22.07.01.(금)

오티

- 자신의 개인번호 재확인
- 명찰 출입증 배부
- 보안서약서 및 고용계약서 작성

꿀팁

- etriware.etri.re.kr : 통합관리 사이트 같은 느낌
  > 월급, 휴가 신청 여기서
  > 메일전송, 메신저, 게시판
- otp.etri.re.kr : 외부 접속 시 vpn 켜서 여기로
- 와이파이 : smart-data, 연구원번호, 비밀번호
- 식당 : 5번건물 융합기술연구센터(A)
- 비상약 : 2번 건물
- 월요일 출입 등록 안 되어 있으면 장일순박사님

연구주제

- 라즈베리파이에 센서(조도, 습도)를 연결해 데이터 수집
- 유니티 혹은 그 이외의 것으로 디지털 트윈 / 에이전트로 이용
- 선형회귀 혹은 그 이외의 인공지능 기법으로 예측 데이터 생성

- 마지막엔 논문도 써보자.
- 마지막 주에는 성과 발표를 하자.

+ OpenCV 이용해서 인물 프로파일링
+ 주차장에 있는 빈 자리 세어주기

라즈베리 파이 동작

- 유심에 라즈베리파이를 구워냈다.
- 여러 다양한 타입의 라즈베리파이 중 적절한 친구를 선택했다.
  > 디스플레이가 있으며, 카메라 연결이 가능한 것
- 해상도가 높아 글자가 보이지 않아 해상도 변경
- 한글이 출력되지 않아 한글 언어팩 설치 함
- 기호가 올바르게 출력되지 않아 언어설정 변경

센서연결 : 온습도센서

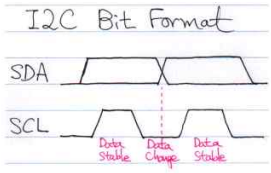
- 라즈베리 파이에 온습도 센서를 연결하여 동작하였음.

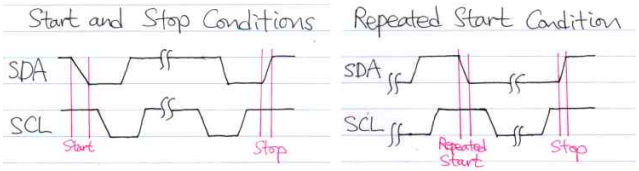
```
#!/usr/bin/python3

import Adafruit_DHT

sensor = Adafruit_DHT.DHT11

pin = 2

humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
if humidity is not None and temperature is not None:
    print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
else:
    print('Failed to get reading. Try again!')
```

센서연결 : 조도센서

- I2C 방식을 이용했으나, i2cdetect –y 1 결과 감지되지 않음
- i2c를 이용하지 않은 경우에는 읽어들인 값이 모두 0임
  > 저항을 변경해보았으나 계속 0값을 읽어들임

- **SCL, SDA 핀의 역할 : I2C 프로토콜**
  > SCL이 high일 때 SDA의 하강엣지 : 신호 시작
  > SCL이 high일 때 SDA의 상승엣지 : 신호 끝





22.07.04.(월)

Github Oranigation 생성

- etri-summer-Intern 조직 생성
- document repository에 문서 공유
- 향후 project repository를 생성해 작업 예정

보안 점검

- IoT-Test 와이파이가 점검으로 인해 가동 중단.
- smart-data만 와이파이 사용해야 하기 때문에 개발 중단.
- 마일스톤 작성하며 계획 수립

마일스톤

* Agent 서버를 노트북이 가능한가?
* 노트북으로 가능한 범위?
* 논문리뷰 함께 부탁드리기
* 프린트를 사용해도 되는가?

아이디어 보충 : 1안 에어컨/제습기 제어

- **온/습도 데이터를 이용해 에어컨, 제습기 제어**
  > 너무 덥다 ==> 에어컨 강하게 가동
  > 너무 춥고 습하다 ==> 에어컨 가동 중단, 제습기 동작

- **LED 표현**
  > 제습기/가습기 각 LED 5개를 둠
  > 제습기 가동 세게 시 LED 5개 점등

아이디어 보충 : 2안 스마트팜

- **유니티로 모델링 구현**
- **습도가 낮아요**
  > 물 준다.

- **습도가 높아요**
  > 조명 킨다.

- **온도가 높아요**
  > 에어컨 가동한다.

데이터 전달법

- 소켓통신?
- 그 이외의 방법?

퍼지 이론

- 퍼지이론에 대해 학습한 내용을 "퍼지이론.md"로 정리함

22.07.05.(화)

점검

- 에트리 웨어에서 고정 ip 할당 가능
- 층마다 서브넷 다름
- etriware/자산관리/IP관리/신규IP신청
- LSTM?
- 오차 : 예측 데이터와 실제를 비교해보기
- 플러그
- 프린트 : http://129.254.85.100/

22.07.06.(수)

환경설정

- anaconda
- pytorch
- vscode c/c++ 컴파일 환경 설정

22.07.07.(목)

와이파이

- IoT-test : spyders22v, 연구원번호, 비밀번호

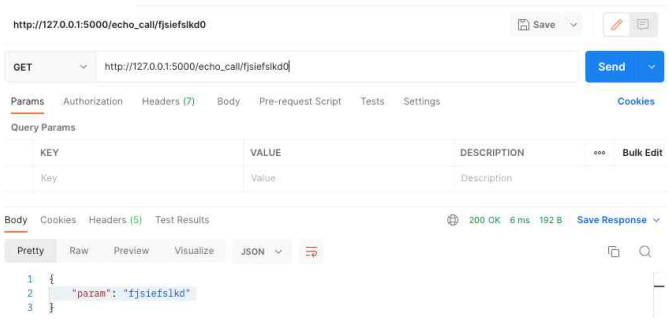
Flask 설정

```python
from flask import Flask, request, jsonify

# Flask 객체 인스턴트 생성
app = Flask(__name__)

# 접속 URL 설정
@app.route('/')
def index():
    return 'Hello, Index!'

@app.route('/home')
def home():
    return 'Hello, Home!'

@app.route('/user')
def user():
    return 'Hello, User!'

@app.route('/echo_call/<param>') #get echo api
def get_echo_call(param):
    return jsonify({"param": param})

```


```python
from flask import Flask, request, jsonify

# Flask 객체 인스턴트 생성
app = Flask(__name__)

# 접속 URL 설정
@app.route('/')
def index():
    return 'Hello, Index!'

@app.route('/home')
def home():
    return 'Hello, Home!'

@app.route('/echo_call/<param>') #get echo api
def get_echo_call(param):
    return jsonify({"param": param})

@app.route('/create', methods=['POST'])
def create():
    print(request.is_json)
    params = request.get_json()
    print(params['user_id'])
    print(params['user_name'])
    return 'ok'

if __name__ == '__main__':
    # 코드 수정 시 자동 반영
    app.run(debug=True)
```

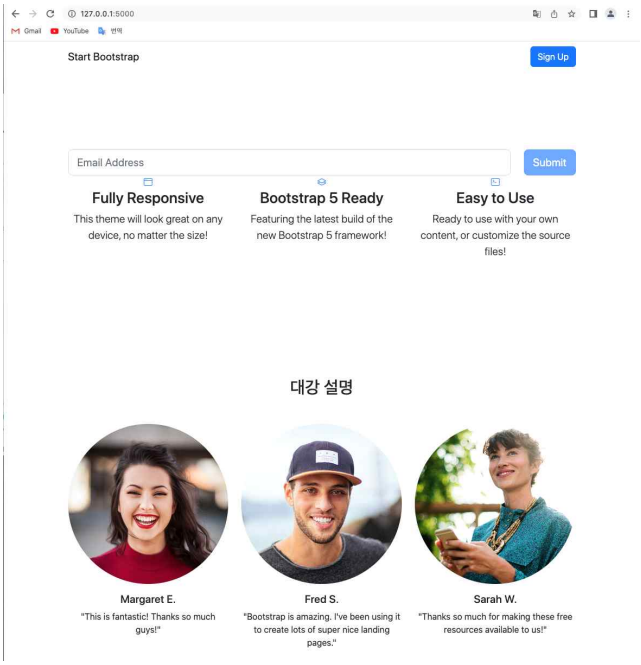

- visual studio에 python flask 동작 가능한 환경 설정
- 로컬 호스트에서의 웹 서버 가동 및 호스팅 가동

Postman을 이용한 테스트



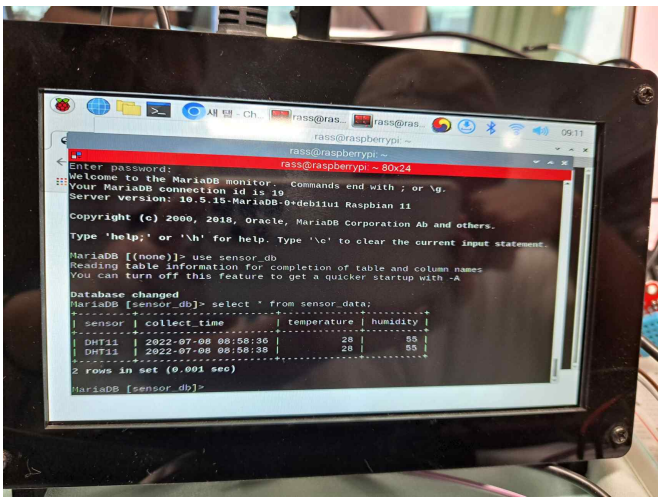
- 임의의 데이터 전송 시 받은 데이터를 그대로 되돌리는 정도
- json 데이터를 받아 처리하는 구조까지 완성됨

웹 페이지 연결



- 기본적인 부트스트랩 템플릿 적용

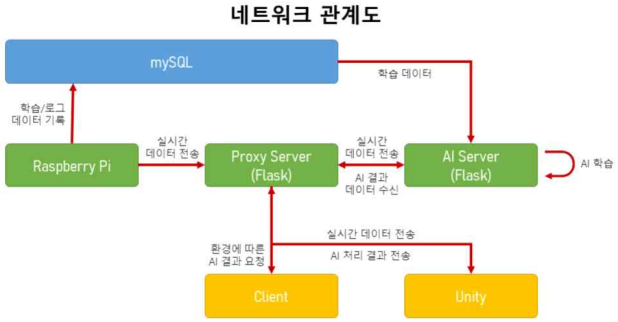
온습도 데이터 DB 저장



- 기준 시간 단위로 측정 후 기록

22.07.08.(금)

네트워크 관계도 정의



- **Raspberry Pi**
  > 센서 데이터를 mySQL에 학습/로그용으로 저장
  > 실시간으로 Flask로 구현된 Proxy Server에 전송

- **Proxy Server**
  > Flask로 구현됨
  > 라즈베리 파이에게 받은 데이터를 AI서버와 유니티에게 전송

- **AI Server**
  > Flask로 구현됨
  > Proxy Server를 통해 센서 데이터를 받아옴
  > 구동1) 제어 동작에 대해 반환해 줌
  > Proxy Server를 통해 시간 정보를 받아옴
  > 구동2) 온/습도 예측 데이터를 반환해 줌

- **Unity**
  > Proxy Server에게 현재 실시간 센서 데이터와 제어 동작 수신
  > 제어 동작에 따른 모델링 변화

- **Client**
  > Proxy Server에게 시간 정보 전달
  > Proxy Server에게 예측 데이터 수신

* AI Server의 구동2)와 Client에 대한 것은 구현 미정
* 구동1)과 Unity에 대해 우선적으로 구현

클라이언트 구현

```
<====== request ======>
Send Data :  {'temperature': 35, 'humidity': 35, 'illuminance': 34}

<====== response ======>
Sever State :  200
Response Data :  ok
```

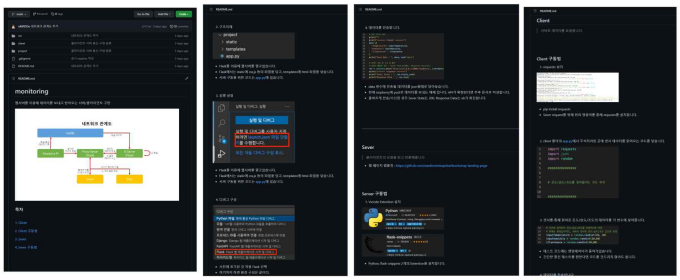
▲ Client의 Request/Response

```
True
수신 온도 :  35
수신 습도 :  35
수신 조도 :  34
```

▲ Proxy Server의 수신

- 클라이언트에서 Server로 송/수신하는 코드를 작성
- json 형태로 ID 속성을 정의하여 전송
- 수신이 올바르게 된 경우
  > 서버 상태 200
  > 서버가 반환한 "ok" 출력

Repository 분리 및 메뉴얼 정리



▲ 정리된 github 메뉴얼

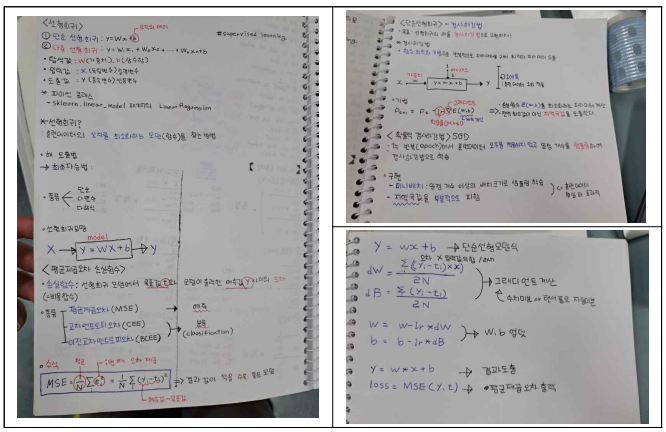
- 이번 서버/클라이언트 개발 내용을 monitoring 레포로 분리
- README를 통해 서버 구동 메뉴얼 작성

22.07.11.(월) ~ 07.13(수)

선형회귀론 학습

- 이전 "제어"를 위해 퍼지제어를 학습
- "예측"을 위해 선형회귀론 학습

선형회귀론 : 필기



- 개념적으로 기억할 부분은 필기로 작성

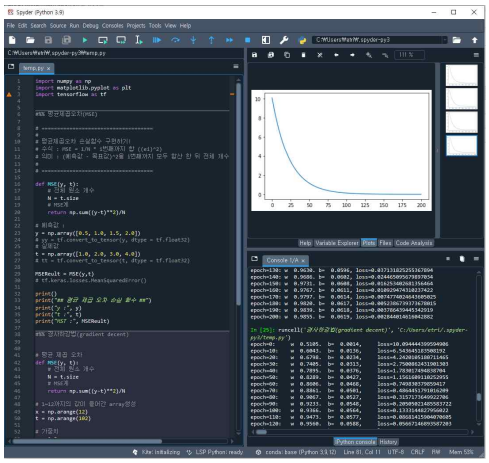
- 학습 내용

> 선형회귀에 대한 기본적 정의
> 선형회귀 종류 및 모델
> 도출한 값의 적정도를 알기 위한, 평균 제곱 오차 손실 함수
> 지역 최적값을 얻기 위한 최적화 기법, 경사하강법

* 도움이 되는 참고 자료

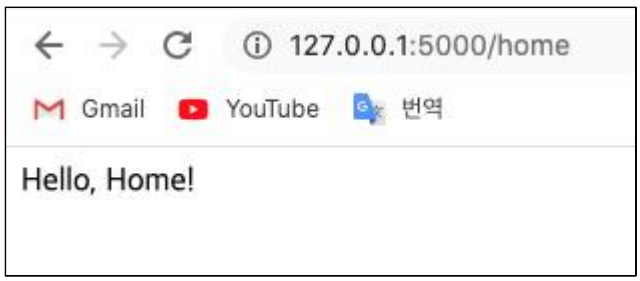
> 머신러닝 알고리즘 간의 비교
> 지도학습과 분류, 회귀, 예측
> 선형회귀모델
> sklearn의 KNN 기반 선형회귀
> 경사하강법의 시각적 구현 자료

선형회귀론 : 실습



- spyder를 이용한 실습
- 각 개념 이수 후 이해한 개념으로 코드 작성
- 이후 자료를 통한 실습 진행

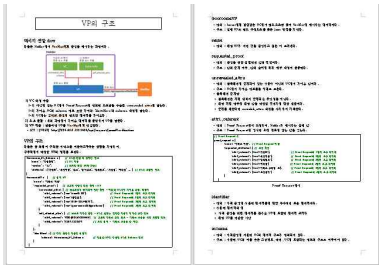
페이지 라우팅



- 페이지 라우팅에 대해 추가적으로 진행

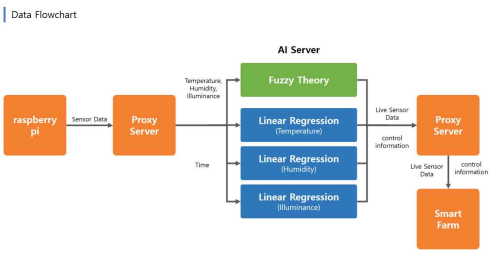
22.07.14.(금)

외부 문서 작업



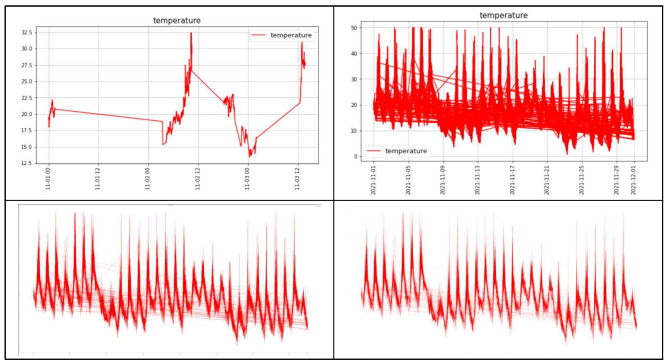
- 외부 업무가 급하게 있어 처리

데이터 흐름도 작성



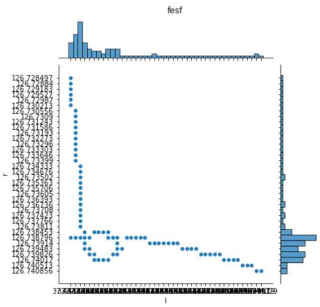
- 각 데이터가 어떻게 흘러갈지 재정립 하였다.
- AI구성.pptx

plot을 이용한 데이터 처리 학습



- plot 이미지와 데이터 양을 늘여가며 효율적인 데이터 표기 파악
- 시계열 데이터를 포맷팅하여 plot으로 표기할 수 있게 됨

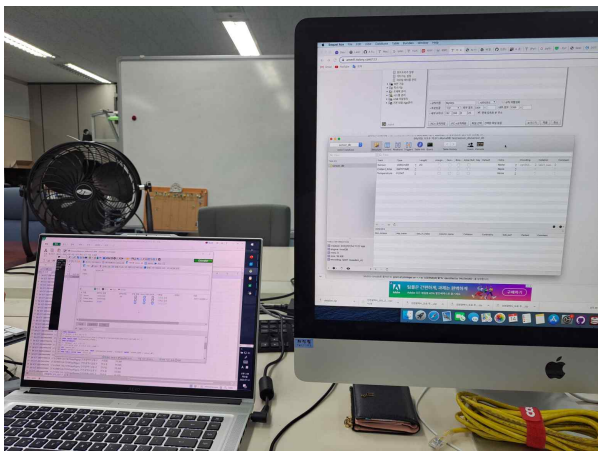
히스토그램과 산점도를 이용한 plot 표현



- 위도와 경도에 따른 온도를 산점도로 표기
- 각 지점에 대해선 히스토그램을 이용해 빈도를 표현

22.07.18.(금)

히스토그램과 산점도를 이용한 plot 표현



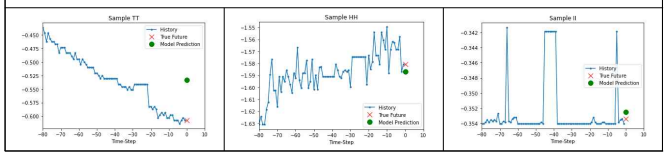
- 윈도우에서 mac에서 작성된 DB 열람 가능
- **용도** : local이 아닌 특정 ip로 데이터 전송을 위함
- **DB GUI 프로그램**
  > Window : heidiSQL
  > MacOS : sequel Ace

- **DB 설정**
  > name : root
  > host : 192.168.0.3
  > PW : 4321

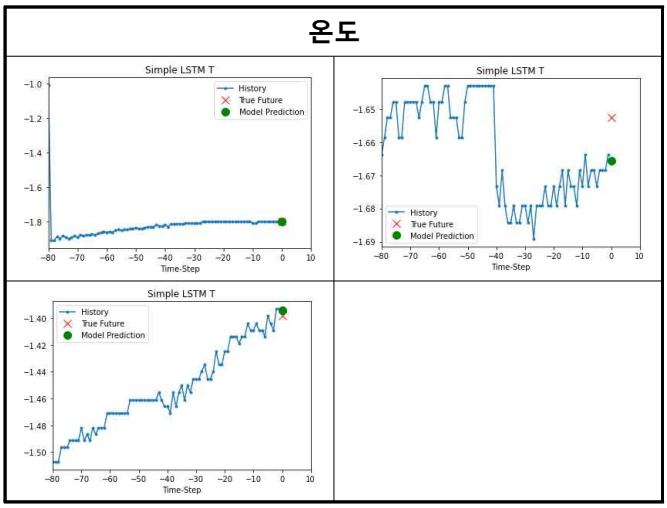
단순한 LSTM 모델을 이용한 예측

- 현재 추론하고 싶은 내용이 시계열 데이터임을 인지
- 시계열 데이터 예측에는 LSTM이 가장 효율적임을 알게 됨
- Keras의 LSTM 모델을 이용하여 학습

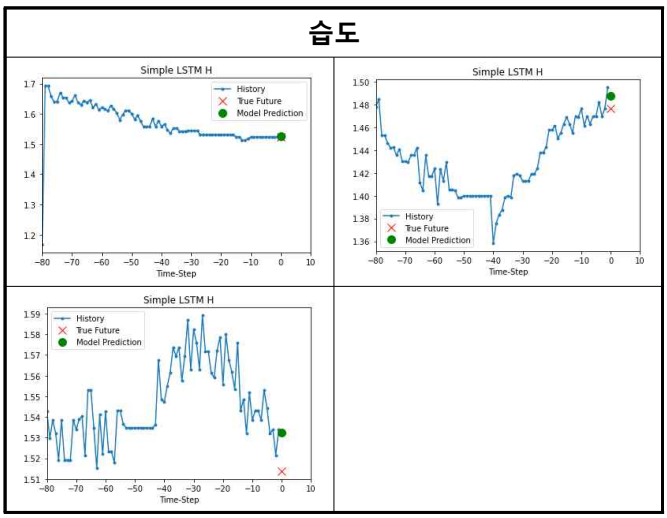
이전값의 20개 평균 값을 이용한 예측



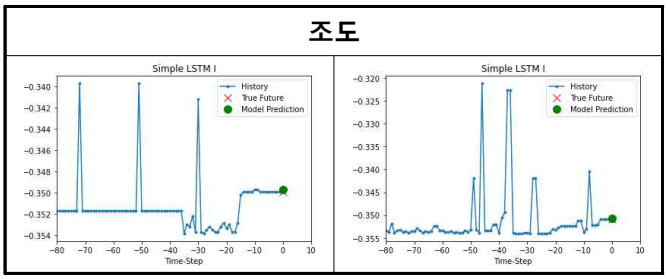
온도



습도



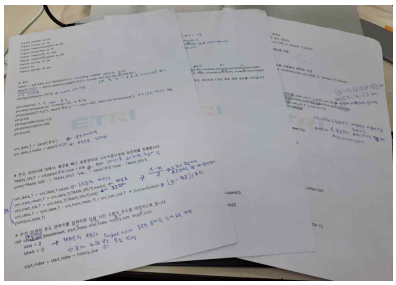
조도



- 튀는 값에 대한 예측을 잘 하지 못 하는 경향을 보였음.

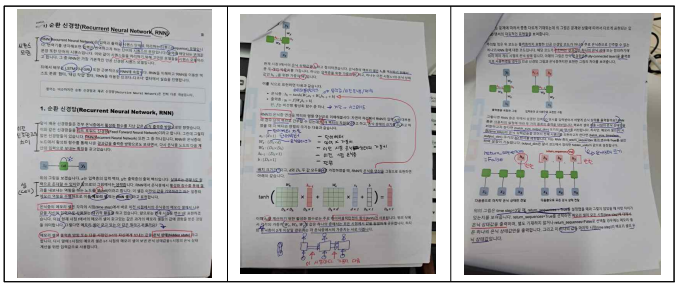
22.07.19.(화)

이전 코드 분석



- 이전 작성한 LSTM의 코드를 분석하여 의미 파악 시도한 흔적
- 기본적인 인공지능과 LSTM의 개념이 부족해 분석의 어려움 발생

RNN 개념학습

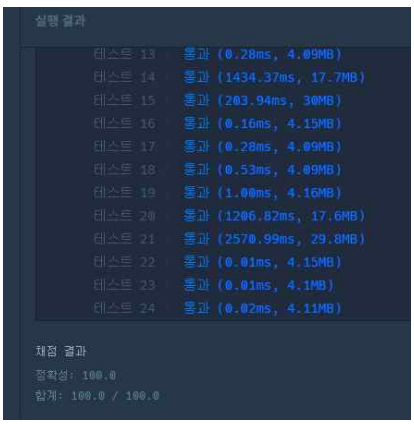


- 먼저 LSTM의 기초가 된다는 RNN에 대해 학습
- **학습내용**
  - > 기본적인 신경망의 구조
  - > 모델의 종류 및 활용방안
  - > RNN 은닉층 연산법과 각 명칭
  - > keras를 이용한 구현법과 함수 매개변수의 의미
  - > 양방향 순환신경망

* https://wikidocs.net/22886

22.07.20.(수)

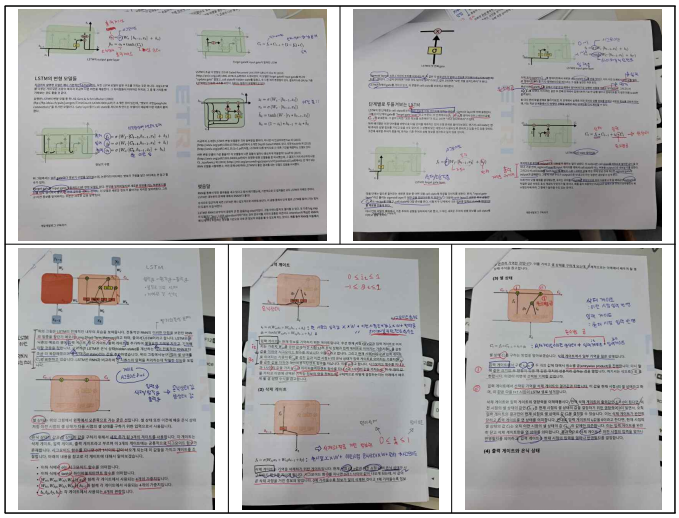
알고리즘 문제 풀이



- 집중이 안 되어, 머리가 잘 안 돌아가서인가 싶어 알고리즘 풀이
- 프로그래머스 알고리즘 문제 풀이
- 2022 KAKAO BLIND RECRUITMENT : 신고 결과 받기

* https://school.programmers.co.kr/learn/courses/30/lessons/92334

LSTM 개념 학습

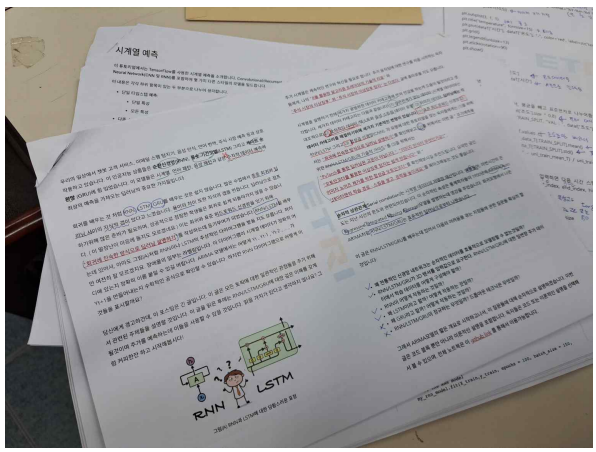


- RNN에 기반된 LSTM에 대해 학습
- **학습 내용**
  - > RNN에 비해 우수한 점
  - > 기본적인 구조
  - > 각 게이트의 역할
  - > 셀상태, 출력
  - > keras를 이용한 구현, 함수와 매개변수의 의미
- 도서관에 대여한 도서를 이용해 LSTM을 구현한 예제 확인

* https://dgkim5360.tistory.com/entry/understanding-long-short-term-memory-lstm-kr
* LSTM : https://wikidocs.net/22888
* keras의 LSTM : https://wikidocs.net/106473
* 도서 : 파이썬 라이브러리를 활용한 머신 러닝

22.07.21(목)

개념 학습



- 배운 개념에 대한 확신을 위한 학습
- 더 세부적인 내용을 다룬 자료를 읽어 보며 깊이를 줌

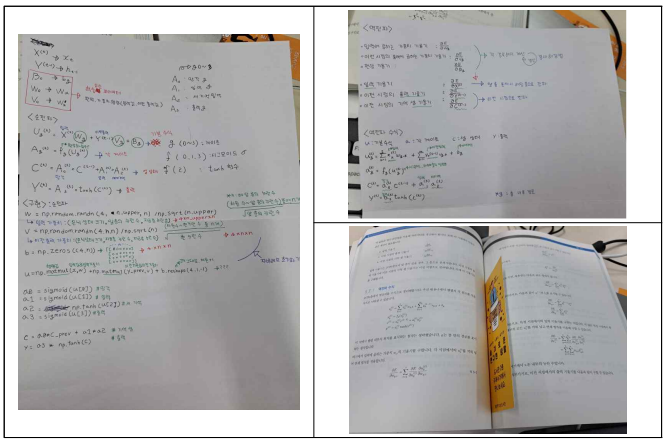
* RNN/LSTM/GUN : https://diane-space.tistory.com/331
* LSTM 구현 : https://colab.research.google.com/github/tensorflow/docs-l10n/blob/master/site/ko/tutorials/structured_data/time_series.ipynb?hl=ko

인공지능 모델 저장

- 저장은 되었는데 불러오기가 어려움

22.07.22(금), 07.26(화)

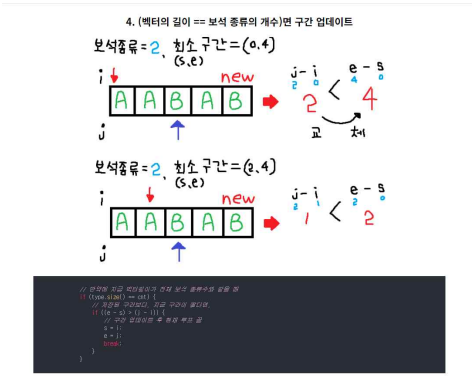
수식적 이해



- 개념적의 틀을 어느 정도 잡고 보니
- 수식적으로 알고리즘을 이용하여 하나하나 구현하고 싶어짐
- 도서관에가 책을 빌려와 수식적인 부분에 대해 공부
  => 이후 그러지 말고 가져다 쓰는 연습부터 하는 것을 추천받음

* 도서 : 핵심 딥 러닝 입문 : 최신 딥러닝 기술만 골라 배우는
* 월요일(07.25) 병가

알고리즘 풀이



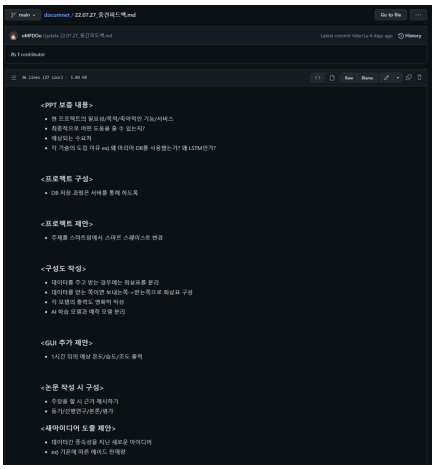
- 저번에 집중에 잘 먹혔던 알고리즘 풀이 재진행
- 문제가 너무 재미있어서 풀이 작성
- 2020 카카오 인턴십 보석 쇼핑
- https://school.programmers.co.kr/learn/courses/30/lessons/67258
- https://mfdo.tistory.com/37?category=1017638

22.07.27.(수)

박사님과 오전 담소

- 왜 대학원인가?
- 일단 연락을 넣어보자!
- 포트폴리오 한 장 보다는 세부 설명되어 있는 것으로 ...

중간 보고 및 피드백



- 중간 보고에 대한 피드백을 마크다운으로 작성해 업로드

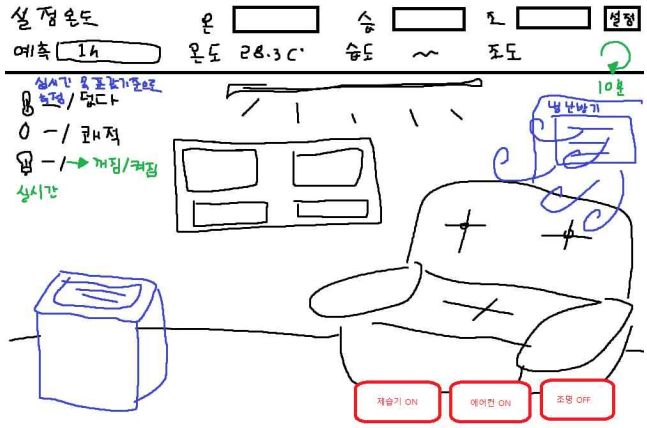
* 피드백 : document/22.07.27_중간피드백.md
* ppt : 하계 에트리 프로젝트 중간발표자료.pptx

22.07.28.(목)

주제 재정의

- 피드백을 기반으로 나아갈 방향을 다시 잡았다.

1) 조도가 너무 의미가 없다.
=> 불이 켜짐/꺼짐의 정보를 알리는데 이용
2) 주제가 스마트팜인데 스마트팜 데이터는 못 얻는다.
=> 스마트 공간으로 변경



▲ 구상도

<받아오는 거>
- 관측시간, 온도/습도/조도

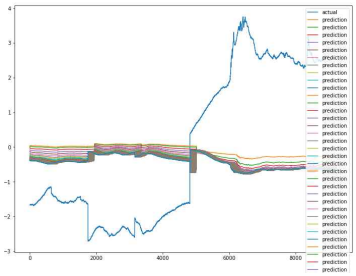
<받아와야하는 거>
- 예측 온/습/조
- 퍼지 제어값(실시간 데이터 기준)

<보내야 하는 거>
- 퍼지에게 : 희망값(온/습/조)
  > 얘가 기준이 되어서 낮다/높다 판단

Batch/Take/Repeat 개념

- 텐서플로우에서 사용하는 저 함수들의 의미가 궁금해져서 공부
- 여러 예제를 돌려보며 이해

그래프 그리기



- 기존의 하나의 예측값이 아닌 여러 예측값을 출력하고 싶었음
- 여러 괴상한 결과와 오류만 발생

22.07.29(금)

서버 API 추가 작성

1) 라즈베리파이 -> 서버 : 온도/습도/조도

```
<====== request ======>
Send Data :  {'temperature': 33.4, 'humidity': 37.3, 'illuminance': 27.8}

<====== response ======>
Sever State :  200
Response Data :  ok
```

```
True
수신 온도 :  33.4
수신 습도 :  37.3
수신 조도 :  27.8
192.168.0.3 - - [01/Aug/2022 11:15:26] "POST /raspberry HTTP/1.1" 200 -
```

2022-08-01 11:15:26	33.4	27.8	37.3	23

- 서버에게 실시간 데이터를 전송하는 API
- 피드백을 반영하여 **서버에서 DB로 데이터 저장**
  > 서버에서 이전에 연결한 DB로 데이터 Insert

* Flask와 python 세팅이 Aanaconda와 엉켜 애를 먹음

2) 서버 -> 유니티 : 온도/습도/조도

Collect_time	Temperature	Illuminance	Humidity	num
2022-07-29 17:35:48	37.1	32.4	58.2	17
2022-07-29 17:40:48	33.7	12.6	38.1	18
2022-07-29 17:45:48	37.6	40.2	31.3	19
2022-08-01 10:44:13	35.5	54.8	28.7	20
2022-08-01 10:49:13	34.1	17.3	54.4	21
2022-08-01 10:54:13	29.8	13	49.4	22
2022-08-01 11:15:26	33.4	27.8	37.3	23

```
전송 데이터 :  {'dataTime': datetime.datetime(2022, 8, 1, 11, 20, 26), 'temperature': 31.8
, 'humidity': 20.1, 'illuminance': 47.9}
192.168.0.3 - - [01/Aug/2022 11:20:56] "POST /getValue HTTP/1.1" 200 -
```

```
<======= response =======>
Sever State :  200
Response Data :  {
  "dataTime": "Mon, 01 Aug 2022 02:20:26 GMT",
  "humidity": 20.1,
  "illuminance": 47.9,
  "temperature": 31.8
}
```

- 서버에서 DB에 저장된 온도/습도/조도/측정시간 반환
- 각 고유번호를 기준으로 가장 최근 정보 반환

API 문서 작성

REST-API

서버로 데이터를 요청하거나 수신합니다.

설정된 REST API

1. (POST) http://192.168.0.2:5000/raspberry

- 용도 : 라즈베리 파이에서 수신한 센서 값을 전송합니다.
- 전송 데이터 타입 data ={ "temperature": 전송할 온도, "humidity": 전송할 습도, "illuminance" : 전송할 조도 }
- 수신 값 : "ok"

2. (POST) http://192.168.0.2:5000/getValue

- 용도 : 최신 온도, 습도, 조도를 반환합니다.
- 전송 데이터 형식 : "" >> 데이터 내용 없음
- 수신 값 :
  { "dataTime" : 수신 시간, "temperature": 현재 온도, "humidity": 현재 습도, "illuminance" : 현재 조도 }

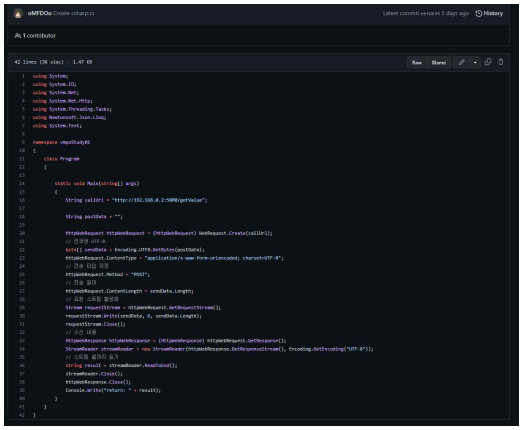
3. (POST) http://192.168.0.2:5000/getRandom

- 용도 : 랜덤 온도, 습도, 조도를 반환합니다.
- 전송 데이터 형식 : "" >> 데이터 내용 없음
- 수신 값 :
  { "temperature": 랜덤 온도, "humidity": 랜덤 습도, "illuminance" : 랜덤 조도 }

- 정의된 API의 주소/반환값/전송값 명시

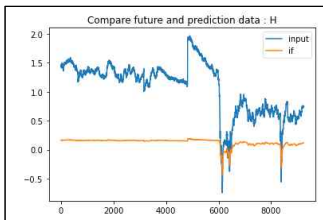
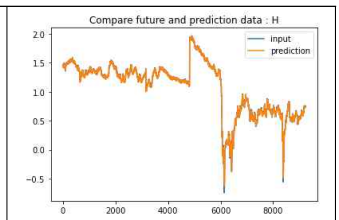
* README.md에 작성

C# 수신 코드 작성



- 유니티는 C#에서 작업하는데, 잘 안된다고 하셔서 작성
- POST 요청임에도 GET 요청을 하셔서 생긴 것이 이유였음

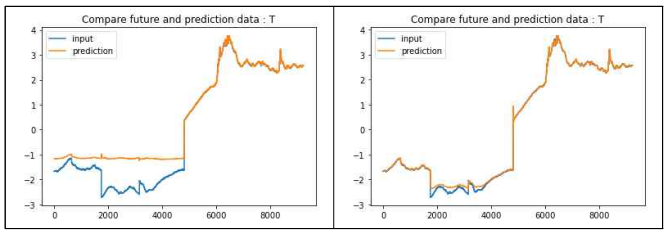
예측 그래프 출력

	
EVALUATION_INTERVAL = 3 EPOCHS = 5	EVALUATION_INTERVAL = 400 EPOCHS = 5

- 드디어 계속 끙끙앓던 그래프 출력 성공
- 이 날 너무 기뻤다.

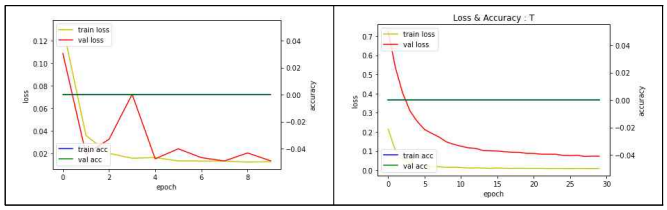
22.08.01.(월)

예측 그래프 버그 수정



- 온도 학습 시 절반만 학습한 듯한 모습을 보였다.
- train_univariate, val_univariate 값을 분리하지 않은 것이 원인
- 이전에 되었던 것이 신기했을 정도

Loss 그래프 출력



- Loss에 대해서는 그래프가 잘 표기된다.
- 하지만 Accuracy 값이 계속 0.0000e+00 값이 뜬다

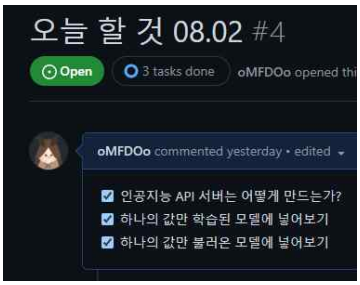
```
Epoch 1/5
400/400 [==============================] - 29s 67ms/step - loss: 0.1101 - acc: 0.0000e+00 - val_loss: 0.3643 - val_acc: 0.0000e+00
Epoch 2/5
400/400 [==============================] - 26s 66ms/step - loss: 0.0384 - acc: 0.0000e+00 - val_loss: 0.2339 - val_acc: 0.0000e+00
Epoch 3/5
400/400 [==============================] - 27s 67ms/step - loss: 0.0341 - acc: 0.0000e+00 - val_loss: 0.1637 - val_acc: 0.0000e+00
Epoch 4/5
400/400 [==============================] - 26s 65ms/step - loss: 0.0289 - acc: 0.0000e+00 - val_loss: 0.1372 - val_acc: 0.0000e+00
Epoch 5/5
400/400 [==============================] - 26s 66ms/step - loss: 0.0263 - acc: 0.0000e+00 - val_loss: 0.1237 - val_acc: 0.0000e+00
```

모델 불러오기 완료

- 모델을 '불러오기' 만 했다.
- API 형식으로 사용하기에는 아직 무리가 존재.

22.08.02.(화)

오늘 할 것!



인공지능 API는 어떻게 만들지?

- 자료를 통해 모델을 서버에서 불러와야 한다는 것을 알았다!
- 추가적으로 실제 데이터를 이용하기 위해 데이터를 변경했다.
  > 기상청 기상자료개방포털의 종관기상관측 자료
  > https://data.kma.go.kr/data/grnd/selectAsosRltmList.do?pgmNo=36

* https://niceman.tistory.com/192

학습된 값 모델에 넣기!

- "하나의 값"을 넣어서는 안 된다.
- 모델 설정 시 univariate pas history값을 200으로 설정했기 때문
  > 최근 200개의 데이터를 통해 유추한다는 의미
- 따라서 200개의 데이터를 랜덤으로 생성하였다.

- 모델 로드

> joblib이용

- 불러온 모델을 이용한 예측

1) 랜덤으로 데이터 생성
2) 3차원의 데이터로 reshape
3) model.predict()로 데이터를 집어넣는다!

22.08.03.(수)

오늘 할 것!

- csv파일 내용 DB에 옮기기
- DB에서 200개의 데이터 가져와서 담아내기
- DB에서 가져온 값 정규화하기
- DB에서 가져온 값 학습 모델에 입력하기

CSV 파일내용 DB에 저장하기

sensor_db.sensor_db: 35,441 행 (총) (대략적), 제한 수: 1,000 » 다음 모

Collect_time	Temperature	Illuminance	Humidity	num
2022-08-03 11:41:34	28.1	999.7	89	54
2022-08-03 11:41:34	27.7	999.8	91	55

- csvToDB API를 만들어서 CSV데이터를 서버로 전달했다.
- 서버는 전달받은 데이터를 DB에 전송한다.
- 약 3만 5000개의 데이터를 전달했다.

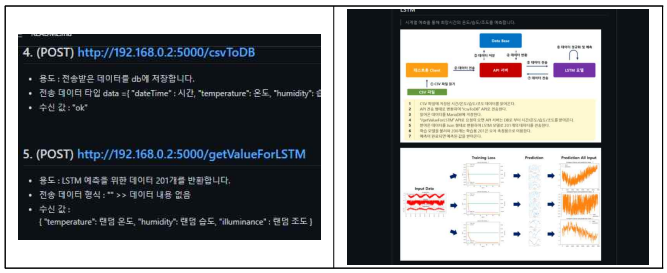
DB에서 가져온 데이터 학습

```
1/1 [==============================] - 0s 442ms/step
[[1.5564077]]
1/1 [==============================] - 0s 25ms/step
re :  [[30.461113]]
real :  31.9
```

```
In [61]: runcell(0, '/Users/etri/Desktop/monitoring/AI/loadModel')
1/1 [==============================] - 0s 400ms/step
1/1 [==============================] - 0s 400ms/step
1/1 [==============================] - 0s 404ms/step
Type          Prediction    Real Value      Loss
-----------  ------------  ------------  --------
temperature      30.2794          31.9     1.62064
humidity        115.783         1004.1   888.317
illuminance    1002.35            67      48.7827
```

- 가져온 데이터는 학습 시 사용한 데이터로 정규화했다
- 정규화한 데이터를 모델에 넣었다.

Readme 수정

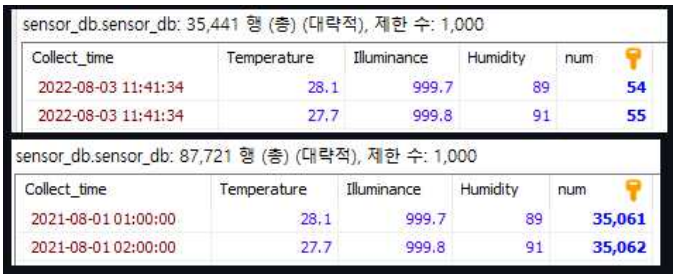


- REST API 추가 정의 및 LSTM 설명

22.08.04.(목)

오늘 할 것!

데이터 증량



sensor_db.sensor_db: 35,441 행 (총) (대략적), 제한 수: 1,000

Collect_time	Temperature	Illuminance	Humidity	num
2022-08-03 11:41:34	28.1	999.7	89	54
2022-08-03 11:41:34	27.7	999.8	91	55

sensor_db.sensor_db: 87,721 행 (총) (대략적), 제한 수: 1,000

Collect_time	Temperature	Illuminance	Humidity	num
2021-08-01 01:00:00	28.1	999.7	89	35,061
2021-08-01 02:00:00	27.7	999.8	91	35,062

- 기존 2018~2022년도의 데이터를 2012년까지 확대했다!

시간간격 데이터 받아오기

```
## 한 날짜의 200개 가져오기
SELECT * FROM sensor_db WHERE DATE(Collect_time) BETWEEN '2020-01-01' AND '2020-01-09';

# 1시간 간격으로 가져오기#
#SELECT * FROM sensor_db
#WHERE Collect_time BETWEEN TO_TIMESTAMP('2019-09-06 00:00:00', 'YYYY-MM-DD HH24:MI:SS') AND TO_TIMESTAMP('2019-09

# 1분 전 데이터부터
select Collect_time from sensor_db where Collect_time > DATE_SUB('2019-09-06 00:00:00',INTERVAL 1 HOUR) LIMIT 201;
select Collect_time from sensor_db where Collect_time > DATE_SUB(NOW(),INTERVAL 12 MONTH);
SELECT * from sensor_db where Collect_time > DATE_SUB('2019-09-06 00:00:00',INTERVAL 12 MONTH) LIMIT 201;


SELECT * from sensor_db where Collect_time = DATE_SUB('2019-09-06 00:00:00',INTERVAL 12 MONTH);
```

- DB에서 datetime을 기준으로 1시간, 3시간 같이 희망하는 시간의 데이터를 받아오는 쿼리를 짜고 싶었다
- 실제로는 입력시간의 1시간 전의 시간을 가져온다.
- 알고리즘을 이용해 반복문으로 시간을 입력 interval 간격으로 쿼리를 생성해 던졌다.

희망 interval로 DB 데이터 가져오기

- **클라이언트**
  > 입력 : 연/월/일/시/분/초/불러올 간격
  > 목적지 : getIntervalValueForLSTM API

- **서버**
  > 입력시간으로부터 201개의 데이터를 DB에서 가져와 반환

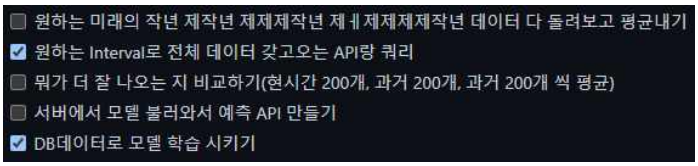
Input 데이터에 대한 고찰

- 다양한 input에 대해 생각을 정리해 문서화 하였다.
- 생각을 정리할 수 있어서 좋았다.

* document/08.04-데이터_input방식.pdf

22.08.05.(금)

오늘 할 것!



희망 간격으로 데이터 가져오기

```
Temperature Standardization ok
[-0.34097665 -0.61186178  0.6342098   1.41977666  1.14889154 -0.31388814
 -0.4493307  -0.69312731 -0.69312731 -0.8827469   0.28205914  1.20306856
  1.23015707  0.68838682 -0.17844558 -0.74730434 -1.12654351 -1.12654351
 -0.2326226   0.76965236  1.39268815  0.66129831 -0.31388814 -0.53059624
 -0.90983541 -1.12654351  0.52585575  1.60939625  1.36559964  1.20306856
  0.01117401 -0.61186178 -0.93692393 -1.12654351  0.6342098   2.01572394
  1.98863542  1.2843341   0.17370509 -0.50350773 -0.96401244 -1.12654351
  0.79674087  1.96154691  2.3678746   1.66357327  0.25497062 -0.09718004
 -0.53059624 -0.39515368  0.58003277  1.77192732  1.20306856  0.95927195
  0.22788211  0.01117401 -0.90983541 -1.099455   -0.39515368  0.22788211
 -1.04527798 -1.099455   -1.18072054 -1.3161631  -1.5328712  -1.5328712
 -0.85565839  0.52585575  1.12180302  0.14661657 -0.4493307  -0.63895029
 -1.26198608 -1.28907459 -0.85565839  0.03826252  0.09243955 -0.36806516
 -0.72021583 -1.07236649 -1.50578269 -1.42451715 -0.53059624  0.49876724
  0.22788211 -0.20553409 -1.20780905 -1.3161631  -1.64122525 -1.47869418
 -0.28679963  0.25497062  0.49876724 -0.25971111 -0.99110095 -0.99110095
 -1.12654351 -1.20780905 -0.6660388  -0.50350773]
meanH :  75.875
stdH  :  14.701509276261401
```

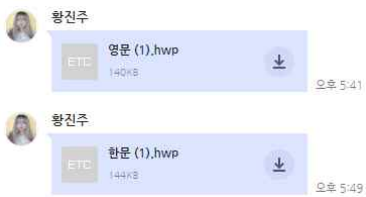
- 희망하는 간격으로 전체 데이터 가져오는 API와 쿼리 완성
- getAllIntervalValue : 희망 간격으로 모든 데이터 가져오기

DB 데이터로 학습

- 막판에 성공하고 집갔당 그래서 사진이 없다

22.08.08.(월)

학교 일 했었다



- indy 8,9,10 문서 번역

22.08.09.(화)

학교 일 했었다