터 형태로 나타남.

	A	B	C	D	E
1	year	도매-상	도매-중	소매-상	
2	20160502	22800	18800	3926	
3	20160503	21600	17600	3887	
4	20160504	20200	16400	3793	
5	20160509	19200	15600	3580	
6	20160510	18400	15000	3482	
7	20160511	18000	14600	3450	
8	20160512	18000	14600	3638	
9	20160513	17800	14200	3621	
10	20160516	17800	14200	3609	
11	20160517	16400	12800	3609	
12	20160518	16000	12400	3578	
13	20160519	15800	12200	3317	
14	20160520	16000	12400	3304	
15	20160523	16400	12800	3304	
16	20160524	16400	12800	3304	

10일치 가격 변동 학습

11일째의 가격 예측

.
.
.

x 1221개의 데이터에서 반복

2. 토마토 가격 예측 모델

```python
look_back=30
batch_size=5

df = pd.read_csv('tomato_1.csv',encoding='cp949')
dataset = df.values

# 학습용 데이터 세트 생성
# 'look_back=10'일 때 'x = [[[x0], .. [x10]] [[x1], .. [x11]] ..], y = [[x11], [x12] ..]'의 형태로 학습 데이터가 만들어진다.
def create_dataset(signal_data, look_back):
    x_arr, y_arr = [], []
    i=1
    for i in range(1,len(signal_data) - look_back):
        x_arr.append(signal_data[i:(i+look_back), 0])
        y_arr.append(signal_data[i+look_back, 0])
    x_arr = numpy.array(x_arr)
    print(x_arr)
    x_arr = numpy.reshape(x_arr, (x_arr.shape[0], x_arr.shape[1], 1))
    return x_arr, numpy.array(y_arr)


scaler = MinMaxScaler(feature_range=(0, 1))
trade_count = scaler.fit_transform(dataset[1:,0:1])  # 학습을 위해 데이터를 0~1의 값으로 정규화한다.

# 데이터 분리
# 훈련
train = trade_count[1:int(len(trade_count) * 0.5)]
# 검증
val = trade_count[int(len(trade_count) * 0.5):int(len(trade_count) * 0.75)]
# 시험
test = trade_count[int(len(trade_count) * 0.75):-1]

# 학습용 데이터 세트 생성
# look_back 파라미터의 값에 따라 다음 값을 예상하기 위해 과거의 데이터를 몇 건 사용할지 결정된다.
x_train, y_train = create_dataset(train, look_back)
x_val, y_val = create_dataset(val, look_back)
x_test, y_test = create_dataset(test, look_back)
```

<학습용 데이터 세트 생성 함수>

- 한번에 계산될 단위에 따라 묶어서 배열에 저장 되는 과정

<학습/검증/테스트 데이터 셋 분리>

<LSTM을 사용한 모델구조>

```python
# 학습 모델 구성
model = Sequential()
model.add(LSTM(32, batch_input_shape=(batch_size, look_back, 1), stateful=True, return_sequences=Tr
model.add(Dropout(0.2)) # overfitting을 막기 위해 20% 가량을 drop
model.add(LSTM(32, batch_input_shape=(batch_size, look_back, 1), stateful=True))
model.add(Dropout(0.2))
model.add(Dense(1))

# 학습 과정 설정
model.compile(loss='mean_squared_error', optimizer='adam')
```

<모델 학습 및 저장>

```python
# 모델 학습
for i in range(1):
    hist = model.fit(x_train, y_train, epochs=100, batch_size=batch_size, validation_data=(x_val, y_
#    with open('model/seq/trade_count_save_state_lstm.history_'+str(i), 'wb') as f :
#        pickle.dump(hist, f)
#    model.save('model/seq/trade_count_save_state_lstm.hdf5_'+str(i))
model.save('model/trade_count_save_state_lstm.hdf5')
```

```
116/116 [==============================] - 4s 31ms/step - loss: 0.0010
Epoch 42/100
116/116 [==============================] - 4s 34ms/step - loss: 0.0013
Epoch 43/100
116/116 [==============================] - 4s 34ms/step - loss: 0.0012
Epoch 44/100
116/116 [==============================] - 4s 36ms/step - loss: 0.0012
Epoch 45/100
116/116 [==============================] - 4s 36ms/step - loss: 0.0013
Epoch 46/100
116/116 [==============================] - 4s 31ms/step - loss: 0.0014
Epoch 47/100
116/116 [==============================] - 4s 31ms/step - loss: 0.0012
Epoch 48/100
116/116 [==============================] - 4s 32ms/step - loss: 0.0013
Epoch 49/100
116/116 [==============================] - 4s 32ms/step - loss: 0.0011
Epoch 50/100
116/116 [==============================] - 4s 34ms/step - loss: 0.0012
```

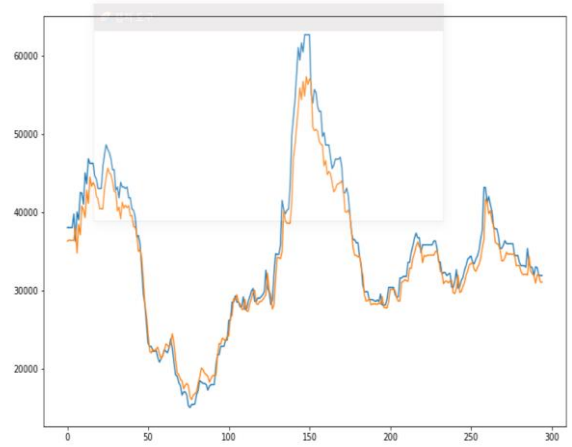
모델 테스트 결과 토마토 상태분석관리 웹 어플리케이션

2. 토마토 가격 예측 모델

```
predictions = model.predict(x_test, batch_size)
real_predictions = scaler.inverse_transform(predictions)  # 0-1의
print("predicted next 10 days trade count", real_predictions[-1])
#label = scaler.inverse_transform(y_test)
re_y=numpy.reshape(y_test, (y_test.shape[0], 1))
real_y=scaler.inverse_transform(re_y)
#real_y = scaler.inverse_transform(y_test)
for i in range(10):
    label = str(real_y[i])
    prediction = str(real_predictions[i])
    print('실제가격:'+label+'예상가격'+prediction)
```

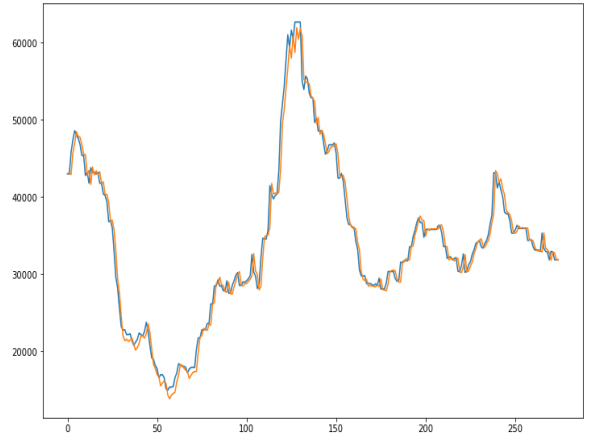
<10일 단위 학습 후 예측 결과>

실제가격:[38000.]예상가격[36277.516]
실제가격:[38000.]예상가격[36407.164]
실제가격:[38000.]예상가격[36364.47]
실제가격:[38000.]예상가격[36374.832]
실제가격:[39750.]예상가격[36365.477]
실제가격:[36250.]예상가격[38007.97]
실제가격:[40000.]예상가격[34749.035]
실제가격:[39000.]예상가격[38417.156]
실제가격:[42500.]예상가격[37074.527]
실제가격:[42400.]예상가격[40763.332]



<30일 단위 학습 후 예측 결과>

실제가격:[43000.]예상가격[43068.95]
실제가격:[43000.]예상가격[43125.555]
실제가격:[45800.]예상가격[42920.918]
실제가격:[47400.]예상가격[45711.438]
실제가격:[48600.]예상가격[46961.832]
실제가격:[48000.]예상가격[48470.016]
실제가격:[47600.]예상가격[47837.984]
실제가격:[46800.]예상가격[47753.21]
실제가격:[45400.]예상가격[46796.348]
실제가격:[45400.]예상가격[45578.656]

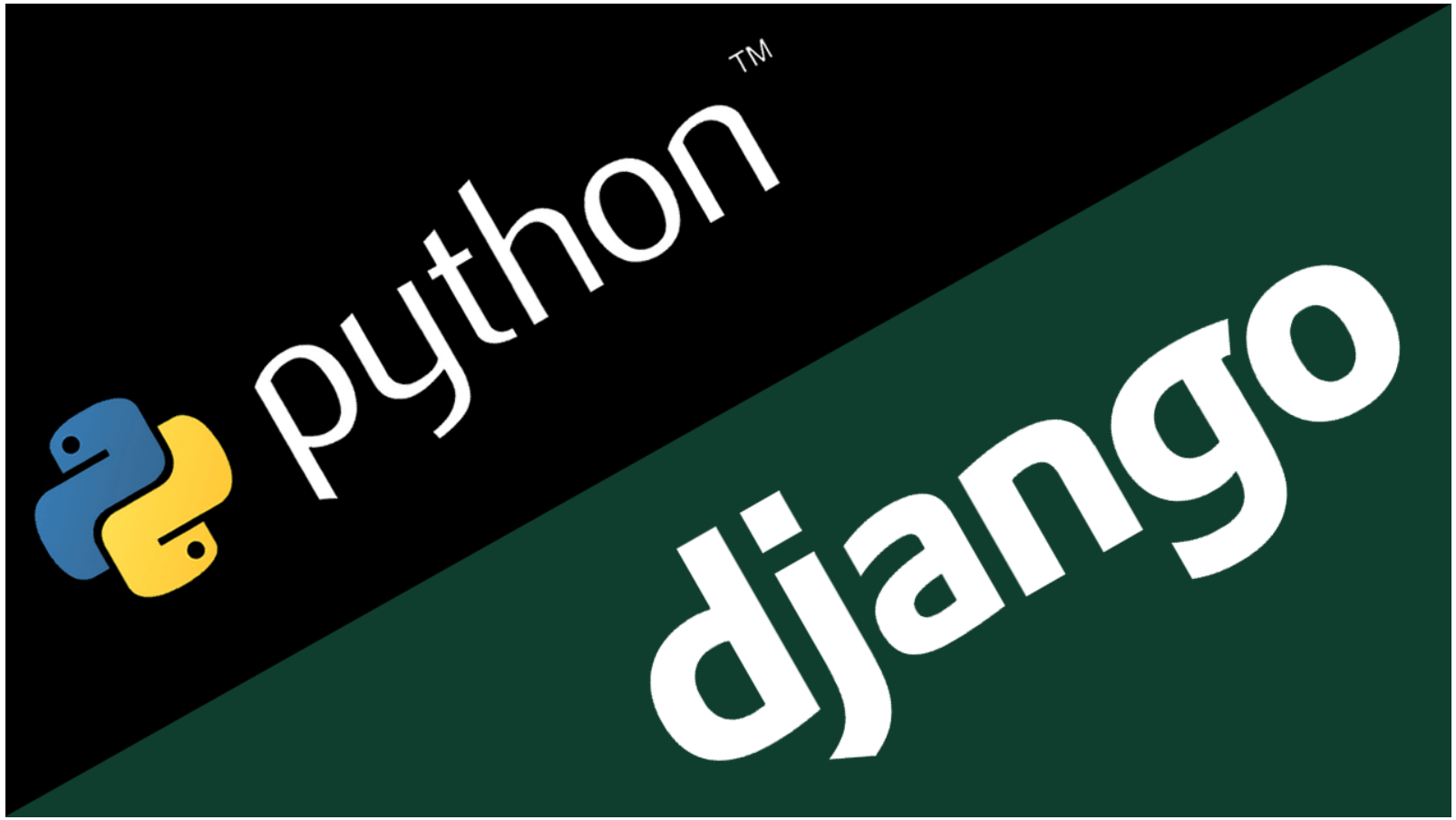


-> 가장 적은 손실률을 얻을 수 있었음. **이 모델로 추후 도매, 소매로 분류해 웹에 적용시켜 사용할 예정**

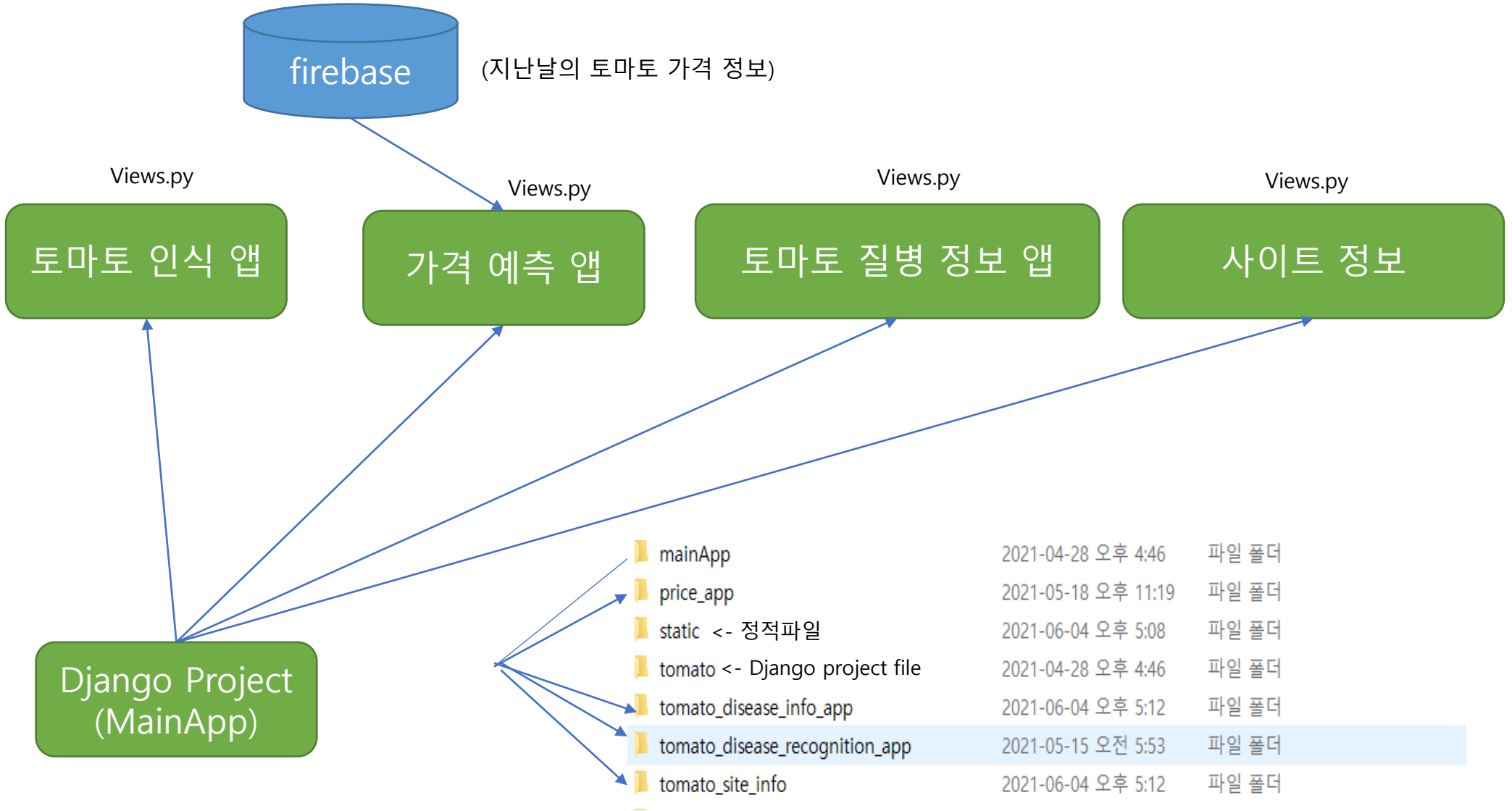
INDEX

04 시스템 구조

개발 환경



시스템 구조 토마토 상태분석관리 웹 어플리케이션



토마토 질병 인식 앱 모델 테스트 코드 함수화 및 적용

```python
def detect_disease_of_tomato(image):
    net= cv2.dnn.readNetFromDarknet('static/models/imageModel/yolov4_custom_of_custom.cfg','static/models/imageModel/yolov4_custom_last.weights')
    classes = []
    with open('static/models/imageModel/classes.names') as f:
        classes = [line.strip() for line in f.readlines()]
    wt,ht,_ =image.shape
    blob = cv2.dnn.blobFromImage(image,1/255,(416,416),(0,0,0),swapRB=True,crop=False)
    net.setInput(blob)
    last_layer = net.getUnconnectedOutLayersNames()
    layer_out = net.forward(last_layer)


    boxes = []
    confidences = []
    class_ids = []

    for output in layer_out:
        for detection in output:
            score = detection[5:]
            class_id =  np.argmax(score)
            confidence = score[class_id]
            if confidence > .5:
                center_x = int(detection[0] * wt)
                center_y = int(detection[1] * ht)
                w = int(detection[2]*wt)
                h = int(detection[3]*ht)

                x= int(center_x-w/2)
                y = int(center_y-h/2)

                boxes.append([x,y,w,h])
                confidences.append((float(confidence)))
                class_ids.append(class_id)


    indexes = cv2.dnn.NMSBoxes(boxes,confidences,0.3,0.9)
    font = cv2.FONT_HERSHEY_PLAIN
    colors = np.random.uniform(0,255,size=(len(boxes),3))



    label_and_confidence_dict = {'label':[],'confidence':[]}

    if(len(indexes) != 0):
        for i in indexes.flatten():
            x,y,w,h = boxes[i]
            label = str(classes[class_ids[i]])
            confidence = str(round(confidences[i],2))
            label_and_confidence_dict['label'].append(label)
            label_and_confidence_dict['confidence'].append(confidence)
            color = colors[i]
            cv2.rectangle(image,(x,y),(x+w,y+h),color,2)
            cv2.putText(image,label + ' '+confidence,(x,y+20),font,2,(0,0,0),2)


    return image,label_and_confidence_dict
```

라벨링된 이미지 + 예측된 classes + 예측된 classes의 accuracy return

토마토 가격 예측 앱 모델 테스트 코드 함수화 및 적용

```
def predict_price(data_array_01):
    look_back=10
    batch_size=2
    dataset=[]
    dataset = [data_array_01[i * 1:(i + 1)] for i in range(len(data_array_01))]
    dataset=numpy.array(dataset)
    def create_dataset(signal_data, look_back):
        x_arr, y_arr = [], []
        i=1
        for i in range(1,len(signal_data) - look_back):
            x_arr.append(signal_data[i:(i+look_back), 0])
            y_arr.append(signal_data[i+look_back, 0])
        x_arr = numpy.array(x_arr)
        print(x_arr)
        x_arr = numpy.reshape(x_arr, (x_arr.shape[0], x_arr.shape[1], 1))
        return x_arr, numpy.array(y_arr)
    scaler = MinMaxScaler(feature_range=(0, 1))
    trade_count = scaler.fit_transform(dataset[1:,0:1])  # 학습을 위해 데이터
를 0~1의 값으로 정규화한다.
    test = trade_count[int(len(trade_count) * 0):-1]
    x_test, y_test = create_dataset(test, look_back)
    #여기부터 테스트 출력
    model = load_model('static/models/priceModel/trade_count_save_state_lstm_tra
in.hdf5')
    predictions = model.predict(x_test, batch_size)
    real_predictions = scaler.inverse_transform(predictions)  # 0~1의 값으로 정규
화된 값을 원래의 크기로 되돌린다.
    return(int(real_predictions[-1]))
```

다음날 예측 값 return

클라이언트 - 서버 통신 부분 – ajax(xhr) 사용

```
function sendSignal() {

    var myHeaders = new Headers();
    myHeaders.append('Content-Type', 'application/json');

    var myInit = {
        method: 'POST',
        headers: myHeaders,
        body: {}
    };
    var body = document.querySelector('body');
    body.appendChild(spinner.el);
    return fetch('', myInit).then(response => response.json()).then(data => {
        body.removeChild(spinner.el)
        console.log(data)
        document.getElementById('price1').textContent =  data['pre_01']  + " 원"
        document.getElementById('price2').textContent = data['pre_02']  + " 원"
        document.getElementById('price3').textContent = data['pre_03']  + " 원"
    }).catch(error => {
        body.removeChild(spinner.el)
        alert(error);
    })


}
```

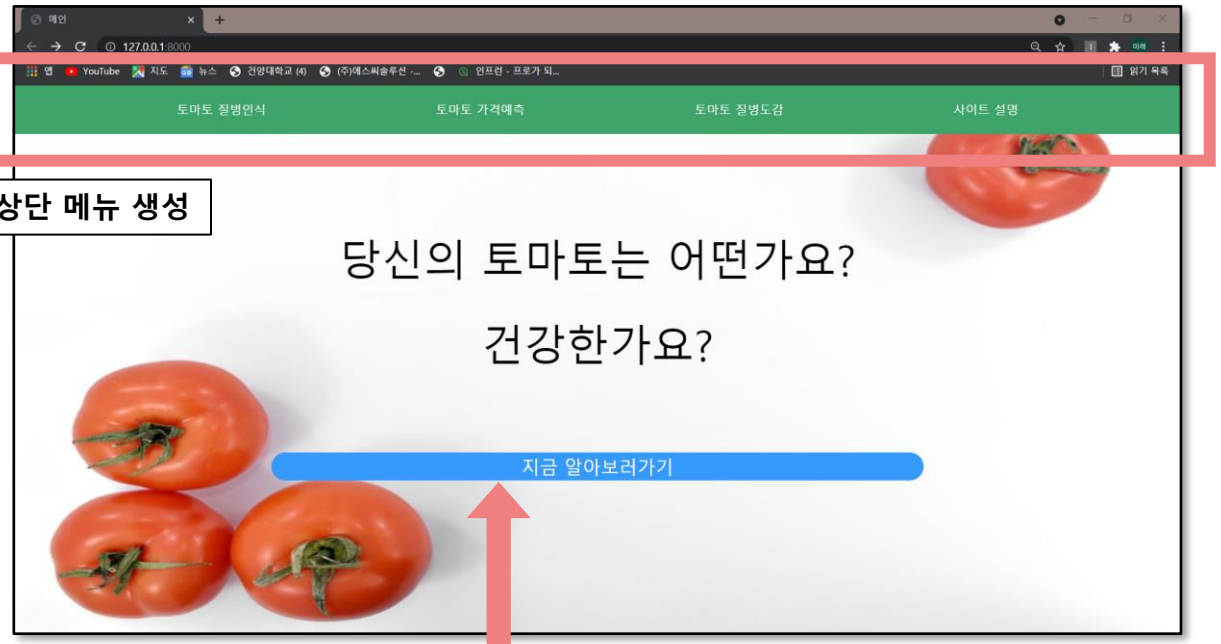
서버와 클라이언트 연결 시 새로고침이 되지 않는다.

INDEX

05 사용자 시나리오

<초기 화면>

주요기능인 토마토 질병인식을 위한 바로가기 버튼 및
추가 기능 접속 4가지가 메뉴를 통해 보여진다.



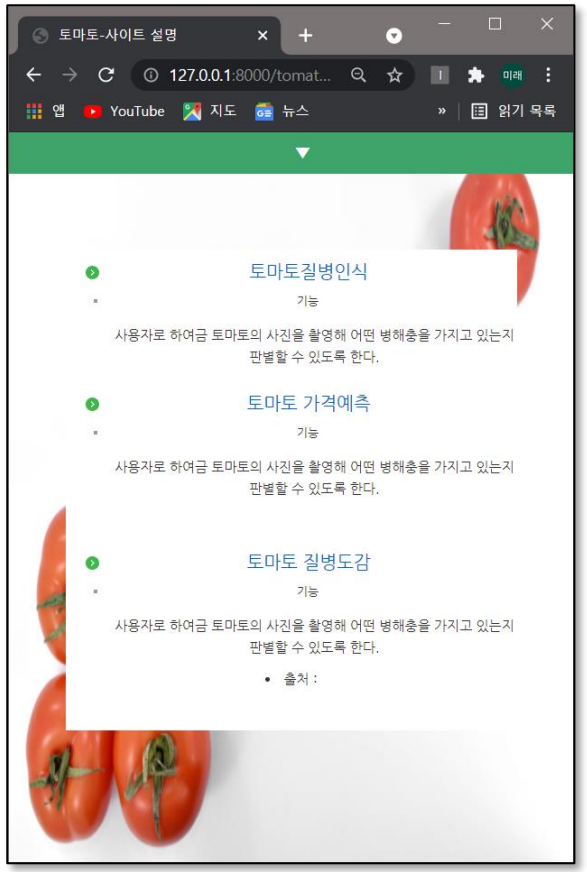
화면 상단 메뉴 생성

클릭 시 토마토 질병인식 메뉴로 바로 이동

<모바일 화면 크기 실행시>

- 메뉴 및 화면 구성요소가 모바일 환경으로 자동 변환

< 사이트 설명 메뉴 화면>

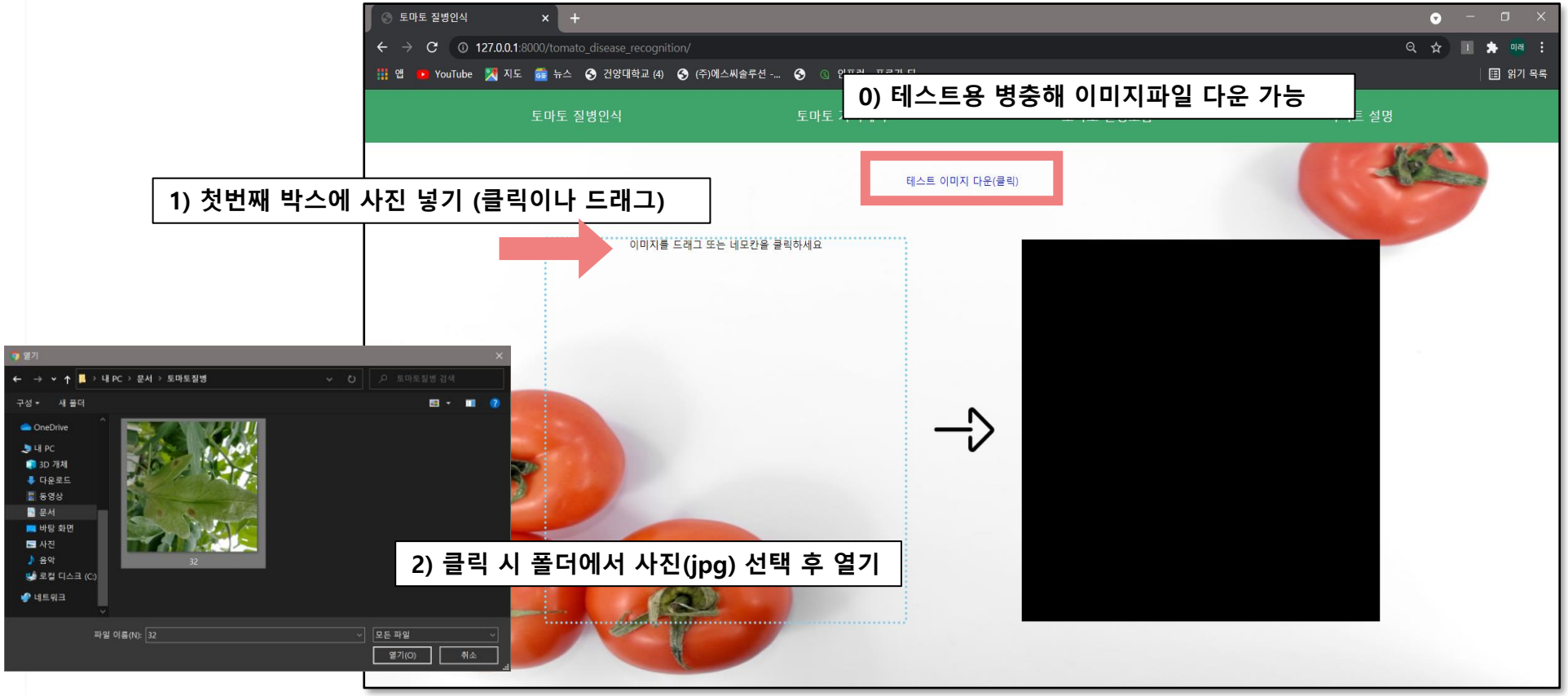


사이트의 기본적인 기능을 설명해주며
사용자가 사이트의 각 메뉴를 활용하는데 도움을 준다.

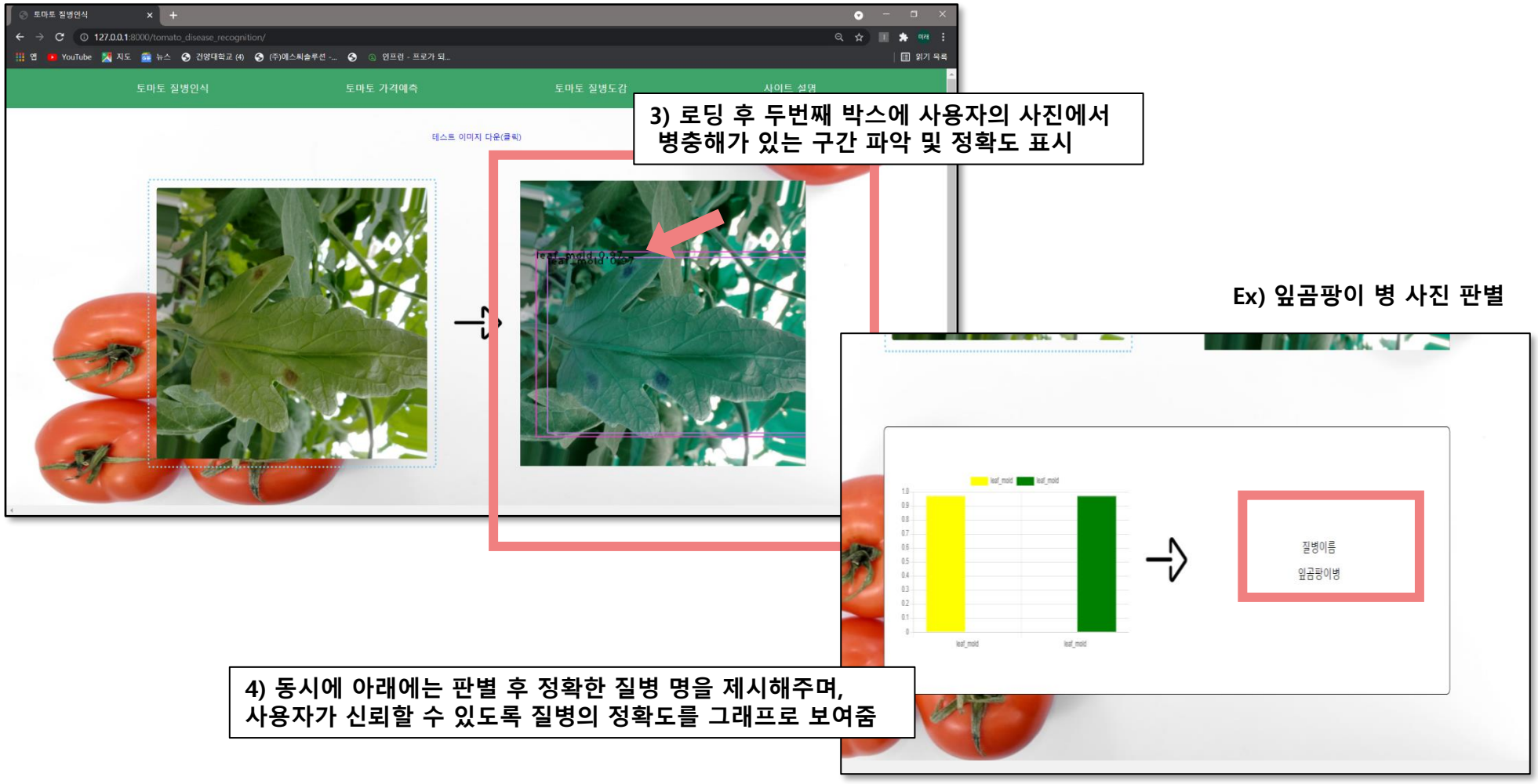
<모바일(작은 화면) ver.>

<토마토 질병인식 화면>

사용자의 파일을 업로드 하면,
사용자의 잎 사진에서 어떤 병충해가 있는지 기존에 훈련된 AI를 통해 판단하여
사진과 그래프로 나타내 사용자에게 알려준다.



<토마토 질병인식 화면>



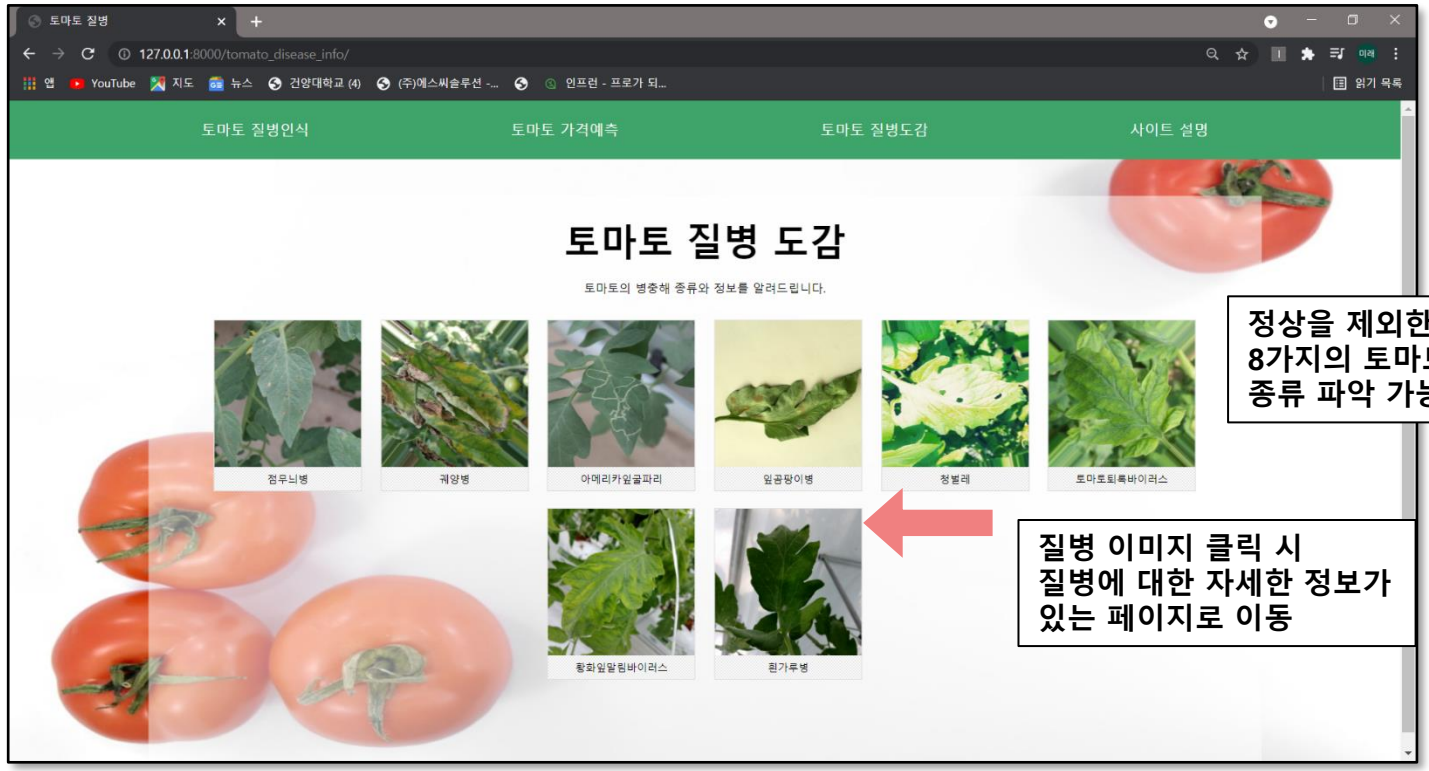
3) 로딩 후 두번째 박스에 사용자의 사진에서 병충해가 있는 구간 파악 및 정확도 표시

Ex) 잎곰팡이 병 사진 판별

4) 동시에 아래에는 판별 후 정확한 질병 명을 제시해주며, 사용자가 신뢰할 수 있도록 질병의 정확도를 그래프로 보여줌

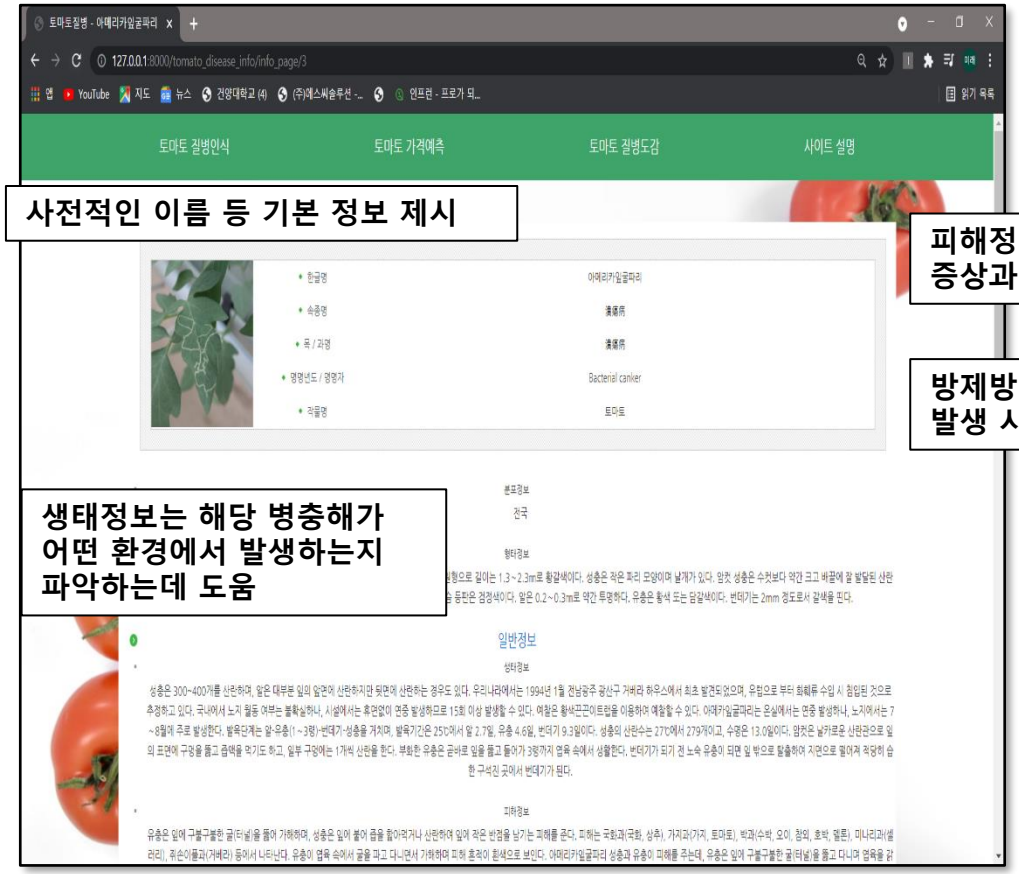
<토마토 질병 도감 화면>

사용자가 토마토 질병에는 어떤 종류가 있는지 확인이 가능하며
토마토 질병인식 후, 판별 받은 병충해에 대해 생태, 피해, 방제 등
다양한 정보를 알 수 있도록 제공한다.



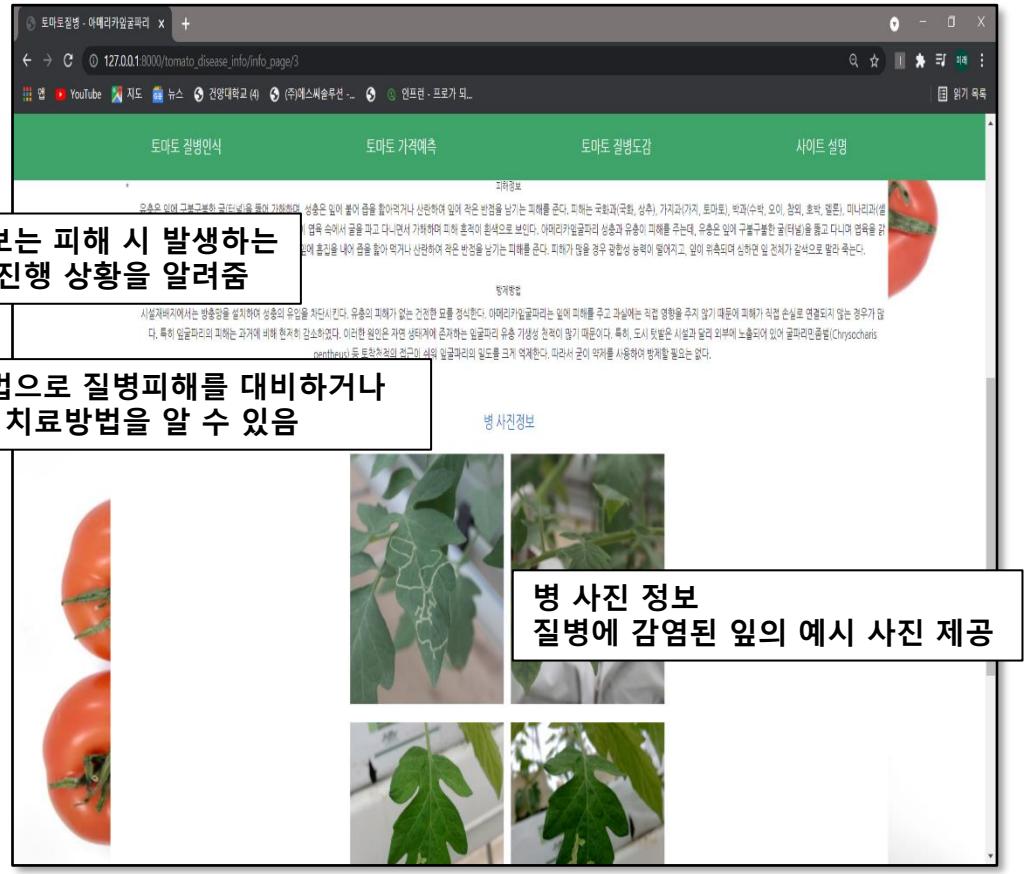
<토마토 질병 도감 화면>

Ex) 아메리카 잎굴파리 질병 정보 페이지



사전적인 이름 등 기본 정보 제시

생태정보는 해당 병충해가
어떤 환경에서 발생하는지
파악하는데 도움



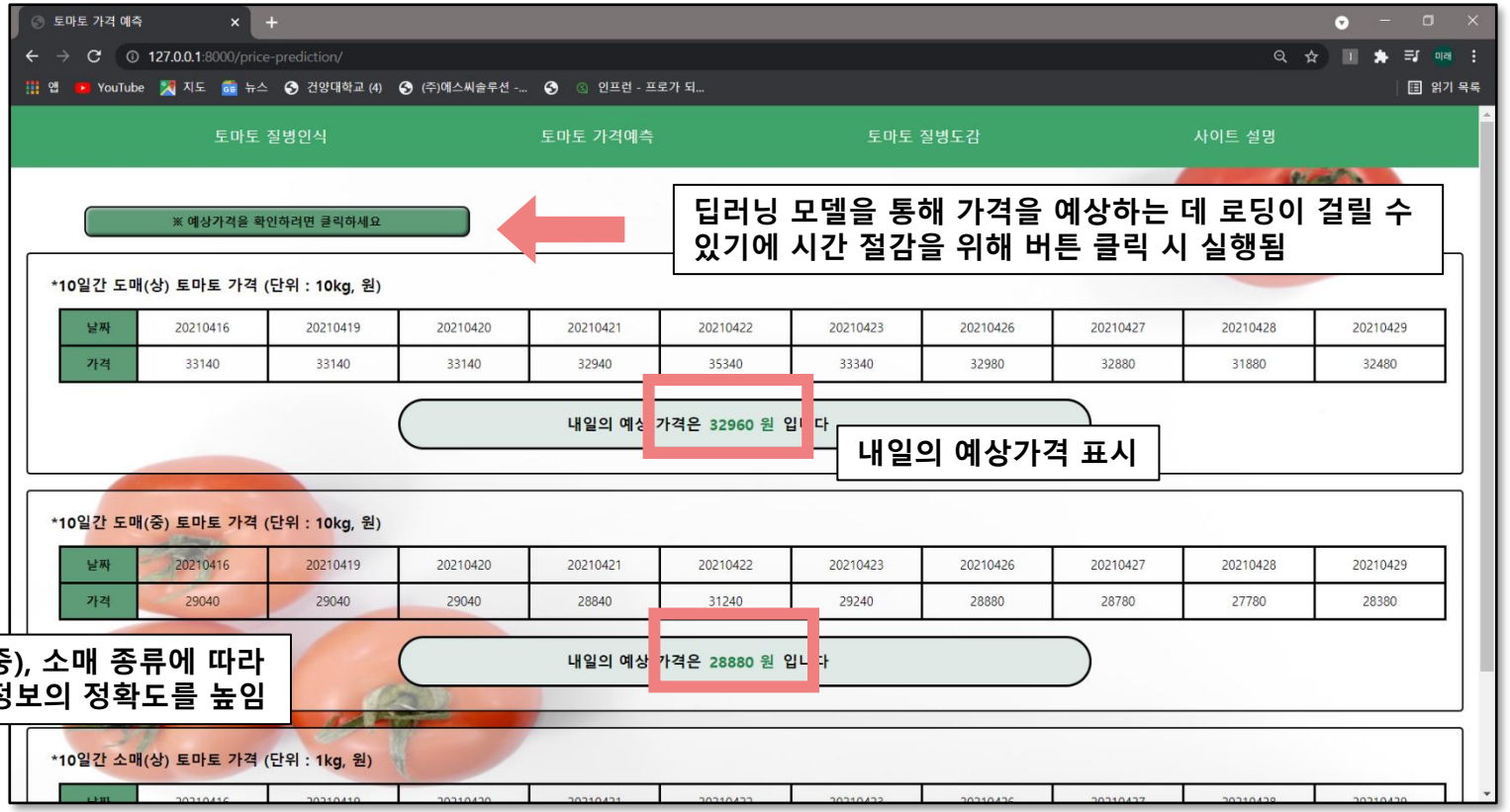
피해정보는 피해 시 발생하는
증상과 진행 상황을 알려줌

방제방법으로 질병피해를 대비하거나
발생 시 치료방법을 알 수 있음

병 사진 정보
질병에 감염된 잎의 예시 사진 제공

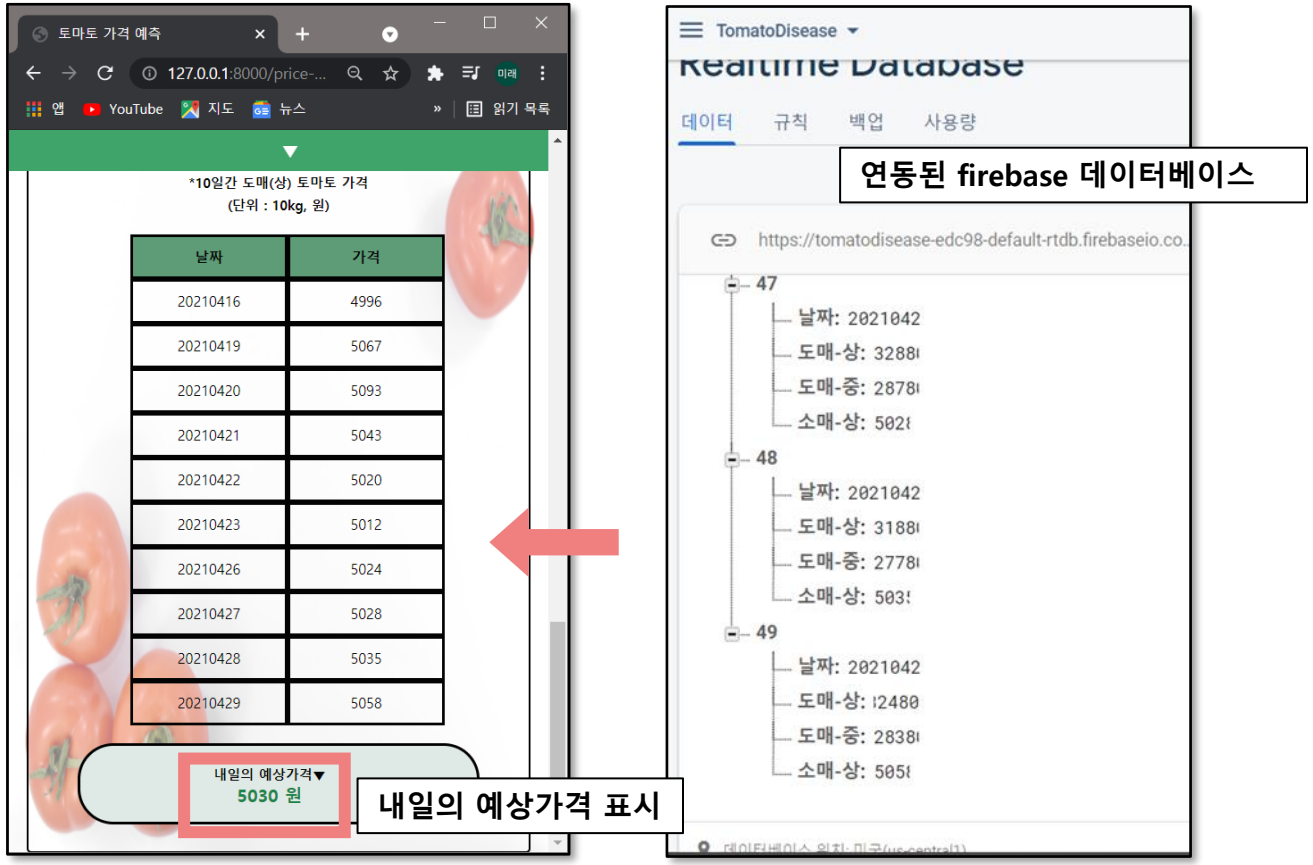
<토마토 가격 예측 화면>

최근 10일 간 토마토 가격의 시세를 알 수 있으며,
AI로 학습된 데이터를 이용해 10일간 데이터를 기준으로 내일의 가격을 예측한다.
도매, 소매, 품질별로 확인이 가능하다.



<토마토 가격 예측 화면>

Ex) 데이터베이스에 4/29일까지 업로드 된 경우 4/30일 예측



연동된 firebase 데이터베이스

내일의 예상가격 표시

<모바일(작은 화면) ver.>

INDEX

06 기대효과

기대효과

토마토 상태분석관리 웹 어플리케이션

off

01

최근 대두되고 있는 스마트팜, 청년농업 또는 개인이 토마토 재배 시
다양한 정보를 얻고, 전염되는 병충해, 가격관리 등 **문제를 해결할 수 있도록 도움**을 준다.

02

토마토에 병충해가 발생했는지, 가격이 얼마가 될 것인지 **AI가 판단**하여
기존에 사람이 예상하던 것보다 **쉽고 정확하게 발생한 병충해 파악**이 가능하다.

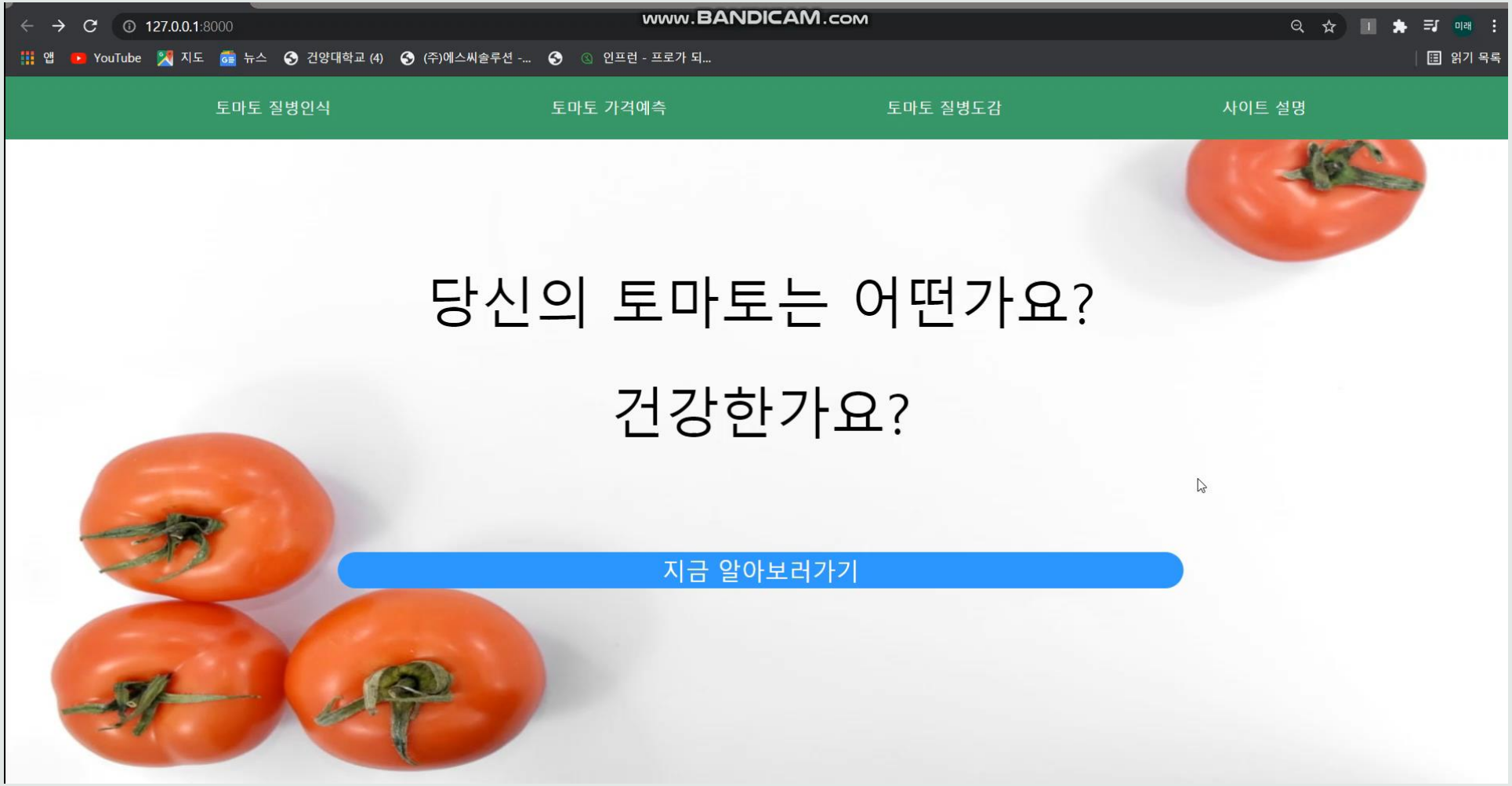
03

이후 **다른 농작물에도 같은 기술을 활용**하여
병충해 진단, 가격예측 등이 활발하게 진행될 수 있다.

프로젝트 시현

배포 예정

프로젝트 시현


127.0.0.1:8000
www.BANDICAM.com
앱
YouTube
지도
뉴스
건양대학교 (4)
(주)에스씨솔루션 -...
인프런 - 프로가 되...
읽기 목록
토마토 질병인식
토마토 가격예측
토마토 질병도감
사이트 설명
당신의 토마토는 어떤가요?
건강한가요?
지금 알아보러가기

Q & A