Project Report

UL Korea internship 2020

SAR team Jeong Young Hwan

EMC-RF team Park So Bin

Internship Technical Report

기술 레포트 및 보고서

인사말

안녕하십니까 유엘코리아 인턴사원 정영환 박소빈 입니다.
이렇게 지면으로 인사드리는것은 처음입니다.
5월이 마무리 되는 이시점에서 잠시 멈춰서 지난 시간을 돌아봅니다.
6개월의 시간은 저희 생각보다 빨리 흘러가 마음이 미묘합니다.

새로운 환경에서 경력의 시발점을 찍는 저희 인턴 사원은 많은것이 서툴고 어리버리했을겁니다. 분명 저희가 다소 귀찮고 불편하셨을 것이라 사료됩니다만 그럼에도 불구하고 많은 분들의 관심과 도움 덕에 이렇게 무사히 인턴십을 마무리 하는 시간에 이르렀습니다. 지면을 빌어 이렇게 감사 드립니다.

이 문서에서는 기술레포트이자 인턴십을 마무리 짓는 결과 보고서 이며 그에 덧붙여 저희가 지난 시간동안 공부 하고 학습하며 여러 사원분들께 묻고 이해하고 알아갔던 내용과 회사생활을 담은 작은 소감문이라고 생각하셔도 좋습니다.

저희를 믿고 이렇게 결과물을 만들수 있게 도와주신 채경진 상무님을 비롯해 최창영 부장님 이준환부장님 김성훈과장님 윤현식대리님 양은희 사원님 송혜림 사원님께 다시금 감사의 인사를 드리며 글을 시작하겠습니다.

저희를 향한 신뢰에 다시금 감사드립니다.

—인턴 사원 정영환 박소빈 올림—

INDEX

1. What is Software?

2. Software Develop Procedure

3. Technical Boundary

4. Introduce Tech

5. Project 1 : SAR Team

   Auto Power Measurement Program

6. Project 2: SAR Team

   CA Combination Auto generate Program(ver.1,2)

7. Project 3: G-EMC Team

   Task Scheduler & Report Auto Program

8. Reference

1 What is Software?

우리는 걸을때 우리의 걸음을 계산한적이 없다.
마치 다리가 자율적으로 움직여 주는것 같은 착각을 일으킬 정도로 너무 자연스럽게 걷지만 신경학적으로 걸음을 분석하면 한걸음 걷는다는것이 얼마나 복잡하고 어려운 일인지 새삼 깨닫곤한다. 컴퓨터도 마찬가지다. 우리가 전원 버튼을 누르고 윈도우 화면이 뜨고 부팅이 되고 익스플로러를 눌러 실시간 검색어를 보는 동안 , 카카오톡에 연락이와 얼굴로 잠금해제를 하고 카톡을 보내는동안 , 디바이스 내부에서 어떤 일이 일어나는지 생각해 본적이 있는가 ?
뇌가 우리몸의 지휘권을 갖고 다리를 자연스레 움직여 준다면 컴퓨터도 마찬가지로 CPU에 탑재된 Software가 지휘권을 갖고 마우스 키보드 그 외 다른 운영체제와 같은 시스템 소프트 웨어들을 제어하고 정돈해주곤한다. 여기서 우리는 “제어” 와 “정돈”에 주목할 필요가 있다.

소프트웨어의 주 목적은 하드웨어와 다른 소프트웨어의 “제어”와 원활한 구동을 위한 “정리”에 있다는 말이다. 여기서 “정리”는 다른 프로그램이 실행하고 난 자리를 치워주고 새로운 프로그램이 착석하게 자리를 마련해주는것 뿐만 아니라 프로그램끼리 올바른 순서로 작동하게끔 줄을 세우는 정리도 포함된다.

위에 프로그램이라고 언급한 바 있다. 프로그램은 비가시 적인 코드들로 이루어진 명령어의 집합이다. 그리고 소프트 웨어는 그러한 프로그램을 가시적으로 활성화 하고 사용할수 있게 끔 “패키징”된 결과물을 말한다.

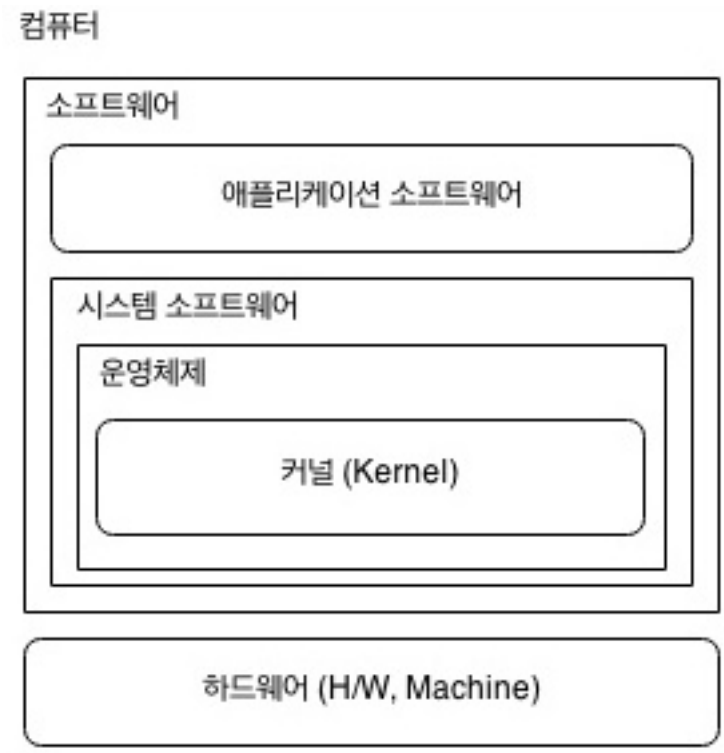
```python
mport cv2,dlib,openface
mport numpy as np
rom os import listdir
rom os.path import isdir, isfile, join

ef train(name):
    data_path = 'crop_face/' + name + '/'
    #파일만 리스트로 만듬
    face_pics = [f for f in listdir(data_path) if isfile(join(data_path,f))]

    Training_Data, Labels = [], []

    for i, files in enumerate(face_pics):
        image_path = data_path + face_pics[i]
        images = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
        # 이미지가 아니면 패스
        if images is None:
            continue
        Training_Data.append(np.asarray(images, dtype=np.uint8))
        Labels.append(i)
    if len(Labels) == 0:
        print("There is no data to train.")
        return None
```

이렇게 비 가시 적인 코드 들의 집합을 쉽게 사용성있게 재구성 한것을 소프트웨어라고 한다.



소프트웨어는 크게 두가지로 분류되는데 컴퓨터 시스템을 구동시키는데 필요한 "시스템소프트웨어" 와 구체적인 업무활동을 돕는 "응용소프트웨어"가 있다.

시스템 소프트 웨어는 컴퓨터시스템을 근본적으로 구동시키고 효율적으로 운용해주는 기능을 가진 프로그램 들의 모임이며 가장 대표적으로 운영체제{Operating System}가 있다. 다시 말해 시스템 소프트웨어는 응용프로그램이 원활하게 실행될수 있도록 길을 닦아주고 정리 해주는 안전요원과 도 같다고 할수 있다.

반면 응용소프트웨어는 컴퓨터를 이용하는 사람들의 구체적인 업무 수행을 도와주는 프로그램이다. 즉 운영체제 내에서 실행되는 모든 소프트 웨어를 말하며 특정 작업을 보다 편리하게 처리할 목적으로 만들어진 프로그램이다. 우리가 흔히 사용하는 MS Word Excel 과도 같은 것들이 바로 그렇다.

용어 정의

- 운영체제 :하드웨어와 사용자간 인터페이스
- 인터페이스 : 상호 소통을 위해 만들어진 물리적 매개체,프로토콜
- 프로토콜 : 정보를 주고 받는데 있어 통신적 규약과 약속
- 플랫폼 : 프로그램이 실행 될 수 있는 기반이 되는 시스템

현재 우리가 인식하는 컴퓨터의 발명 이전의 소프트 웨어의 최초 이론은 앨런튜링이 1935년 수필 [Computable numbers with an application to the Entscheidungs Problem]에서 제안된 결정문 제에서 비롯되었다. 이는 컴퓨터 과학과 소프트웨어 공학이 태동하는 계시로 작용해 결과론적으로는 둘다 모두 소프트 웨어 제작에 대해 연구하는 학문으로 발전했다.

Layers of Software

- 플랫폼 : 위의 용어설명에서도 언급한바 있지만 좀더 자세히 언급하면 Bios 혹은 펌웨어로 설명되는 장치 드라이버 , 운영체제 그리고 GUI등을 총체적으로 포함하는 단위 집합으로서 컴퓨터와 주변기기의 통신을 제어한다. 일반적으로 하드웨어와 패키지 형태로 제공되어 연산을 돕는 기능을 한다.

- User Develop Software : 사용자의 특화된 요구 기능을 수행하고 충족시키기 위함에 뿌리를 둔다. 일반적으로 위에 언급한 응용소프트 웨어를 변형해 유저 자신을 위한 새로운 커스텀 형태로 존재 하며 제어를 유저기반에 두기 때문에 특정한 형태가 없는것이 일반적이다.

Develop Software

소프트 웨어는 프로그래밍 언어와 관련된 유틸리티를 사용하여 제작되는데 일반적으로 스크립터 인터프리터와 같은 단일 프로그램 (compiler or linker)등을 꼽을수 있고 다른 도구를 포함하게 된다면 Integrated Develop Environment로 분류 가능하다.

Language

컴퓨팅리소스를 제어하는 도구로서 C언어 C++ c# 비주얼베이직 ,cobol, ADA,JAVA, Python 등이 있고 아래 제작된 프로그램들의 대부분은 python 으로 제작되었다.

2 Software Develop Procedure

소프트웨어 개발 방법론은 객체지향 CBD기반의 정보화 사업 개발 단계별 프로세스 및 산출물을 표준화 하여 실질적인 산업 개발시 구축 사업자와의 개발방법론 제안에 대한 긴장 완화 생산성 및 시스템의 품질 향상 , 개발 과정의 체계적 관리에 목적을 둔다.
실질적으로 정부에서는 관련 규정을 제정하여 국가 소프트웨어와 출판 소프트웨어에 대한 정보화 업무 규정 준수를 고려 하고 있다.

관련 규정

- 특허청 정보화 업무규정 (훈령 제 715조)
- 행안부 "정보시스템 구축운영지침 제 45조 , 행안부 제 2011-36조"

구성 개념

- 개발 방법론은 국제 표준 ISO/IEC 12207의 개발프로세스를 참조하여 산업 환경의 전반적인 흐름에 맞추어 변형한다.

개발의 방법론에 해당하기 때문에 Method + Knowledge 형태로 구성되어 소프트 개발에 필요한 반복적인 과정 을 체계적으로 정리한것에 해당한다.

요약하면 수많은 프로그램은 독립적이지 않고 각자 Dependency를 가지고 동작하기때문에 체계적으로 개발하는 과정을 정립해 소프트 웨어 개발의 분석 사례를 도식화 한것에 해당 한다.

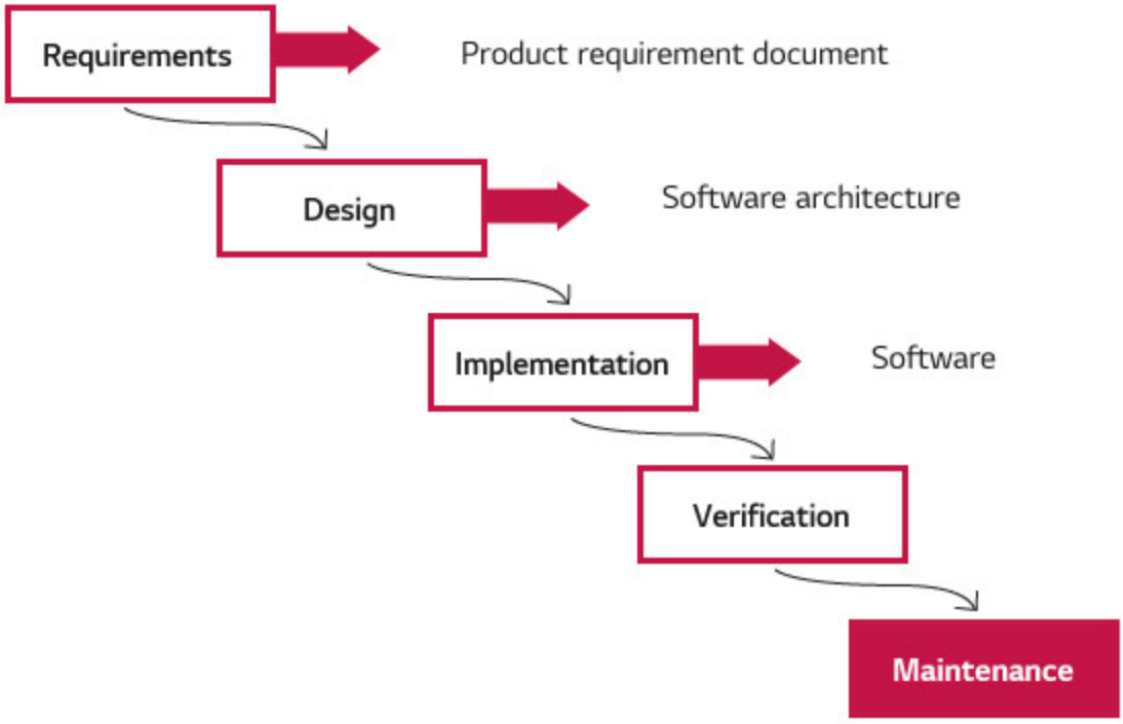
개발 방법론은 시간이 지나며 점차 수정되고 발전되는 과정을 거쳤다.

구조적 방법론

구조적 방법론은 절차 중심의 소프트웨어 개발 방법론으로 코드를 제한된 구조에서 생성하여 순차적으로 실행시키는 특징을 갖는다. 알고리즘 단위인 순차 , 선택,반복의 구조로 코드를 펴현하게 되므로 프로세스 단위로 문제를 해결하고 코드 보안을 유지 하며 유지 보수를 용이하게 만드는 장점이 있다.

일반적으로 폭포수 모델로 설명이 되는 개발 방법론적 접근에서 시스템에 대한 요구와 디자인이 이루어 지고 구조가 생성되고 나면 구현 과정에서 백트래킹이 일어나 진동하기도 한다.

그림으로 표현 하면 다음과 같다.



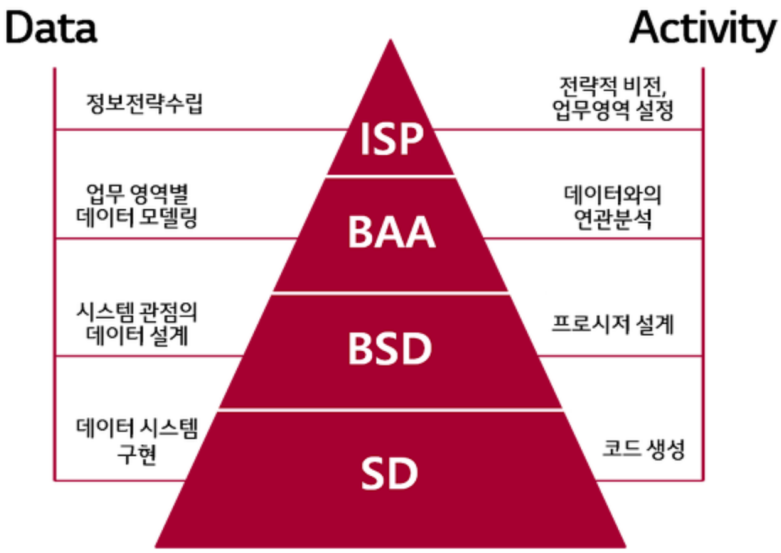
정형화된 분석절차에 따라 사용자 요구사항을 파악하여 문서화 하는 체계적인 분석 이론을 말한다. 프로그램의 로직을 중심으로 발전하였고 도형 중심의 분석이 주되며 자료 흐름도 기반으로 프로세스가 이루어 진다.

정보 공학적 방법론

1980 년대 태동한 방법론으로 정보 시스템이 단순 업무지원뿐아니라 경영 전략을 창출하는 MIS로 진화 하면서 회사의 경영 시스템을 수용하는 전략적 도구로 발전하면서 태동된 방법론이다.

요약하면 과거에는 컴퓨터가 계산기처럼 단순 업무 지원을 했다면 80년대에 들어서면서 컴퓨터를 이용해 사내 업무를 처리하게 되면서 프로그램에 회사의 이념이나 프로세스를 담을 필요가 생겼다는 말이다.

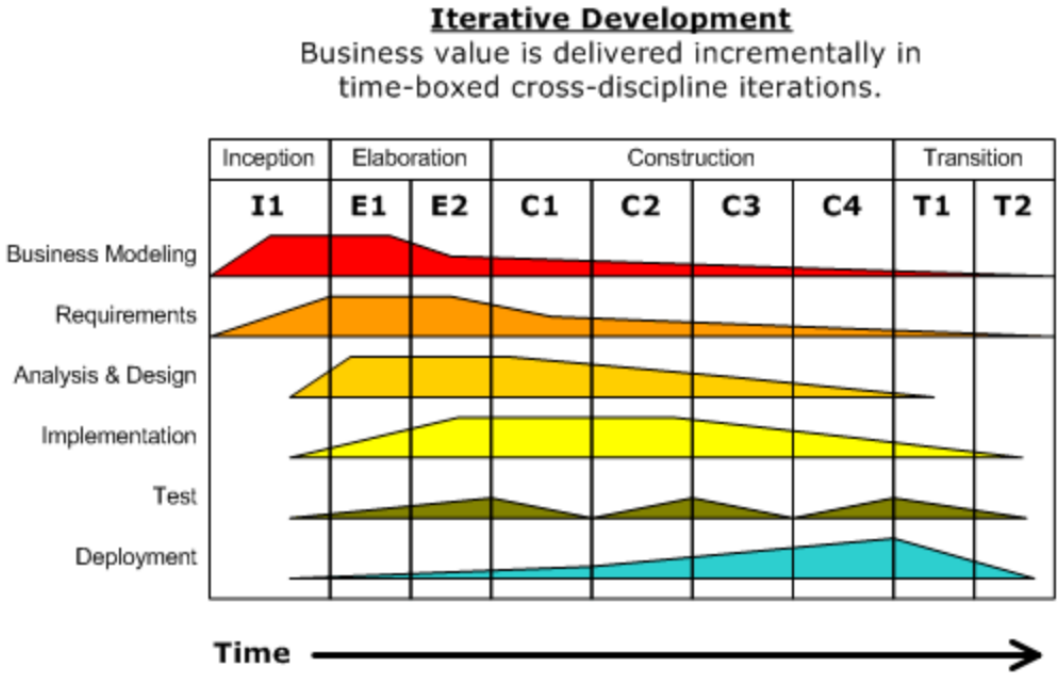
따라서 정보공학 방법론은 설계와 구현 단계에서 데이터를 우선적으로 개발하고 문제 영역을 세분화 하고 Top-Down 방식으로 진행되어 기존보다 빠른 결과물을 얻을수 있었다.



객체 지향 방법론

분석과 설계과정의 전단계를 데이터 중심으로 개발하는 방법론이다.

이 방법론은 개발 단계에서 반복과 점층적 모델을 사용하여 사용자의 요구사항을 반영하고 모든 단계를 유기적으로 협력시켜 전체 프로세스의 방향성을 유지하고 "재 사용성"을 높이는데 초점을 둔다.



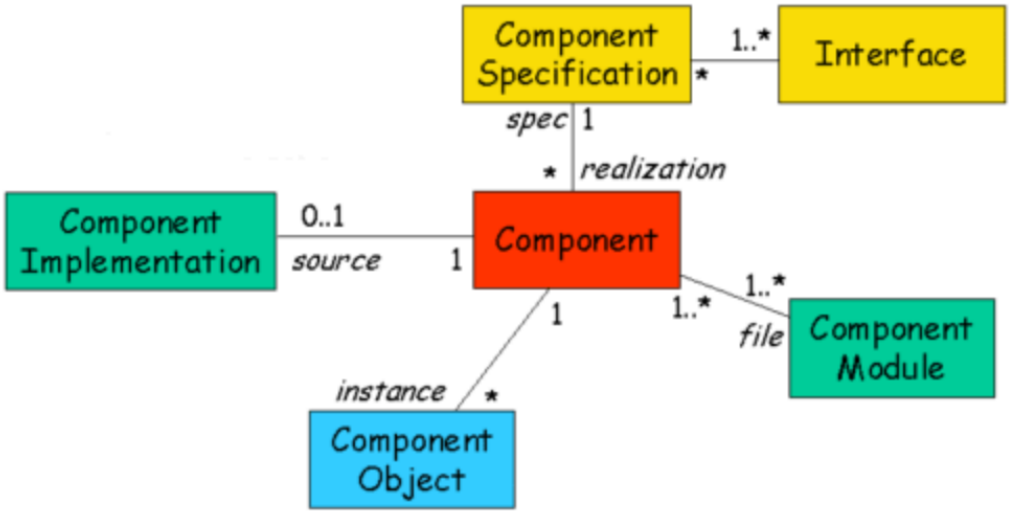
업무영역을 상호작용하는 객체들의 집단으로 이해 하고 시스템을 구축한다. 자료와 기능을 캡슐화 하고 객체간 상호작용을 해서 데이터를 공유한다.

서로의 공유는 Message 형태로 움직이게 되어 안전하고 안정되고 중요한 측면만 세부 모델링이 가능하여 데이터 분석의 초점이 명확하다.

CBD 분석 방법론

Component Based Development 분석방법론으로 문제를 조각으로 나누어 각각 컴포넌트를 구성한후 다시 조합하는 재 사용성에 초점을 둔 방식이다.

아무리 복잡한 시스템이라고 할지라도 단계별로 나누어 생각하게 되기 때문에 전체 시스템이나 프로그램에 영향을 주지 않고 빠르게 문제를 해결한다는 장점이 있다.



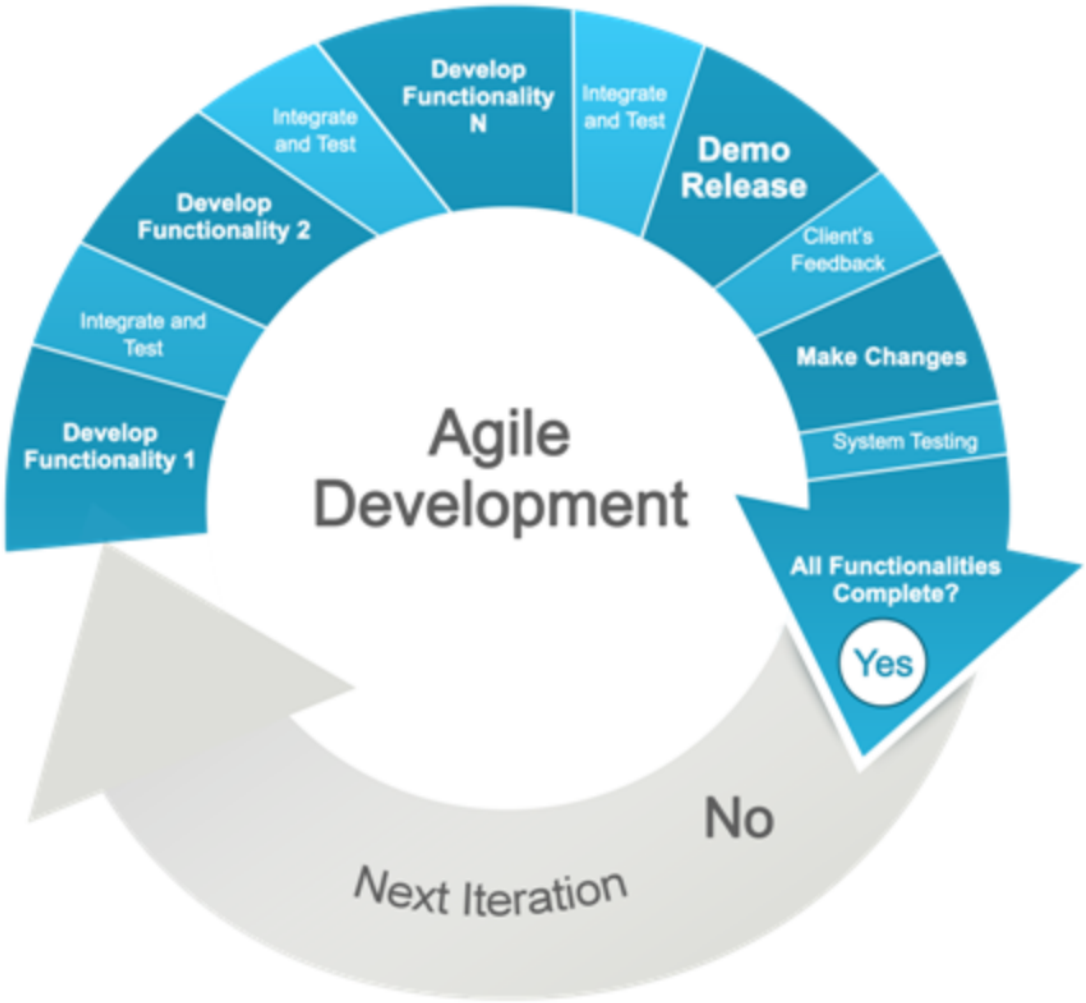
반복 점진적 개발프로세스를 제공하고 표준화 된 산출물을 작성하고 컴포넌트를 제작한다.

이는 유지보수를 용이하게 만들어 시스템에 소모되는 비용을 최소화 할 수 있다.

Agile 방법론

이전 방식들은 개발과정에서 과도한 문서나 형식적인 절차가 많다는 단점이 있었다.
이는 끊임 없는 비용의 누수를 발생시켰고 이를 해결하기 위해 에자일이라는 새로운 방법론이 대두되었다.

에자일 방법론은 고객과의 협력을 중시하고 프로세스나 도구에 국학되지 않는 자기 적응적 방식을 채택하는데 일정한 주기를 가지고 프로토타입을 만들어 내어 고객의 요구사항을 반영하기 쉽고 변화에 빠르게 대응할수 있다는 장점이 있다.



이하 프로젝트에서 진행되는 방법은 틀은 에자일을 유지 하되 일부 CBD를 차용해서 개발 시간을 줄이고자 노력했다.

반복적 개발 이슈나 사원분들의 추가적인 요구 사항을 한꺼번에 취합하기가 어렵고 개발의 과정에서 수차례 수정 요구사항이 있었다.

이는 기존의 코드 작업을 중단하고 처음부터 다시 시작해야 하는 경우도 있었고 사원 분들마다 다른 요구 사항을 내세우는 경우도 있어 통일되지 않은 결과를 가지고 머리를 싸매야 하는 경우도 있어서

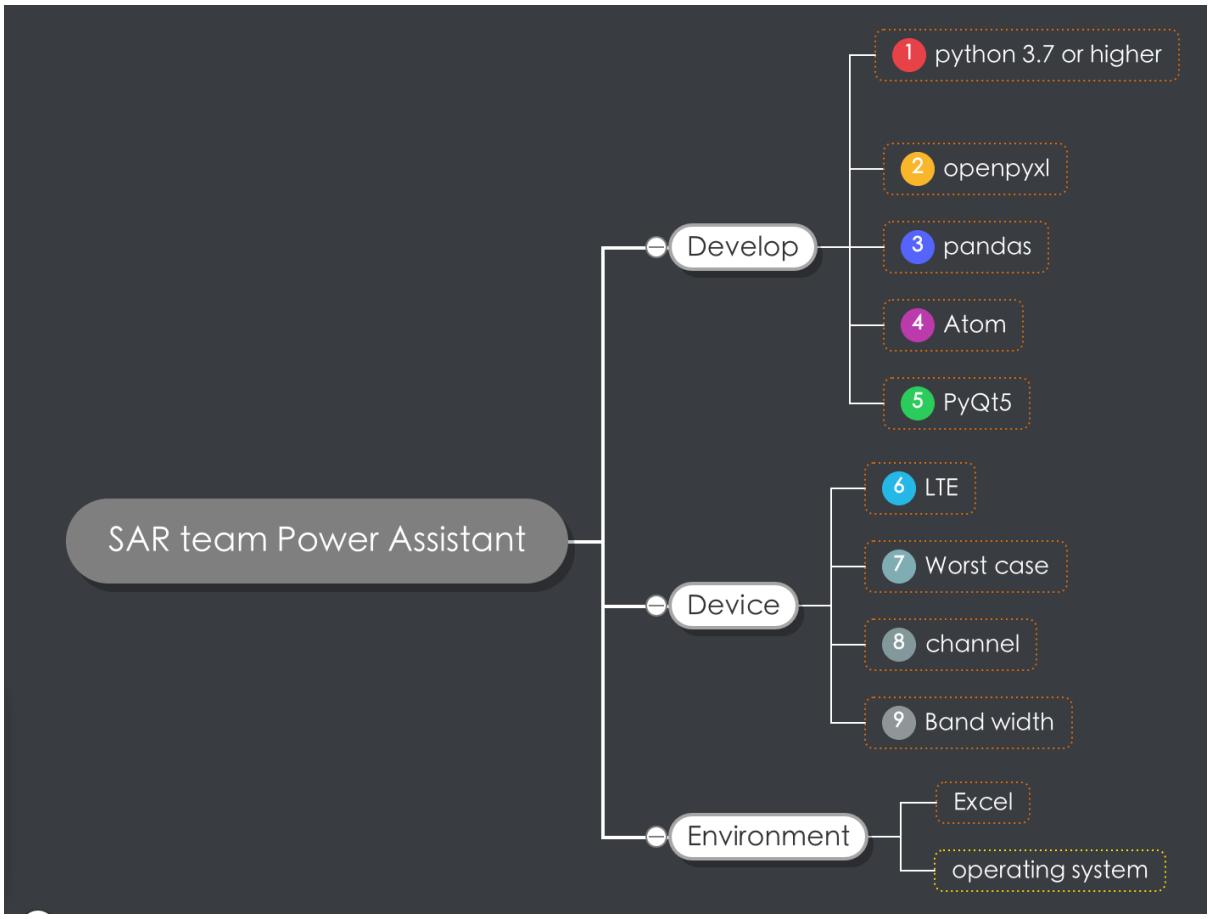
일괄적인 프로토 타입을 제작하고 사원분들을 방문해 결과를 보여드리고 피드백을 받는 형태로 진행했다.

피드백의 형태를 재 취합하고 코드를 수정하는 과정에서는 객체 지향적인 덩어리 기준의 코드 재활용성을 고려해서 수정사항이 많았음에도 불구하고 큰 시간을 소요 하지 않은것을 장점으로 꼽을수 있겠다.

3 Technical Boundary

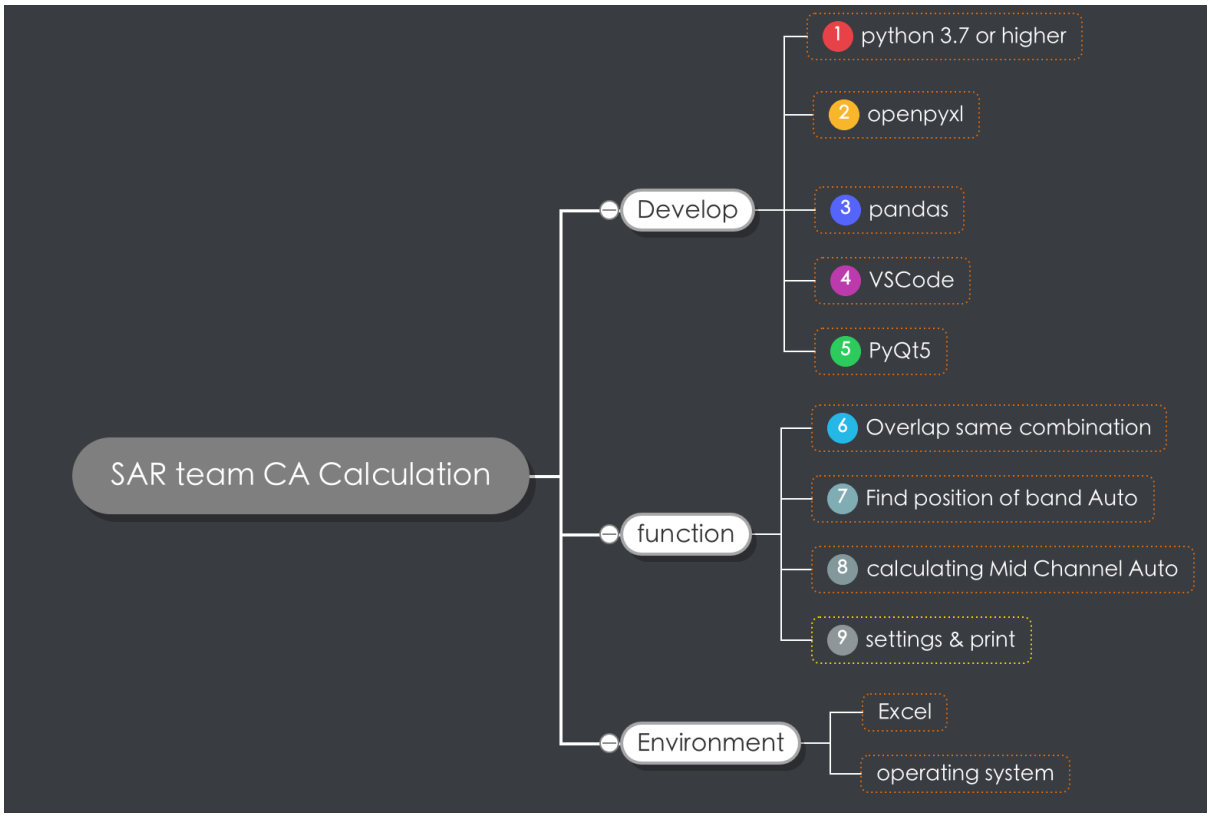
이하 내용에서는 프로젝트를 진행하며 사용된 기술적 내용을 열거 하고 해당 내용들이 어떤식으로 연결되어 프로그램과 구조화 되었는지 가시적으로 보여주고자 한다.

SAR Team Power Table Assistant



파이썬 환경에서 개발을 시작했고 측정 장비에서 출력되는 포멧 형태가 json 형식임을 인지했다. 따라서 일부 데이터는 pandas로 일부 데이터는 openpyxl 로 읽어 table 형태로 정리 시켰으며 PyQt5 를 활용해 GUI 디자인을 했다. 결과물은 excel 형태로 출력시켜 최대한 의존성을 해치지 않고자 했다. 가장 기본이 되는 프로젝트

SAR Team CA Calculation Assistant

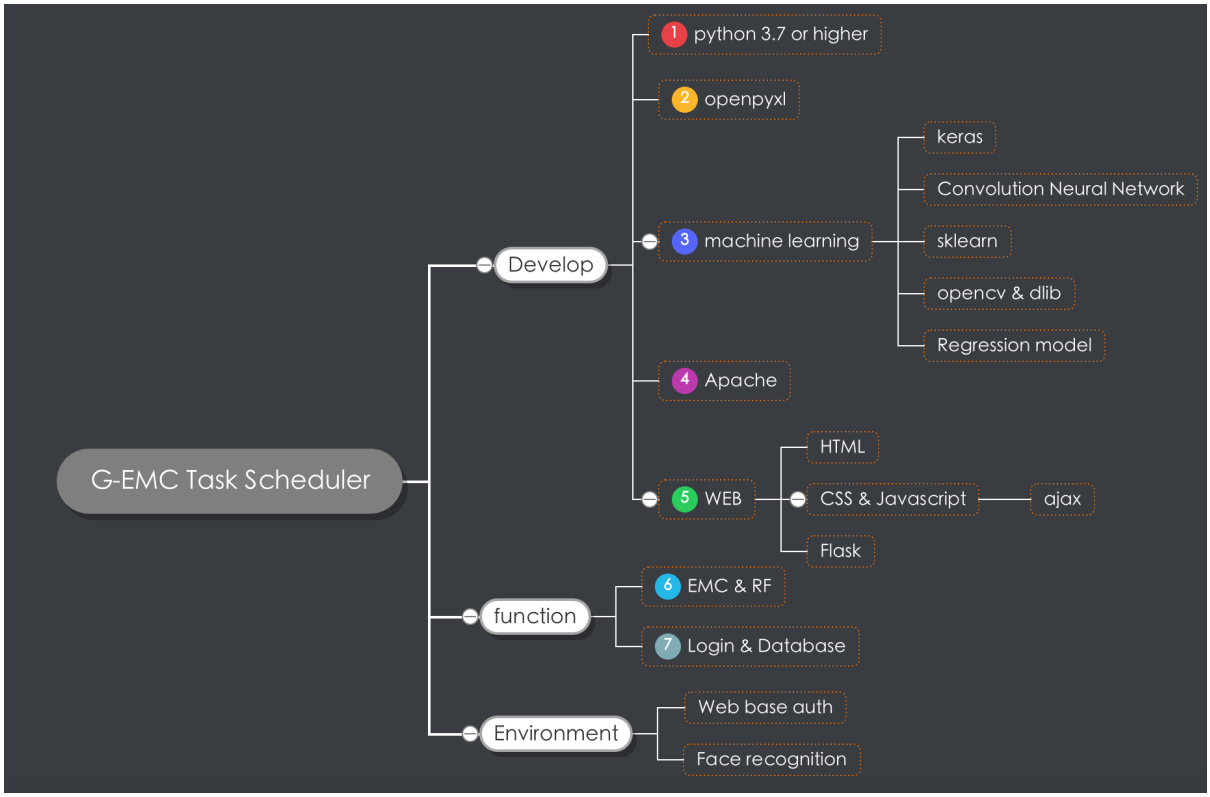


CA calculation에는 여러가지 알고리즘이 접목되어 더 많은 컴퓨팅 리소스를 발생시킨다. 정렬 알고리즘, 중복연산, 가장 먼 밴드 찾기, inter & intra 에 따른 케이스 분배, 출력 자동화, 최적화 문제, 환경 설정문제, 타 환경 작동 오류 문제 같은 요소들이 고려되었다.

마찬가지로 python 환경에서 작성되었고 openpyxl 을 통해서 제조사 리스트를 받아와 연산을 실행한다.

수작업으로 진행되던 이슈들을 단 몇분 내에 해결 할수 있어 사원분들 생산성 향상에 도움이 되고자 한다.

G-EMC Task Scheduler



웹 베이스 기반 사내 서버 구축과 기계학습 기반 생체인식 모델을 결합 시킨 업무 스케쥴러에 해당한다.

기능에는

- 성적서 자동화
- 팀 책임자 자동 메일 발송
- 전체 진행률 가시화
- 더미데이터를 통한 회귀 학습으로 시험 종료 일자 자동 계산
- 로그인시 얼굴인식을 통한 보안 강화
- 프로젝트 참여 인원 집계 및 업무 참여 확인기능

을 탑재해 사내 업무 스케쥴을 실시간으로 업데이트 하고 확인 할수 있게 설계 하였다. (진행중)

4 Introduce Tech

이 장에서는 해당 프로젝트에 포함된 기술과 그 용도 그리고 호환성 문제와 몇가지 알고리즘 을 소개 하고 깊지 않은 수준에서 어떤식으로 동작하는지 설명 하고자 한다.

실질적인 프로그램을 이루는 여러가지 요소들과 그 의존성에 대해서 서술 하겠다.

Python

1991년프로그램귀도반로섬이발표한고급프로그래밍 언어 로 플랫폼에 독립적이며 인터프리터식 객체지향적 동적 타이핑을 지원하는 대화형언 어이다. 파이썬은 비영리의 파이썬 소프트웨어 재단이 관리하는 개방 형,공동체 기반 개발모델을 가지고 있으며 C언어로 구현된 C파이썬이 표준으로 매김한다.

이러한 파이썬은 간단한 문법과 빠른 개발 환경을 제공해 성과물을 빠르게 생산해 낼수 있다는것이 장점이다.

Aug 2019	Aug 2018	Change	Programming Language	Ratings
1	1		Java	16.028%
2	2		C	15.154%
3	4	^	Python	10.020%

이처럼 파이썬은 세계 3대 언어에 선정되어 C ,JAVA와 함께 주위의 많은 소프트 웨어에 이식되어 있다.

추가적으로 python은 해외에서 가장 유명한 오픈 프로그래머 커뮤니티 STACK OVERFLOW에서 가장 큰 커뮤니티를 차지 하고 있고 python 태그를 통한 질문의 수는 하루에 120만개에 이른다.

추가적인 스크립트에 대해서는

http://docs.python.org/3/ 에서 확인 가능하다.

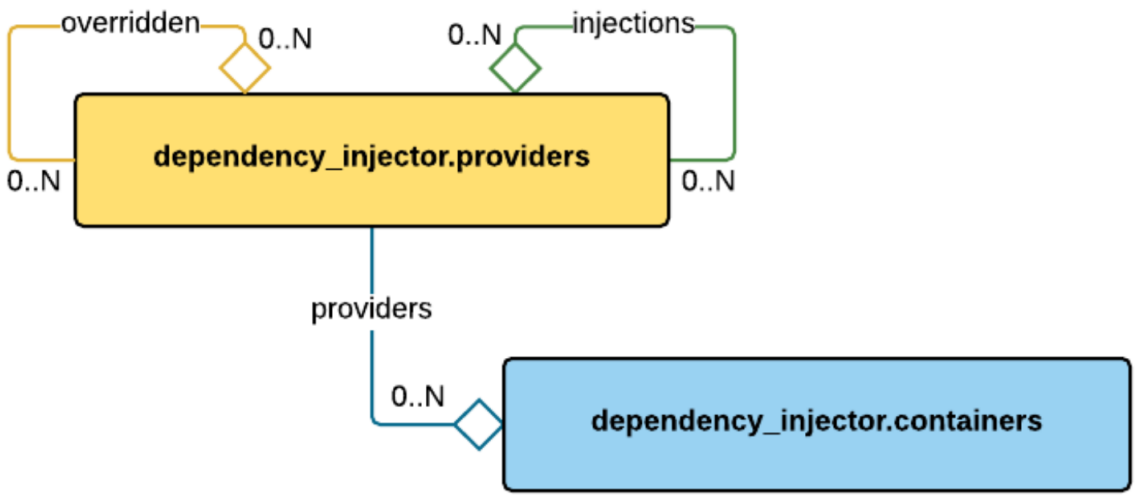
Python Dependency

종속성 주입은 개체간 종속성을 정의 하기 위한 소프트웨어 엔지니어링 기술 이다. 기본적으로 주어진 코드 모음을 구동하는것에 필요한 리소스를 제공하고 그 결과를 받아가는것으로 필요한 리소스는 코드에 종속되어 있다고 말한다. 코드를 작성할때 정의된 다양한 클래스가 있고 객체가 존재한다. 이러한 객체 들은 다른 클래스의 변화에 종속되어있는 경우가 많다. 이렇게 “의존 당하는 클래스”또한 더 잘 사용되는 라이브러리에 의존되어있다. 이런 구성요소를 두고 직접적으로 컴퓨팅 리소스를 사용하는 일은 일종의 “낭비”로 취급 되기 때문에 “종속성 주입”은 이러한 종속 차원의 정의를 처리하고 외부에서 인스턴스화 하거나 작성하는 방법을 제공한다. 종속성 컨테이너는 이 동작을 구현하는데 사용되며 구성요소의 종속성 맵을 보유한다.

객체 A 가 객체 B에 의존하는 경우 변화를 선언하기 위해 헤더파일 형태로 import 함수를 작성해 준다. 이렇게 주입된 의존성은 외부코드에 위임된다.

위에서 언급했듯 파이썬은 타 언어에 비해서 느린 속도를 가지고 있고 그만큼 사용자 친화적이라는 말과도 동일시 되어 낮은 유저 진입 장벽을 갖는다. 이는 추상화 되어있는 코드 구성을 가지고 있어 작은 변화에도 느리게 반응한다는것을 의미한다.

그렇기때문에 Dependency가 필요한 이유이기도 하다. 실질적으로 코드가 구현되는 환경은 가지각색이고 어떤 환경에서는 굉장히 열악한 리소스 자원을 제공할 가능성도 높다. 따라서 그런 문제들을 해결하기 위해 python의 일부 자원 문제 해결법에는 C에 의존하는 방침을 취하고 있다.



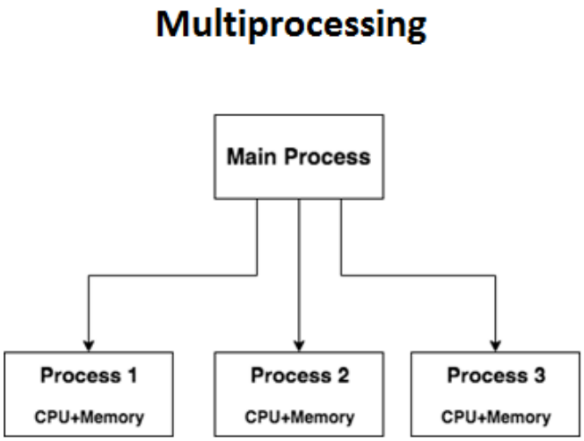
C는 아다시피 가장 기계와 친한 언어로 자리 매김한다. 이는 굉장히 구체적이며 하나의 Instruction을 수행할때 거쳐야 할 추상레이어가 얇다는 의미이다. 이를 조금 더 자세히 설명하면 실질적인 명령어 계산은 C에서 이루어 지고 python 에서는 다른 클래스간의 교두보 역할을 하게 만들어 원활한 구동환경을 만드는데 일조한다.

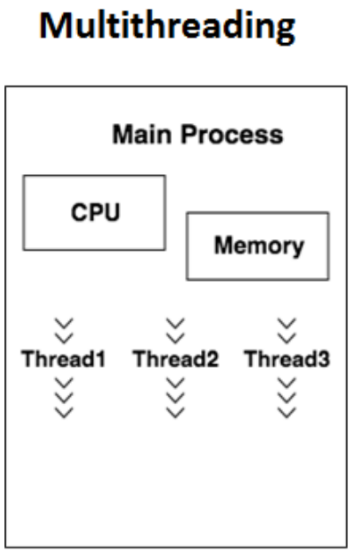
Multiple threads and parallel processing

다중 작업은 현재의 운영체제 환경에서는 흔한일이 되었다. 카카오톡을 하면서 엑셀을 켜는것 만으로도 다중 작업을 하고있는 셈이니 말이다. 멀티 쓰레드 프로그래밍은 그런의미에서 뛰어난 작업 환경을 보장하는 요소로 자리 매김한다. Python 작업을 하다 보면 동시성과 병렬처리에 대한 이야기를 들을수 있다. 동시성은 Concurrency 병렬처리는 Parallelism으로 부른다.

Concurrency : A condition that exists when at least two threads are making progress. A more generalized form of parallelism that can include time-slicing as a form of virtual parallelism.
Parallelism : A condition that arises when at least two threads are executing simultaneously.

(참고 : Sun Micro systems :오라클 Multithread guide)





Program은 작업을 수행하는 구성 단위로 일반적으로 디스크에서 메모리로 배치해와 사용된다. 자체에 메모리 공간을 할당하게 한다.

거기에 Process는 우리가 작동하는데 필요한 모든 자원과 함께 메모리에 로드된 프로그램을 호출하게 된다.

쓰레드는 하나의 흐름이다. 프로세스내의 실행 단위이다. 프로세스의 일부로서 여러 스레드를 실행할수 있고. 각 쓰레드는 프로세스와 메모리 공간을 사용하고 다른 스레드와 결과를 공유한다.

그런의미에서 멀티쓰레딩은 프로세스에서 여러 스레드를 생성하여 서로 다른 작업을 거의 동시에 수행하는 기술이다. 이것은 동시에 실행되는듯한 착각을 불러일으키는 기술이다. 실질적으로 python 에서는 GIL이 있어 스레드가 동시에 실행되는것을 방지한다.

멀티프로세싱은 가장 의미 있는 형태의 병렬처리를 실행하는 기술이다. 여러프로세스가 CPU코어에서 실행되며 리소스는 공유하지 않는다. 각 프로세스는 자체 메모리 공간에서 많은 스레드를 실행할수 있다. python에서 각 프로세스는 명령어 실행작업을 수행하는 자체 인터프리터 인스턴스를 갖는다.

따라서 서로 데이터를 공유하지 않는 범위 내에서의 코드 실행이라면 컴퓨팅 리소스가 남는 한도 내에서 멀티 쓰레딩을 사용했을 경우 높은 이득을 얻을수 있다.

Pandas

판다스를 프로그래밍 환경을 지원하는 소프트웨어 라이브러리로 작성된 python programming 서브모듈 형태의 패키지다. 데이터 조작과 분석을 위한 특히나 숫자 테이블 혹은 시계열 데이터 조작을 위한 DataFrame을 갖는다.

Pandas 라이브러리 기능

- 통합 인덱싱 지원 데이터 조작을 위한 DataFrame 객체
- 메모리 내 데이터 구조와 다른 형식의 데이터를 읽고 쓰기
- 누락된 데이터의 데이터 정렬 및 통합 처리
- 데이터 세트의 재구성 및 피벗
- 대용량 데이터세트의 레이블 기반 슬라이싱 고급 인덱싱 및 하부 설정
- 데이터 구조 열 삽입 삭제
- 엔진별로 그룹화 되어 데이터 세트에서 분할적용 조합작업수행
- 데이터 세트 병합 결합
- 저차원 데이터 구조 확장 및 계층 인덱싱
- 데이터 필터링

Pandas는 Cython 혹은 C로 작성된 세트 구성으로 성능에 최적화 되어 있다.

Openpyxl

openpyxl은 MS사의 엑셀 파일을 읽고 쓰는 python 라이브러리다. 기존라이브러리가 없기 깨문에 python에서 open XML형식으로 읽고 쓰는 기능을 한다.

해당 라이브러리는 필요에 의한 개인이 구축한 라이브러리도 필요한 모든 기능을 지원하지 않는다.

PyQt5

pyqt는 크로스플랫폼 GUI 툴킷 qt의 python 바인딩 라이브러리다.

이는 python 플러그인으로 구현되어 있다.

영국회사 RiverBank Computing이 개발한 무료 소프트웨어 로서 GPL및 상용 라이센스를 포함하여 운영체제간 호환성에 대해 검증되었다.

특징

- 상당한 GUI위젯 세트
- SQL데이터베이스에 엑세스 가능한 클래스

- 자동으로 채워지는 데이터 인식 위젯
- XML의 Parsing 지원
- 윈도우 Active X 컨트롤을 임베딩하는 클래스 지원

Opencv

opencv(open computer vision)은 실시간 컴퓨터 비전을 목표오로 한 프로그래밍 라이브러리다. 원래는 인텔이 개발했다. 실시간 이미지 프로세싱에 중점을 두고 개발되었다. Intel CPU에서 사용되는경우 속도의 향상을 볼수있는 IPP를 지원한다. (intel performance primitive) 해당 라이브러리는 크로스 플랫폰의 형태를 띄고 있으며 tensorflow torch/pytorch 등의 프레임 워크를 지원한다.

베이스 개발언어는 C/C++로 개발되었으며 OCTAVE에 바인딩 되어 프로그래머에게 개발환경을 지원한다.

인간과 컴퓨터의 상호작용을 지원하는 형태로 물체인식 안면인식 모바일 로보틱스 제스쳐 인식등 카메라와 결합되어 실시간 현상 인식을 분석하는데 초점을 둔다.

Sklearn

사이킷런은 기계학습 라이브러리로서 파이썬 프로그래밍 언어를 지원하는 형태이다.

분류 회귀 분석 및 클러스터링을 포함한 여러가지 기계학습 알고리즘 서포트 벡터 머신 , 랜덤 포레스트 , 그래디언트 부스팅 , DBSCAN을 지원하고 수치 과학 라이브러리들과 상호작용 하며 의존성을 갖는다.

Sklearn 은 python으로 작성되어 고성능 선형대수 및 배열 연산에 최적화 되어있으며 해당 연산에는 numpy를 사용한다.

일부 핵심 알고리즘은 Cython으로 작성되어 있어 속도의 향상을 도모 한다. 서포트 벡터 머신은 LIBSVM의 cython 래퍼로 구성된다.

HTML

HTML은 하이퍼 텍스트 마크업 언어로서 웹 페이지를 위한 지배적인 마크업 언어이다. (개발자들은 HTML은 언어로 치지 않는다.)

html은 제목 단락 목록과 같은 본문을 위한 구조적 의미를 나타낸것 뿐아니라 리크 인용과 그 밖의 항목들로 구조적 문서를 만들수 있는 방법을 제곤한다. 그리고 이미지와 객체를 내장하고 대화형 양식을 생성하는데 사용될수 있다. Html은 웹페이지 컨텐츠를안의 꺾쇠 괄호에 둘러싸인 태그로 구성되어 있는 html 형태로 작성한다.

웹브라우저의 같은 html 내의 처리 장치의 행동에 영향을 주는 javaScript와 본문 그리고 그 외의 외관과 배치를 정의하는 CSS 같은 스크립트를 포함 한다.

Css

종속형 시트 혹은 케스케이딩 스타일 시트로서 마크업 언어가 실제로 표기되는 방법을 기술하는 언어로 HTML에 참조되는 형태로 제공되며 에리아웃과 스타일을 정의 할때 자유도가 높다.

단순한 문법을 가지고 수많은 영어 키워드를 사용하여 프로퍼티의 이름을 규정한다.

따로 CSS만의 코드가 필요하지 않고 그대로 HTML 문서에 참조 형태로 내포된다.

JavaScript

자바스크립트는 객체 기반의 스크립트 프로그래밍 언어이다. 웹브라우저 내에서 작동되며 다른 응용프로그램의 내장 객체에도 접근하는 역할을 한다. 또한 Node.js와 같은 런타임 환경에서와 같이 서버 가이드 네트워크 프로그래밍에도 사용된다.

자바스크립트는 본래 넷스케이프 커니케이션즈 코퍼레이션이 처음에는 모카라는 이름으로 출시한 언어이며 속성과 매소드를 갖는 객체 개념이 있고 변수 선언 함수 정의 연산자 그리고 제어문등을 활용 가능하다.

Ajax

Ajax(asynchronous Javascript and XML)로서 비동기적인 웹어플리케이션의 제작을 위해 html css DOM Javascript XMLhttprequest를 이용한다. 자체가 하나의 독특한 기술을 의미하는것이 아니라 함께 하는 기술의 집합을 의미한다.

기존 기술과의 차이점은 기존의 웹어플리케이션은 브라우저에서 폼을 채우고 이를 웹서버로 제출하면 하나의 요청으로 받아 들이는 형태로서 새로운 웹페이지가 데이터 기반으로 제작되어 돌아오는 방식이였다. 이때 최초의 폼을 가지고 있던 페이지와 사용자가 이 폼을 채워 결과물로서 되돌려 받은 페이지는 일반적으로 유사한 내용르 가지고 있는 경우가 많다. 겨로가적으로 중복되는 html을 한번더 받게 되는것은 많은 대역폭을 낭비하는 결과를 가져온다.

반면에 AJAX프로그래밍은 일단 필요한 데이터만 웹서버에 요청해서 받은 다음 클라이언트에서 데이터에 대한 처리를 할수있가. 보통 XML 기반의 웹 서비스 프로토콜이 사용되며 웹서버의 응답을 처리하기위해

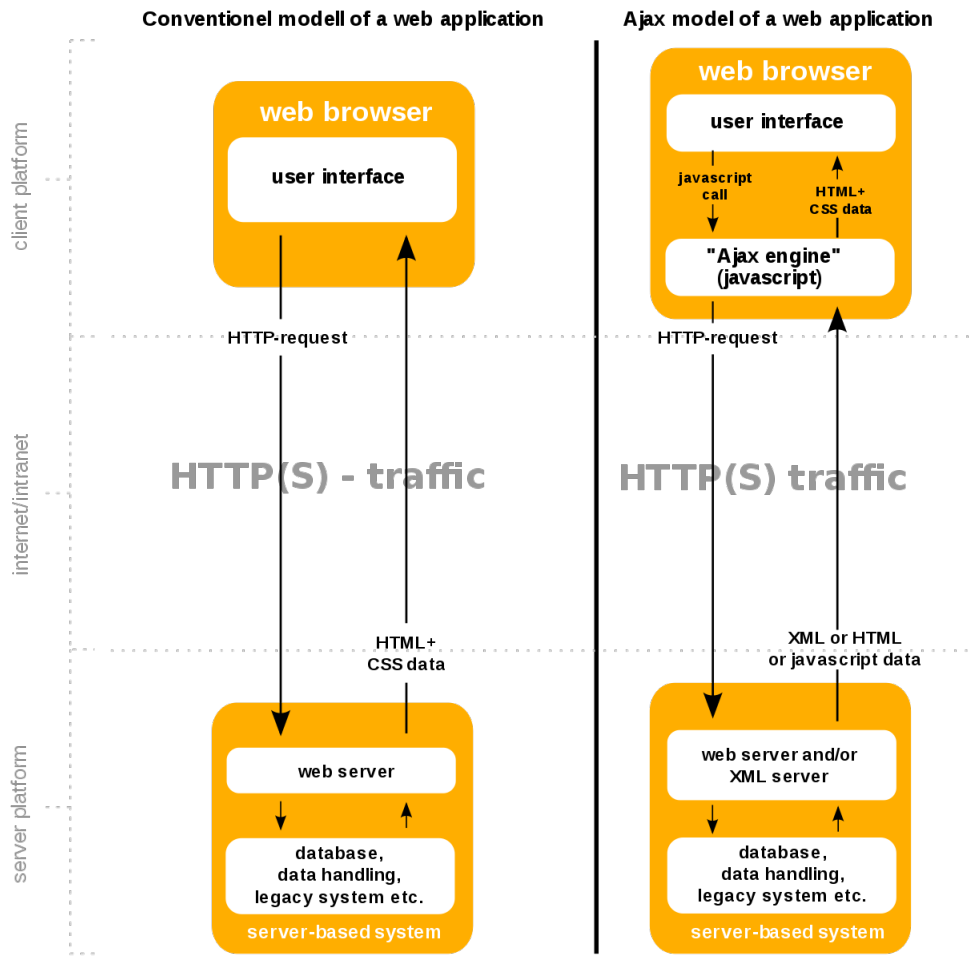
클라이언트 쪽에서는 자바스크립트를 사용한다. 웹서버에서 전적으로 처리하던 데이터 처리의 일부가 클라이언트에서 처리되기 시작하므로 어플리케이션의 응답성도 좋아진다.

또한 웹서버의 데이터 처리에 대한 부하를 줄여주는 일이 요청을 주는 수많은 데이터에 대해서 일어나기 때문에 전체 부하량도 줄어들게 된다.

페이지 이동 없이 고속으로 화면을 전환 할수있다.

서버의 처리를 기다리지 않고 비동기 요철을 가능하게 한다.

수신하는 데이터 양을 줄이고 클라이언트에게 데이터 처리를 위임한다.



MySQL

세계에서 가장 많이 쓰이는 오픈소스의 관계형 데이터 베이스 관리 시스템이다. 다중스레드 다중사용자 형식의 구조 질의어 형태의 데이터베이스 관리시스템으로 오라클이 관리 지원하고 있으며 이중라이센스가 지원된다.

MySQL은 미디어 위키와 드루팔. 같은 웹 어플리케이션에 적용 되며 LAMP,MAMP,WAMP 플랫폼일 데이터 베이스 구성체로서 작동하며 버그질라와 같은 오픈소스 버그 추적 도구에도 사용된다.

웹 어플리케이션으로서 MySQL은 PHP와도 맞물려 있다.

실질적인 소유주는 오라클이며 자체 사용 가능은 오라클 데이터베이스와 맞물려 있어 오픈소스 진영에서 MariaDB 를 만들어 배포했다.

Flask

플라스크는 파이썬으로 작성된 마이크로 웹프레임워크의 하나로 Werkzeug 툴킷과 Jinja2탬플릿 엔진에 기반을 둔다.

특별한 도구나 프레임 워크가 필요하지 않아 마이크로 프레임워크라고 부른다. 데이터 베이스 추상화 계층 양식 유효성 확인 기타 기존의 서드 파티 라이브러리가 갖는 공통 기능을 제공하는 구성요소가 없다.

그러나 플라스크는 자체에서 구현되는 확장 기능을 가지므로 객체 매퍼 양식 유효성 업로드 확인 과 같은 인증 기술 공통 도구들을 위해 존재한다.

추가적인 정보는 http://flask.palletsprojects.com 에서 확인 가능하다.

Apache HTTP Server

아파치 HTTP서버는 아파치 소프트 웨어 재단에서 관리하는 HTTP서버이다. 리눅스, 아파치 웹서버 , MySQL,PHP로 웹서버를 운용하는것

을 머릿글자를 따서 LAMP라고 부른다. tomcat과 같은 웹어플리케이션 서버와 함께 사용가능하다.

mod_SSL을 설치해 HTTPs로 보안을 강화할수 있다.

아파치 웹서버는 세계에서 가장 인기 있는 웹서버로 전체 웹서버의 50%를 차지 하는 점유율을 갖는다.

한국 웹페이지의 42%가 아파치로 구동되고 있다.

Technical Manual

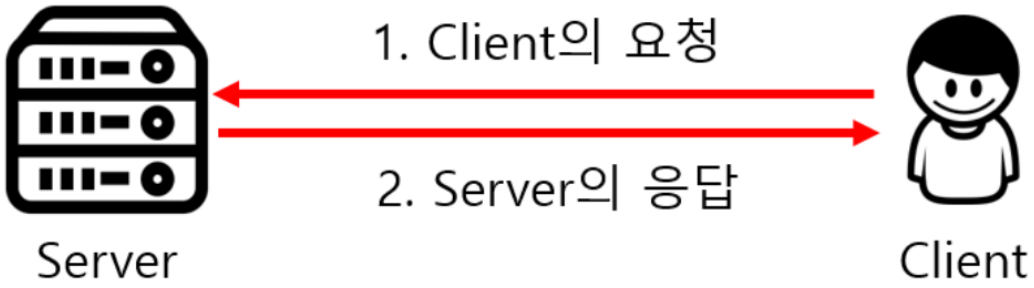
HTTP 통신과 Socket 통신의 차이

일반적으로 단말기에서 필요로 하는 데이터 들은 server에서 관리 한다. 네트워크를 통해서 서버로부터 데이터를 가져오기 위한 통신 방식은 크게 위 두가지로 나눠진다.

1. **HTTP통신**

   클라이언트의 요청이 있을때만 서버가 응답하여 해당 정보를 전송하고 곧바로 연결을 종료하는 방식 으로 서버가 클라이언트로 요청을 보낼수는 없다. 이는 시기산 연결이 아닌 필요한 경우에만 서버로 접근을 허용하는 컨텐츠 위주의 데이터를 사용할때 용이하다. 만약 어떤 데이터에 대한 접근이 실시간으로 이루어 지고 데이터를 받은 후에도 연결이 유지된다면 지속적으로 통신 포트를 열어두고 있어 부하가 걸리게 된다. 일반적으로 모바일 어플

의 경우에는 필요한 경우에만 서버로 전송을 요청하는 경우가 많은데 이러한 웹서버로 HTTP통신을 주로 사용하며 비용 및 유지보수등 많은 부분에서 성능이 좋다.



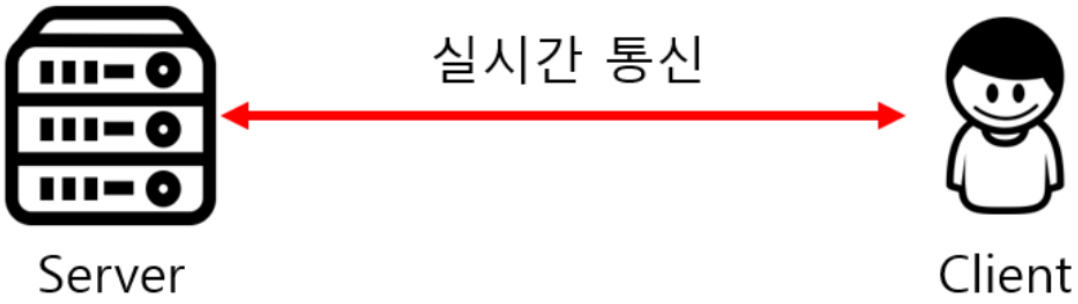
2. Socket 통신

소켓 통신은 서버와 클라이언트가 특정 포트를 통해서 연결을 성립하고 있어서 실시간으로 양방향 통신을 하는 방식이다. 클라이언트는 필요한 경우에 요청을 보내는 HTTP와 달리 서버 역시 클라이언트로 요청을 보낼수가 있다.

계속 연결을 하고 있는 연결 지향적형 통신이기 때문에 실시간 통신이 필요한 경우에 사용한다. 예를 들면 실시간스트리밍이나 영상통화 실시간 채팅과 같은 경우에 해당한다.

이런 상황을 HTTP로 구현하게 되면 계속해서 연결을 요청하기 때문에 서버에 부하가 걸리게 된다.

따라서 온라인 게임과 같은 양 방향성 통신을 위해 구현 되어 있으며 연결이 끊기지 않게끔 포트의 연결을 유지하는것이 중요하다.



Face Recognition Model

얼굴인식은 기계학습이 시작되면서 가장 많은 관심을 받은 분야중 하나이다.

영상속에서 얼굴을 인식하고 그 얼굴이 누구의 얼굴인지 알아채는 과정속에서는 고유 얼굴의 아이덴티티가 보존되면서 빠른 인식 속도를 만족시켜야 한다는 어려운 문제를 헤쳐야만 한다.

이번 프로젝트에서는 얼굴을 인식하는 간단한 방법부터 좀더 복잡한 방법까지 순차적으로 구현해 보면서 적절한 상황에서 보안을 유지 할 수 있는 요소를 어떻게 챙길수 있을지 생각해 보도록 한다.

얼굴인식 이란 얼굴의 특징점을 추출하여 저장된 데이터베이스내 자료와 비교하여 신원을 확인하는 기술이다. 특징점을 추출하는 방법은 크게 2가지인데

- 동맥정맥등을 파악해 피의 흐름으로 생기는 열점을 파악하는 적외선적인 방법

- 3차원 측정을 통해 눈 입 코 턱간의 간격 뼈의 크기 정도를 통해 파악하는 법 등이 있다.

얼굴인식 자체는 타인의 얼굴을 통해 신원을 확인해 직접적인 접촉이 발생하지 않아 사용성이 높지만 오류율이 높아 독집적인 사용보다는 보조의 수준에서 그치곤했다.

하지만 딥러닝이 점차 발전함에 따라 그 완성도가 높아져 현재는 독립적인 인증 수단으로 활용되고 있다.

얼굴인식 시스템의 구성 단계

- 영상획득
- 잡음처리 및 영상처리 분할
- 얼굴영역 검출
- 특징추출 및 특징점간 연산
- 검증

의 과정을 거치는데 여기서 특징점간 연산에는 Euclidean 거리를 적용해 점간의 거리를 연산하는 방식으로 간단한 얼굴인식을 구현해 보겠다.

예시) keras 를 활용한 얼굴 인식

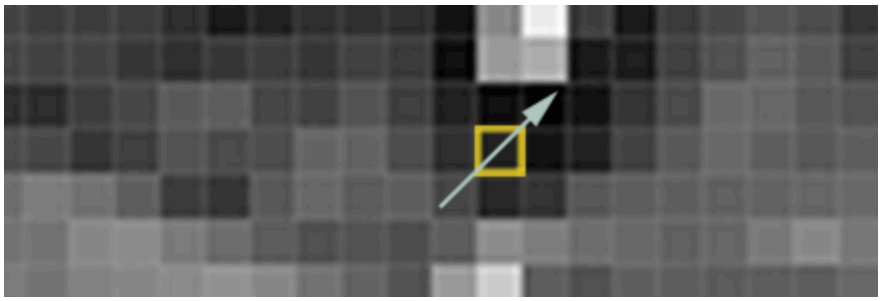
조금더 학습되어 있고 군집형 데이터를 만들기 때문에 변화에 적응형 데이터를 만들수 있다는 장점이 있다.

일단은 임의의 화면에서 얼굴을 찾아내야 한다.

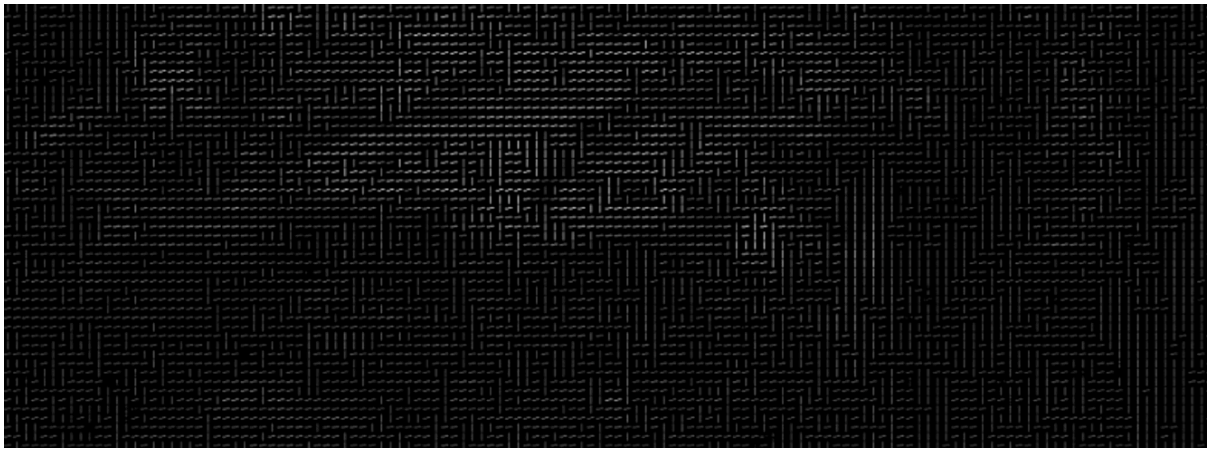
임의의 화면에서 얼굴을 찾는 알고리즘에는 몇가지가 있지만 대표적인 알고리즘을 찾아 보면 2005년에 발명된 Histogram of oriented Gradient 줄여서 HOG라고 불리는 알고리즘을 사용해 보도록 한다.

이미지를 흑백으로 바꾸고 한번에 하나씩 단일 픽셀을 잡고 분석을 시작한다. 그리고 그 픽셀을 둘러싸는 데이터를 찾는다.

여기에서 이미지가 어두워 지는 방향으로 벡터를 설정한다.



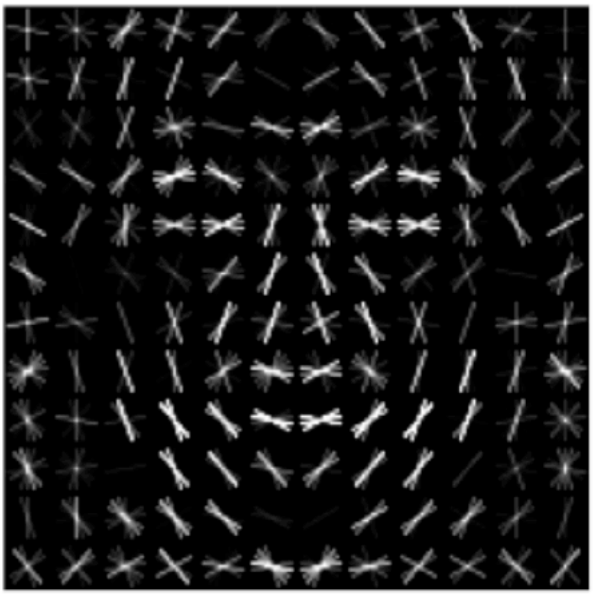
위처럼 벡터화를 모든 픽셀에서 적용하면 다음과 같은 벡터화 이미지를 얻는다.



이런 벡터화 이미지는 경사 형태로 나타나게 되고 그 기울기가 어떤 의미를 갖는지 파악할수 있어샤 한다.
밝기가 변하는 방향성을 따지는 것만으로 일반 이비지와 동일한 값을 가지면서 수치를 탈락된 형태의 일괄 배열을 얻을수 있게된다.
이를 모든 픽셀에 적용하는것은 낭비이므로 위의 영상 분석에도 있었듯 몇개의 픽셀을 묶어 배치 처리 한다.
이를 적용하게 되면 이미지의 성질은 남고 그래디언트는 줄어들어 방향성만 보존된다.

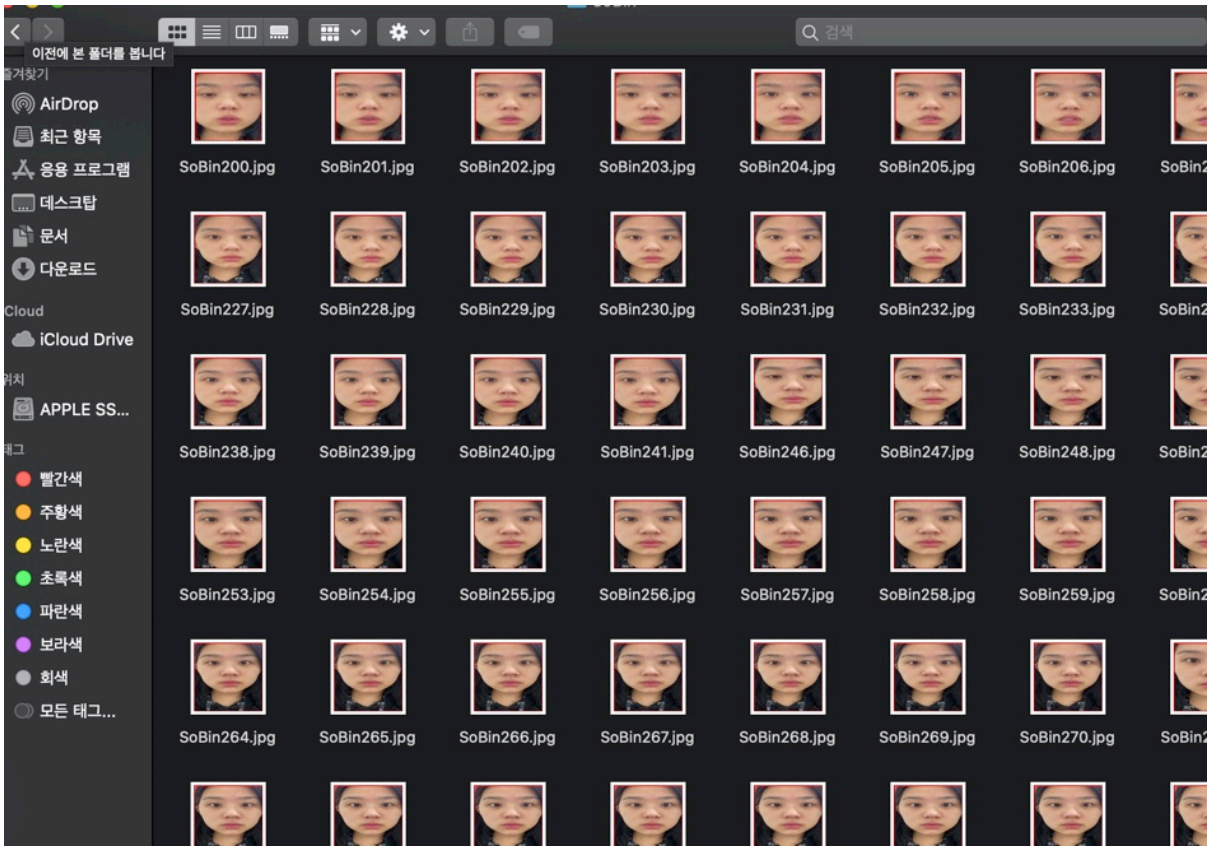
해당 알고리즘이 HOG의 핵심이다.
이를 중심으로 얼굴의 특징 교차점을 찾아낸다.

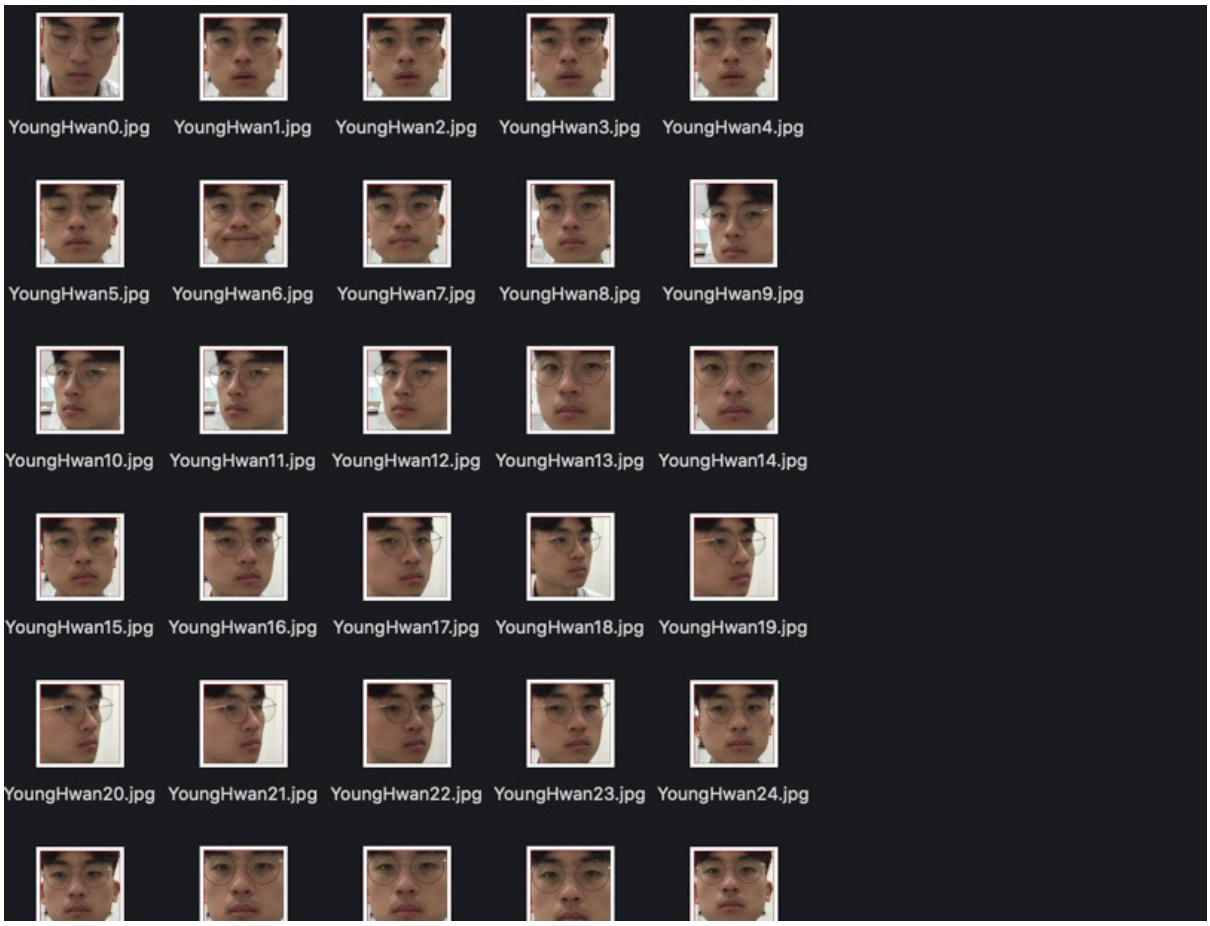
HOG face pattern generated
from lots of face images



이러한 방식을 통해 이미지에서 얼굴을 찾아낼수 있고 얼굴의 윤곽 부터 눈 코 귀 등의 점을 찾을 수 있게 된다.

그 점은 해당 벡터의 교차점과 같다.


이전에 본 폴더를 봅니다
검색
AirDrop
최근 항목
응용 프로그램
데스크탑
문서
다운로드
Cloud
iCloud Drive
APPLE SS...
빨간색
주황색
노란색
초록색
파란색
보라색
회색
모든 태그...
SoBin200.jpg
SoBin201.jpg
SoBin202.jpg
SoBin203.jpg
SoBin204.jpg
SoBin205.jpg
SoBin206.jpg
SoBin227.jpg
SoBin228.jpg
SoBin229.jpg
SoBin230.jpg
SoBin231.jpg
SoBin232.jpg
SoBin233.jpg
SoBin238.jpg
SoBin239.jpg
SoBin240.jpg
SoBin241.jpg
SoBin246.jpg
SoBin247.jpg
SoBin248.jpg
SoBin253.jpg
SoBin254.jpg
SoBin255.jpg
SoBin256.jpg
SoBin257.jpg
SoBin258.jpg
SoBin259.jpg
SoBin264.jpg
SoBin265.jpg
SoBin266.jpg
SoBin267.jpg
SoBin268.jpg
SoBin269.jpg
SoBin270.jpg


YoungHwan0.jpg
YoungHwan1.jpg
YoungHwan2.jpg
YoungHwan3.jpg
YoungHwan4.jpg
YoungHwan5.jpg
YoungHwan6.jpg
YoungHwan7.jpg
YoungHwan8.jpg
YoungHwan9.jpg
YoungHwan10.jpg
YoungHwan11.jpg
YoungHwan12.jpg
YoungHwan13.jpg
YoungHwan14.jpg
YoungHwan15.jpg
YoungHwan16.jpg
YoungHwan17.jpg
YoungHwan18.jpg
YoungHwan19.jpg
YoungHwan20.jpg
YoungHwan21.jpg
YoungHwan22.jpg
YoungHwan23.jpg
YoungHwan24.jpg

이렇게 얼굴을 찾아서 모아보자.

서로 다른 타인의 얼굴셋을 구한다.
해당 이미지를 keras 에 학습 시켜 분류 할수 있게 한다.

```
Epoch 17/30
2925/2925 [==============================] - 11s 4ms/step - loss: 6.0149 - accuracy: 0.6089
Epoch 18/30
2925/2925 [==============================] - 11s 4ms/step - loss: 6.0149 - accuracy: 0.6089
Epoch 19/30
2925/2925 [==============================] - 11s 4ms/step - loss: 6.0149 - accuracy: 0.6089
Epoch 20/30
2925/2925 [==============================] - 11s 4ms/step - loss: 6.0149 - accuracy: 0.6089
Epoch 21/30
2925/2925 [==============================] - 11s 4ms/step - loss: 6.0149 - accuracy: 0.6089
Epoch 22/30
1312/2925 [============>.................] - ETA: 6s - loss: 6.0016 - accuracy: 0.6098
```

활성화 함수로는 relu를 사용하고 분류 예제 이므로 출력 계층2 레이어에 softmax를 붙여주며 중간에 입력과 출력층을 붙여주고 flatten으로 동결 시켜준다.

옵티마이저는 RMSprop를 사용했고 학습률을 0.01을 붙여 적당한 수치로 학습할수 있게 했다.

```
Epoch 1/30
2925/2925 [==============================] - 11s 4ms/step - loss: 0.3548 - accuracy: 0.9221
Epoch 2/30
2925/2925 [==============================] - 12s 4ms/step - loss: 0.1173 - accuracy: 0.9839
Epoch 3/30
2925/2925 [==============================] - 13s 4ms/step - loss: 0.0078 - accuracy: 0.9983
Epoch 4/30
2925/2925 [==============================] - 12s 4ms/step - loss: 0.1064 - accuracy: 0.9901
Epoch 5/30
2925/2925 [==============================] - 12s 4ms/step - loss: 0.0504 - accuracy: 0.9945
Epoch 6/30
2925/2925 [==============================] - 11s 4ms/step - loss: 0.0199 - accuracy: 0.9983
Epoch 7/30
2925/2925 [==============================] - 11s 4ms/step - loss: 0.0091 - accuracy: 0.9986
```

이상하다 학습률이 올라가지 않는다. 이렇게 동결된 학습률의 경우에는 높은 확률도 데이터셋이 문제다.

이미지의 수가 적어 img_generator를 사용해 뒤집고 돌리는 과정에서 얼굴의 탈주가 일어난것이 문제가 된듯하다.

얼굴을 더 많이 촬영하고 뒤집거나 돌리는 행동을 하지 말아보자

훨씬 안정적인 수치를 보여준다. 게다가 성별도 다르고 아예 비슷하지도 않은 다른 얼굴이기 때문에 학습률도 빠르게 증가한다.

```
입력값 : 1_3.jpg
예측결과 : [ 1 ] SoBin / Score 1.0
입력값 : sample_4.jpg
예측결과 : [ 1 ] SoBin / Score 1.0
입력값 : sample_6.jpg
예측결과 : [ 1 ] SoBin / Score 1.0
```

올바른 결과를 보여준다.

실질 적용 모델 버전

방금의 문제는 케라스가 인지 하기에 case by case 형태로 인식 한다는 단점이 존재 한다 따라서 얼굴은 존재 하지만 학습시키지 않은 얼굴각도에 대해서는 낮은 적중률을 보인다.
이를 보정하기 위해서는 어떻게 할수 있을까?

모든 케이스를 학습 시킨다는것은 말이 안된다.

좀더 general 한 케이스를 학습시키고 거기에서 제약을 거는 것이 좀더 현명한 방법일 것이다.

그런의미에서 얼굴의 general한 케이스를 어떻게 찾을수 있을지 생각해보자

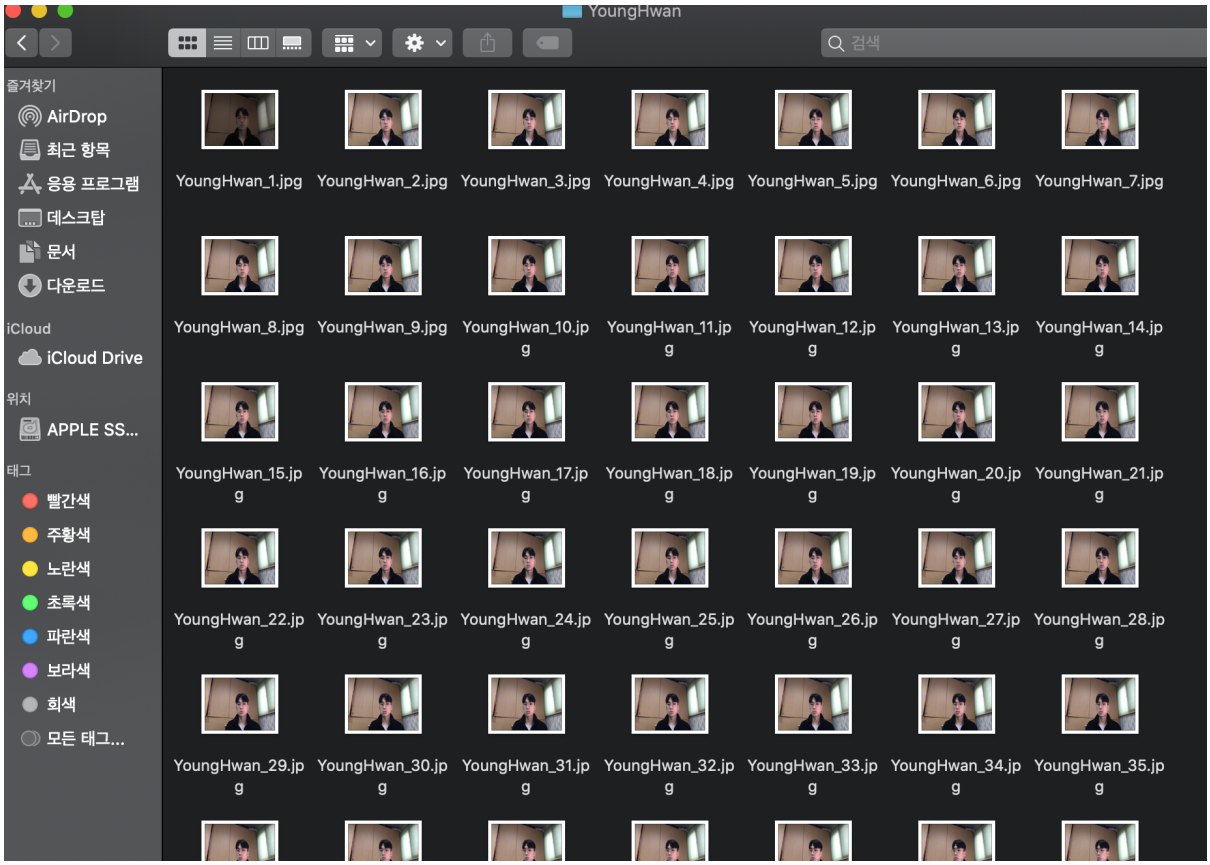
이를 해결하기 위해서 각각의 사진을 꾸깃꾸깃 접어 눈과 코 그리고 입술이 화면의 중앙에 오게 배치 한다면 어떨까?
이렇게 되면 항상 같은 자리에 눈과 코 입술이 위치하고 얼굴의 가장 큰 특징이 고정되어 있으니 적중률도 올라갈 것이다.

이렇게 문제를 해결하는 방식의 이름을

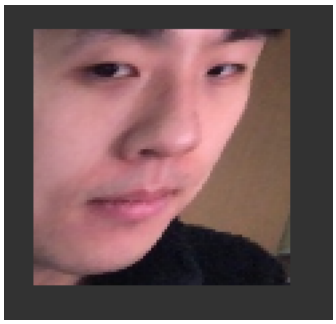
Face landmark estimation 이라고 한다.

이 문제를 위해 2014년 Vahid Kazemi and Josephine Sullivan 의 One Millisecond Face Alignment with an Ensemble of Regression Trees 를 참조 한다.

얼굴에 있는 68개의 랜드마크를 찾아 그중 눈과 코 입술을 고정하는 방식으로 접근해 본다.



다시 문제를 해결하기 위해 얼굴을 촬영한다.
제너레이터로 불려도 좋지만 실질적인 얼굴을 만들기 위해서 프레임 단위로 캡쳐 하도록 한다.



위와 같은 문제를 해결 하기 위해 돌아간 얼굴의 눈과 입을 화면의 중앙에 대치 하도록 얼굴을 비틀었다

이제 얼굴을 인식시켜 보자 이번에는 무작정 집어넣고 학습 시키는게 아니라 얼굴의 특징간 거리를 연산해보자

이 방법은 2015년 구글의 Florian Schroff가 발표한 A Unified Embedding for Face Recognition and Clustering ,가칭 facenet에서 가져온다.
얼굴의 특징을 연산 시키면 128개의 연간 값을 얻고

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	-0.0332478	0.23946142	0.03778371	0.06220788	-0.050417	0.16246817	-0.015704	0.00949947	0.05379989	0.12188932	0.16637443	-0.0512447	-0.0209792	-0.1303369	0.01413867	-0.0476247	-0.06543	0.22771564
2	-0.0222046	0.23762044	-0.0076077	0.08659405	-0.0276727	0.15841052	-0.0340667	-0.039109	0.00715795	0.12671885	0.13783506	-0.0360545	-0.0285258	-0.1364869	-0.025185	-0.0066943	-0.0716138	0.22611056
3	0.01902958	0.24407606	0.01798095	0.12066659	-0.0796827	0.16571648	-0.0521099	-0.0443767	0.00602388	0.08992764	0.16106382	-0.0558145	-0.0105116	-0.1576085	0.05748408	-0.0314646	-0.0435845	0.15906014
4	-0.0314422	0.25831059	-0.078853	0.00894889	-0.0885839	0.15478215	0.00806108	-0.0475097	-0.0306962	0.13427202	0.10696602	-0.0734044	0.03408831	-0.0618875	-0.0911302	-0.0057599	-0.1059631	0.18666957
5	-0.0167512	0.24792285	0.06393857	0.12132957	-0.0562842	0.10563717	-0.0698934	-0.0440757	0.03600154	0.08884626	0.15014154	-0.0810808	-0.0474054	-0.0861761	0.0686776	-0.0655752	0.00372823	0.19271813
6	-0.0403613	0.23384041	0.04757695	0.09682545	-0.0205737	0.1260698	-0.0004909	-0.0086284	-0.0190231	0.15878086	0.10367662	-0.0559826	-0.0624326	-0.1405259	0.02868397	-0.067982	-0.0269306	0.26447636
7	-0.0215582	0.28360191	0.02204047	0.12393177	-0.0528768	0.14115977	-0.0395029	-0.0643267	0.01449611	0.11654554	0.13057584	-0.0759752	-0.0569676	-0.1332817	0.03104265	-0.0768298	-0.0478037	0.20282181
8	-0.0393637	0.27577305	-0.0285469	0.04689509	-0.1151859	0.15300089	-0.0202118	-0.0198584	0.0011524	0.11446692	0.12489171	-0.083307	0.00341211	-0.0925561	-0.0573304	-0.0597957	-0.0907758	0.18027547
9	-0.0164458	0.24872237	0.07807904	0.11522721	-0.0457818	0.11279879	-0.0615826	-0.0353646	0.04207309	0.06356622	0.15151308	-0.0779478	-0.0405206	-0.082575	0.06732699	-0.072392	0.01116855	0.19871999
10	-0.0453434	0.24770567	0.05244706	0.07590897	-0.0304779	0.139406	-0.0150515	-0.0161546	0.02417314	0.12961347	0.12660472	-0.0573517	-0.031439	-0.143409	-0.0026723	-0.0537576	-0.0907833	0.2364486
11	-0.0350873	0.22976916	-0.0739919	0.10981299	-0.0819975	0.13548474	0.0633091	0.00270342	0.00882379	0.1425886	0.12223097	-0.0404109	-0.0051765	-0.1352677	-0.0868304	-0.0540304	-0.1039019	0.12624033
12	-0.0247465	0.26523867	-0.0379376	0.0474149	-0.0663942	0.15077458	-0.0596071	-0.0480152	-0.0336072	0.11606289	0.11132342	-0.0819418	-0.0148133	-0.0915559	-0.0519433	-0.0279012	-0.0491495	0.24274383
13	-0.0039536	0.25449967	0.01669411	0.10282013	-0.0426523	0.13204473	-0.03617	-0.0416189	0.05805783	0.03976227	0.13023971	-0.0457299	-0.007882	-0.150723	0.05232026	-0.0171222	-0.0795565	0.12395755
14	-0.0216041	0.25204894	0.0233827	0.1029289	-0.0531933	0.14697978	-0.055824	-0.0722391	0.04001437	0.13592178	0.14128199	-0.0758906	-0.0177141	-0.1286916	0.03837737	-0.0069279	-0.0657826	0.21433237
15	-0.0532923	0.28679329	0.01214302	0.042936	-0.0175111	0.12836701	0.00523946	-0.0359965	0.03798737	0.15659879	0.12321224	-0.069606	-0.0394287	-0.1041569	0.01501832	-0.0429489	-0.0620203	0.24987145
16	-0.0231759	0.20942737	0.0259975	0.0927807	-0.0045313	0.15692329	-0.0202015	-0.0423411	0.01393092	0.11527615	0.13098612	-0.0457129	-0.0241191	-0.1395026	-0.0145335	-0.005982	-0.0908794	0.24763443
17	-0.0275788	0.25559977	-0.0008127	0.09937051	-0.0061203	0.13414332	-0.0066082	-0.0369531	0.01005528	0.12609826	0.11145117	-0.0537974	-0.0202715	-0.1299056	-0.0155683	0.0030106	-0.0713837	0.22269414
18	-0.0536836	0.27658859	-0.0160184	0.0470215	-0.0258139	0.10254065	0.01967445	-0.0283729	0.02124842	0.17585801	0.09765132	-0.070318	-0.0278622	-0.0779036	-0.0071083	-0.0508153	-0.0180474	0.27788949
19	0.05383235	0.2581006	-0.0011383	0.11441179	-0.080302	0.16593648	-0.0383901	-0.0540542	-0.0299249	0.10023955	0.16566126	-0.0699796	0.01041959	-0.1344981	0.0433155	-0.0253015	-0.013561	0.17483373
20	-0.0296539	0.2474243	0.00799369	0.10863312	-0.0836276	0.14307682	-0.0631409	-0.0585764	0.0811649	0.0855436	0.15949565	-0.0485543	-0.0169906	-0.1231827	0.00828425	-0.0377555	-0.0660732	0.17380738
21	-0.0255715	0.21307427	-0.0049814	0.10088877	-0.0553909	0.17546125	-0.0285828	-0.033028	0.04952098	0.10187092	0.1493264	-0.0565695	-0.0123372	-0.1268799	-0.0065316	-0.0146872	-0.09501	0.20572625
22	-0.0300679	0.25035426	0.02980818	0.09906244	-0.0723985	0.16177428	-0.0734581	-0.0619919	0.05714402	0.11037295	0.13115542	-0.0825465	-0.0430343	-0.1302922	0.06298935	-0.0684255	-0.0652195	0.19380049
23	-0.0228397	0.25300679	0.00943237	0.11243979	-0.081043	0.15628091	-0.0698308	-0.0643419	0.07698923	0.09265563	0.17030101	-0.0543584	-0.0323639	-0.1202224	0.02764579	-0.0340189	-0.0708101	0.16470559
24	-0.032773	0.23706271	0.01041816	0.08738983	-0.0823282	0.15201075	-0.0646152	-0.0160522	0.07665377	0.11784973	0.15429936	-0.0621558	-0.0275337	-0.11097	0.03883814	-0.0568472	-0.0675995	0.17801754
25	-0.0939693	0.12980488	-0.0674094	0.06908146	0.06329463	0.11620094	0.03225993	-0.0805718	0.01412385	0.15977545	0.02746348	-0.025702	0.0161646	-0.0284649	0.00751896	0.05946723	-0.0810353	0.13882472
26	0.00382196	0.22587989	0.03742764	0.13012123	-0.0724248	0.13346504	-0.0630523	-0.0503352	0.04326519	0.09743668	0.15170205	-0.0622263	-0.0289559	-0.0755409	0.06594932	-0.0916353	-0.0069427	0.15576382
27	-0.0226867	0.25266585	0.01467397	0.07730006	0.01343833	0.14731631	0.01443242	-0.0350381	-0.0137049	0.17320144	0.11293391	-0.0639637	-0.0554795	-0.1477155	0.02664452	-0.0234818	-0.049203	0.25248912
28	-0.0386616	0.22943406	-0.0201764	0.12130069	-0.0753327	0.12628563	-0.073671	-0.0439278	0.06771687	0.11675311	0.15797649	-0.026295	-0.0166204	-0.1261405	-0.000822	-0.048233	-0.0440285	0.18461184
29	-0.0645569	0.19120611	-0.0219237	0.11160928	-0.1418315	0.13791826	0.012495	0.02476814	0.10840743	0.10656854	0.13525958	-0.077897	-0.0015603	-0.1172167	-0.0722731	-0.081365	-0.1545213	0.11688908
30	0.01111375	0.2245414	0.04510489	0.07456888	-0.0532669	0.1378164	-0.0413709	-0.0370223	0.02188927	0.12054282	0.12830567	-0.0671315	-0.024468	-0.0954654	0.03414378	-0.085475	-0.0174144	0.2343425

1	./aligned-images/YoungHwan/YoungHwan_1888.png
1	./aligned-images/YoungHwan/YoungHwan_867.png
1	./aligned-images/YoungHwan/YoungHwan_1491.png
1	./aligned-images/YoungHwan/YoungHwan_128.png
1	./aligned-images/YoungHwan/YoungHwan_1000.png
1	./aligned-images/YoungHwan/YoungHwan_1737.png
1	./aligned-images/YoungHwan/YoungHwan_1606.png
1	./aligned-images/YoungHwan/YoungHwan_8.png
1	./aligned-images/YoungHwan/YoungHwan_1001.png
1	./aligned-images/YoungHwan/YoungHwan_1087.png
1	./aligned-images/YoungHwan/YoungHwan_40.png
1	./aligned-images/YoungHwan/YoungHwan_253.png
1	./aligned-images/YoungHwan/YoungHwan_743.png
1	./aligned-images/YoungHwan/YoungHwan_1778.png
1	./aligned-images/YoungHwan/YoungHwan_1879.png
1	./aligned-images/YoungHwan/YoungHwan_671.png
1	./aligned-images/YoungHwan/YoungHwan_388.png
1	./aligned-images/YoungHwan/YoungHwan_1698.png
1	./aligned-images/YoungHwan/YoungHwan_1493.png
1	./aligned-images/YoungHwan/YoungHwan_1244.png
1	./aligned-images/YoungHwan/YoungHwan_1671.png
1	./aligned-images/YoungHwan/YoungHwan_1800.png
1	./aligned-images/YoungHwan/YoungHwan_709.png
1	./aligned-images/YoungHwan/YoungHwan_1393.png
1	./aligned-images/YoungHwan/YoungHwan_776.png
1	./aligned-images/YoungHwan/YoungHwan_1481.png
1	./aligned-images/YoungHwan/YoungHwan_1094.png
1	./aligned-images/YoungHwan/YoungHwan_1258.png
1	./aligned-images/YoungHwan/YoungHwan_60.png
1	./aligned-images/YoungHwan/YoungHwan_1623.png

그 연산 값을 기준으로 사람을 분류 해본다.

라벨에 label encoder 형태로 만들었으면 해당 데이터를 classifier 하도록 한다.

여기서는 sklearn 을 활용해 분류 해 본다.

분류기는 GridsearchSVM을 사용해 분류를 하고 확인을 해보면 결과를 확인할수 있다.

```
=== sample1.jpg ===
Predict ['YoungHwan'] with 1.00 confidence.

=== sample5.jpg ===
Predict ['YoungHwan'] with 0.54 confidence.

=== sample6.jpg ===
Predict ['SoBin'] with 0.90 confidence.

=== test2.jpg ===
Predict ['YoungHwan'] with 0.98 confidence.
```

훨신 유연하게 인식하는 모습이다.
물론 조도가 애매하고 사진이 구려서 얼굴이 정확하게 찍히지 않은 데이터셋도 많을것이고

추가적인 이미지 처리를 해주지도 않았고 feature importance를 고려해 128개의 feature에 비례하는 양의 데이터셋을 모으지도 못했다.

데이터가 충분하지 않은 경우에는 GridSearchSVM보다는 RamdomForestClassifier가 더 강력한 분류기로서 기능하기 때문에 추가적인 실험이 더 필요하다.

얼굴인식의 문제에는 더 많은 요소들이 결합된다.

실질적으로 웹서비스에서 얼굴인식이 사용되지 않는 이유는 각 사람들의 데이터를 저장하는것도 문제지만 분류의 기준이 애매해지기 때문일것이다.

일반적으로 스마트 폰 환경에서는 개인이 혼자 사용하는 경우가 대다수이기 때문에 '나다 아니다' 하는 문제만 해결하지만 A B C D E F G 를 전부 구분하는것은 어려운 문제이기 때문이다.

그런의미에서 여러명의 사람을 구분하기 위해 인원수만큼 모델을 생성하고 사진을 전부 모델에 집어 넣어 predict 하면 어떨까??

다음 문제에서 확인해 보자

얼굴인식은 사실 인간의 입장에서는 쉬운문제이지만 컴퓨터를 학습시킬때는 정확도에 영향을 주는 요소들이 있어 어려운 문제로 여겨지는데 이러한 문제들을 해결하기위해 지난 20년 간 수많은 알고리즘이 구성되게 되었다.

대표적인 얼굴인식 알고리즘으로는

- EigenFace
- 국소이진패턴히스토그램(LBPH)
- FisherFace
- Scale Invariant Feature Transform

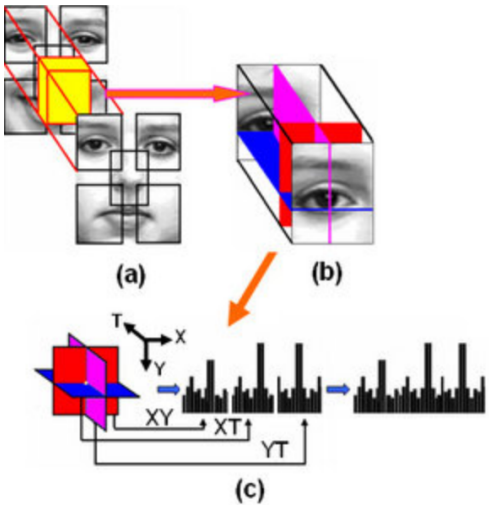
등이 있다. 각 방법은 이미지 정보를 추출하고 입력이미지와 일치 시키는 접근 방식을 갖는다.

이번 버전에서는 가장 오래된 얼굴인식 알고리즘중 하나로 꼽히는 Local Binary Pattern Histogram을 활용해 보자

코드 구현으로는 cv2에서 모델만 임포트 하는것 뿐이라 어려울게 없지만 내용을 이해 하고 넘어가자

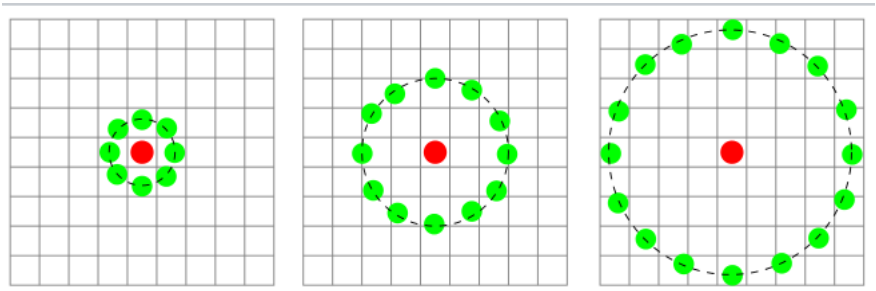
LBP는 각 픽셀의 주변을 임계값으로 지정해 이미지의 픽셀을 레이블링 하고 결과를 0 과 1로 나타내는 방법이다.

94년에 처음 나왔고 버전 3 에서 언급했던 HOG 즉 히스토 그램의 방향성 그래디 언트와 결합해 높은 성능을 뽑을수 있게 되었다.



2차원 텍스쳐가 두가지의 상보적인 추정값을 갖는다고 가정한다. 각 픽셀을 A x A의 그리드로 나누고 각 그리드의 중심 값을 임계값 삼아 확장해 나가는 방식이다.

확장이라고 하면 이해가 안갈수 있지만 다음 그림을 보고 간단히 이해해보자.



각 셀에 8개의 이웃 픽겔로 움직이며 인접값보다 크면 0 아니면 1을 쓴다.

2^8 = 256 자리 이진수로 변환 가능하다는것을 의미한다.

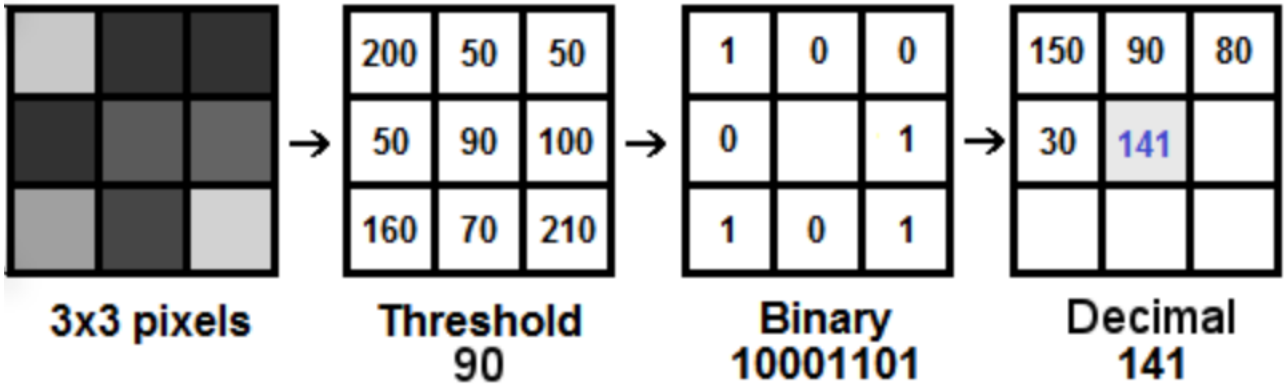
매개변수는 총 4개를 사용한다.

- 반지름
- 이웃수 ->많은 포인트를 설정하면 연상량이 높아진다.
- 그리드 x
- 그리드 y

LBPH의 1번 계산 단계는 얼굴의 특징을 강조하기 위해 원본이미지를 더 잘 표현하는 중간 이미지를 만드는 과정이다. 이를 위해서 매개변수의 반지름과 이웃수에 해당하는 슬라이딩 윈도우를 활용한다.

(해당 슬라이딩 윈도우는 다른 프로젝트의 Object Detection에도 차용된다. R-CNN 기반)

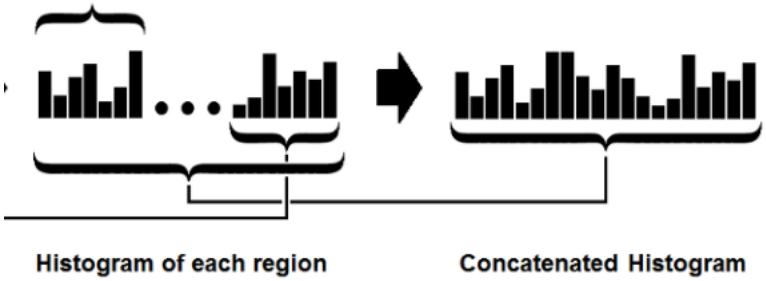
예를 들면 다음과 같다.



-이미지를 회색으로 바꾸고
-이미지의 일부를 잘라 밝기 기준으로 보여준다.
-중간 값을 기준으로 이진수로 치환한다. 8개의 수치를 얻는다.
-픽셀의 중앙에 이진수의 십진수 값으로 치환한다.

이렇게 돌아가는 방식을 다른수의 반지름과 이웃을 사용하도록 확장되어 순환LBP라고 불린다.

여기서 퍼지는 과정에서 일부 데이터 포인트가 픽셀과 픽셀 사이에 있을수 있는데 이러한 경우는 가까운 4개의 픽셀의 내삽을 활용해 조정한다.



위를 통해 이미지는 마치 찰흙에 얼굴을 박은 듯한 이미지로 치환되고
파라미터 그리드 XY에 따라서 구분되어 그리드 분포를 찾을수 있게 된다. 또한 추가적으로

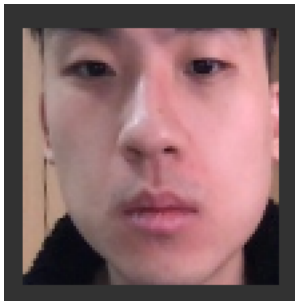
$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

각 값은 유클리드 거리, 카이제곱 , 절대값등의 연산 접근법을 통해 신뢰도를 표현한다.

그럼 다음 알고리즘을 통해 얼굴을 인식한 결과를 살펴 보자

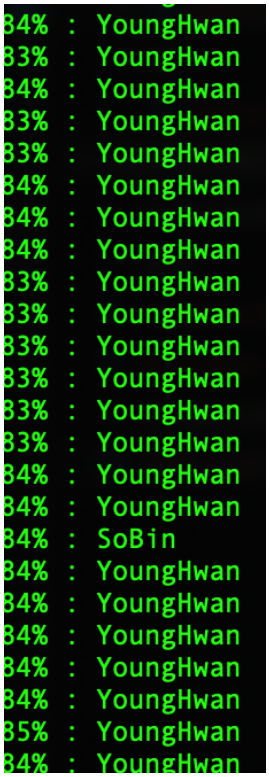
위 버전 3에서 사용했던 데이터셋을 차용해 얼굴을 찾자

얼굴을 찾는 알고리즘은 dlib의 get_frontal_face_detector()를 활용했고 여기서 정확성을 높이기 위해 얼굴의 위치를 고정 시켜 인코딩 했다.



이렇게 고정된 형태의 얼굴 이미지를 학습시켜 주면 결과를 간단히 확인 할수 있다.

한글이 깨져 보인다.
하지만 원활하게 잘 인식하는 모습이다.



물론 데이터셋이 크지 않고 한 장소에서만 찍어 편향된 모습을 보이긴 하지만 프레임 단위로 잘 처리 하는 모습을 볼수 있다.

위의 모델들을 활용해 사내 보안 프로젝트를 관리하는 Task Sheduler에 보안을 강화하는 용도로 활용하기 위해 작업중에 있다.

실질적으로 활용하기 위해서는 위처럼 인식률이 튀는 현상을 바로 잡아야 하는데 추가적인 양질의 데이터를 보강해서 더욱 고도화 하고자 한다.

Regression Model

회귀 모델을 한마디로 정의 하면 어떤 자료에 대해서 그 값에 영향을 주는 조건을 고려하여 구한 평균이다. 통계학적인 관점에서 보면 모든 데이터는 변량으로서의 의미를 갖고 아래와 같이 표현이 가능하다.

$$y = h(x_1, x_2, x_3, \ldots, x_k; \beta_1, \beta_2, \beta_3, \ldots, \beta_k) + \epsilon$$

위 수식에서 h가 위에서 말하는 조건의 평균을 구하는 함수이며 우리는 이것을 회귀 모델이라고 한다. 이 함수는 어쩐 조건이 주어지면 각 조건의 영향력을 고려하여 조건에서의 평균값을 구해준다.

여기서 우리가 주목해야 할것은 “각 조건의 영향력”이다.

뒤에 붙는 엡실론은 오차항을 의미한다. 측정상의 오차나 모든 정보를 파악할수 없는 환경적 영향이 있기 때문에 불확실성이 포함된다.
이것을 시스템 상에서는 잡음 혹은 noise라고 부르기도 한다.
이런 노이즈가 자기 마음대로 나올것 같지만 사실은 평균이 0 이고 일정한 정규분포를 띄는 성질을 갖는다.

우리가 회귀 분석을 한다는것은 함수가 무엇인지를 찾는 과정을 의미한다. 그럼 우리가 추정한 회귀 모델이 정말 맞는지를 어떻게 확신 할 수 있을까?

엄밀하게 말하면 정확하게 맞다! 라고 단정 지을수는 없다.
하지만 아마 그럴것이다라고 어느정도 확신을 가질수는 있다.
바로 그것이 예측치와 실제치 사이의 잔차(residual)이 우리가 가정한 엡실론 안에 들어가는지 확인 하면 된다.
이작업을 우리는 모델검정이라고 한다.

어쨋든 결과적으로 만들고 싶은것은 최대한 실제 h에 맞는 모델을 만드는것이다. 만약 추정을 잘못하면 몇가지의 중요한 조건들을 놓치게 되고 그렇게 되면 실제화 추정이 달라지게 된다. 이를 underfitting 이라고 한다. 반대로 실제 종송변수에 영향을 주는 조건뿐 아니라 단순한 노이즈 까지 반영 된다면 해당 데이터에 너무 “잘 맞는” 모델이 나오게 된다. 이를 overfitting이라고 한다.
오버피팅은 너무 잘맞아서 유도리가 없어 새로운 데이터에 적응을 못하는 상태를 의미한다.

일반적으로 오버피팅의 문제를 자주 다루게 되지만 사실 현실 세계에서 우리가 만드는 대부분의 회귀 모델은 언더피팅의 문제도 가지고 있다.
따라서 함수와 엡실론 사이에서 적정한 줄다리기를 통해 최적의 조건을 갖는 모델을 찾는 과정을 회귀 분석이라고 할수 있다.

선형성 + 비선형성

가장먼저 고려해야할 가정은 선형성과 비선형성이다. 선형성이란 어떤 집합의 원소쌍이 아래의 성질을 만족시키는 것이다.

$$f(c \times u) = c \times f(u)(\text{여기서}c\text{는상수})$$

$$f(u + v) = f(u) + f(v)$$

선형이냐 비 선형이냐를 결정하는 기준은 변수가 아니라 회귀 계수다. 다시 말해 회귀식에서 x를 기준으로 선형함수인지를 판단하는것이 아니다.
일반적으로 쉽게 접하는 모델은 선형성을 가진 경우가 일반적이지만 프로젝트에서 다룰 몇가지 문제들을 살펴보면 딥러닝 기반의 비선형 회귀 모델인 경우가 많다.

회귀모델의 경우에 성분과 관계가 직관적인 경우가 많기 때문에 조건의 영향력을 계산하기가 쉽다 대신 직선으로만 표현되는 문제이기 때문에 표현에 한계가 있다.

따라서 예측이 복잡한 경우에는 딥러닝을 활용한다.
딥러닝은 대표적인 비선형 선형모델이다.

이게 무슨말이나면 현실세계의 대다수의 문제를 풀어내는것이 가능하다는 말이다.
즉 언더피팅의 문제가 발생할 확률이 낮다.

종속변수 개수에 따른 구분

일반적으로 y=h(x)+e 형태로 구성되면 종속 변수는 y 하나 같지만 여러개인 문제에 관해서도 회귀모델을 만들수 있어야 한다.

그래서 y 가 하나면 단변량 회귀 모델(univariate)이라고 하고 종속변수가 2개 이상인것을 다변량회귀모델이라고 한다.

고전적 선형회귀 모델

고전적 선형 회귀 모델은 단변량 선형회귀 모델을 기본으로 한다.

$$\hat{y} = \beta_0 + \sum_{i=1}^{p} \beta_i x_i$$

고전적 회귀 모델은 독립변수의 개수에 따라 명칭을 달리한다.
그리고 단순한 만큼 데이터에 대해 갖는 가정이 굉장히 많다.

- 오차항은 평균이 0이고 분산이 일정한 정규 분포를 갖는다.
- 독립변수와 종속변수는 선형 관계이다.
- 오차항은 자기 상관성이 없다.
- 데이터에 아웃라이어가 없다.
- 독립변수와 오차항은 서로 독립이다.
- 독립변수 간에서는 서로 선형적으로 독립이다.

만약 실제 데이터가 이런 가정을 만족하지 않는다면 고전적 선형회귀 모델은 실제 데이터를 반영하지 못하게 된다.

따라서 다른 방법을 사용해야 한다. 일반적으로 알려진 가이드 라인으로는 다음과 같은 것들이 있다.

- 독립 변수와 종속 변수가 선형 관계가 아닌 경우: Polynomial regression, Generalized Additive Model (GAM)
- 오차항의 확률분포가 정규분포가 아닌 경우: Generalized Linear Model (GLM)
- 오차항에 자기 상관성이 있는 경우: Auto-regression
- 데이터에 아웃라이어가 있는 경우: Robust regression, Quantile regression
- 독립변수 간에 상관성이 있는 경우(다중공선성): Ridge regression, Lasso regression, Elastic Net regression, Principal Component Regression (PCR), Partial Least Square (PLS) regression

일반적으로 많은 수의 피쳐 들은 서로 corr 값을 갖고 그런 의존성 데이터의 특징을 파악 하기 위해서는 feature importance를 고려해 문제를 해결할수 있어야 한다.
이를 위해 알려진 방법은 regulation이라고 불리는 기법이다. 로버스트 회귀 처럼 고전적 모델과 동일하지만 회귀 계수를 추정하는 방법에서 차이가 있다.

가장 특징적인 모델 몇가지를 살펴보자

- Classical linear regression: $argmin_\beta \sum \epsilon_i^2$
- Ridge: $argmin_\beta \epsilon_i^2 + \lambda \sum \beta_k^2$
- Lasso: $argmin_\beta \epsilon_i^2 + \lambda \sum \|\beta_k\|$
- Elastic net: $argmin_\beta \epsilon_i^2 + \lambda_1 \sum \beta_k^2 + \lambda_2 \sum \|\beta_k\|$

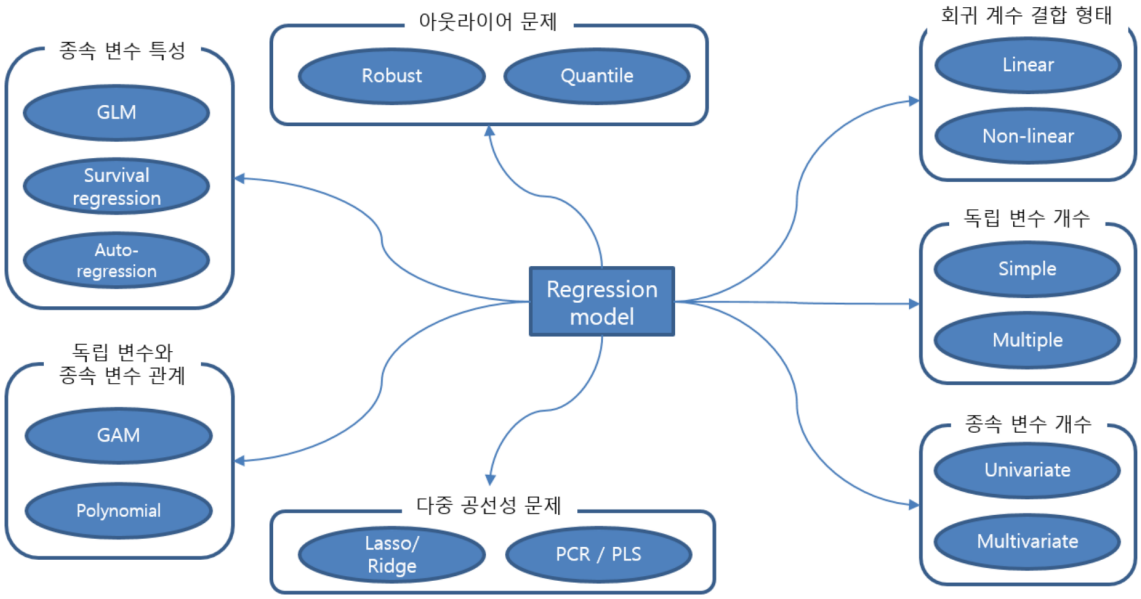
위의 수식을 보면 알수 있지만 고전적인 선형회귀 모델은 회귀 계수를 추정할때 잔차의 제곱의 합을 계산한다. 이 합수를 비용함수 혹은 loss function이라고 하는데 이렇게 생긴 비용함수가 최소가 되는것을 찾는과정이 회귀계수를 찾는 과정이다.

하지만 다른 모델들을 보면 추가항들이 있음을 알수 있다. 이런 추가항들을 패널티 함수 하고 부른다. 말 그대로 회귀 계수 값이 너무 작아지거나 커지지 않도록 패널티를 줌으로서 회귀계수 값을 보정하는 기능을 한다.

이 패널티 함수의 특징에 따라 Ridge 와 Lasso가 구분된다. 그리고 Elasticnet은 이 두개를 결합한 방식이다.

독립 변수의 개수가 데이터 보다 많을때 이렇게 규제항을 사용해 변수 선택의 효과를 가져갈수가 있게 된다.

지금 까지 한 이야기를 정리하면 다음과 같다.



5 Project . 1 SAR Team Auto Power Measurement

이 프로젝트에 관한 자세한 내용을 추가적인 첨부 자료에서 확인 할 수 있다.

이미 완료한 프로젝트 결과물이기 때문에 배경 지식에 관한 설명은 생략하고 바로 코드 분석과 결과물 자동화 출력을 확인 하도록 한다.

측정 데이터 관찰

위에서 언급한 것처럼 소프트웨어를 만드는 과정에서 가장 큰 부분은 입력 데이터가 어떤 것이냐 하는 내용이다. 입력 데이터 확인을 해보자.

```
,1,12,20.50998,24.17917,6.322861,0.83,0.83,
,1,24,20.52176,24.2659,6.85215,0.83,0.83,
,12,0,20.71823,25.61649,7.667542,0.83,0.83,
```

Csv 형태 즉 쉼표 기반 데이터 구분 파일이다. 이러한 데이터는 앞에서 부터 key 기반 value 형태로 구성되어 있기 때문에 순차적으로 읽어주게 되면 올바른 값을 가질수 있다.

일괄적으로 데이터를 읽어오기 위해서는 데이터 변환이 필요하다.

pandas를 활용해 DataFrame 형태로 변환해 준다.

```
import sys, UI
import pandas as pd
import openpyxl as op
from PyQt5.QtWidgets import *
from PyQt5.QtGui import QIcon,QFont
from PyQt5.QtCore import QCoreApplication,QDate,Qt
```

외부 모듈을 임포트 해주고 python과 호환성을 체크 한다.

일반적으로 각 모듈간 아구가 맞지 않아 돌아가지 않는 경우가 허다하다.

위의 csv 파일을 데이터프레임으로 바꿨다고 했다.

그 결과를 가시적으로 활용하기 위해 excel 로 출력해본다.

ID	Band	BW	FREQ	CHN	Mode	RBO	RBS	UE	Peak Pwr	FE	EAO	EAI	MPR
140	LTE41W	5MHz	0	39750	QPSK	1	0	20.53619	24.25912	7.839203	0.83	0.83	
141	LTE41W	5MHz	0	39750	QPSK	1	12	20.50998	24.17917	6.322861	0.83	0.83	
142	LTE41W	5MHz	0	39750	QPSK	1	24	20.52176	24.2659	6.85215	0.83	0.83	
143	LTE41W	5MHz	0	39750	QPSK	12	0	20.71823	25.61649	7.667542	0.83	0.83	
144	LTE41W	5MHz	0	39750	QPSK	12	6	20.70999	25.54453	8.869171	0.83	0.83	
145	LTE41W	5MHz	0	39750	QPSK	12	11	20.69659	25.55676	7.982254	0.83	0.83	
146	LTE41W	5MHz	0	39750	QPSK	25	0	20.72638	25.99982	9.799004	0.83	0.83	
147	LTE41W	5MHz	0	39750	16QAM	1	0	20.59857	25.05502	8.926392	0.83	0.83	
148	LTE41W	5MHz	0	39750	16QAM	1	12	20.6008	25.04013	5.679131	0.83	0.83	
149	LTE41W	5MHz	0	39750	16QAM	1	24	20.59885	25.07306	6.036758	0.83	0.83	
150	LTE41W	5MHz	0	39750	16QAM	12	0	20.75168	26.55835	7.867813	0.83	0.83	
151	LTE41W	5MHz	0	39750	16QAM	12	6	20.75436	26.42084	7.46727	0.83	0.83	

그림19. .csv 파일 형태로 [그림 17]의 raw data를 각 항목에 맞춰 data 정렬

행과 열에 맞춰서 같은 포멧을 가진 데이터가 정렬되어 있다.

여기서 데이터가 어긋나게 되면 굉장히 곤란해 지므로 올바른 데이터를 얻을수 있는지 확인하는 과정을 거쳐야 한다.

Raw 데이터는 (UL korea) 사내 양식에 맞게 올바르게 출력되고 있다.

값이 중복되는지 확인하고 오차 범위내의 수치 인지 확인해야 한다.

만약 오차를 벗어났으면 재 측정을 통해서 결과를 다시 확인한다.

프로그램에서 연산하는 변수들은 다음과 같다.

- ✓ Frequency Band
- ✓ 각 주파수 별 Bandwidth
- ✓ Channel
- ✓ Resource Block allOcation (이하 RBO)
- ✓ Resource Block offset (이하 RBS)
- ✓ 측정된 Power (이하 UE)

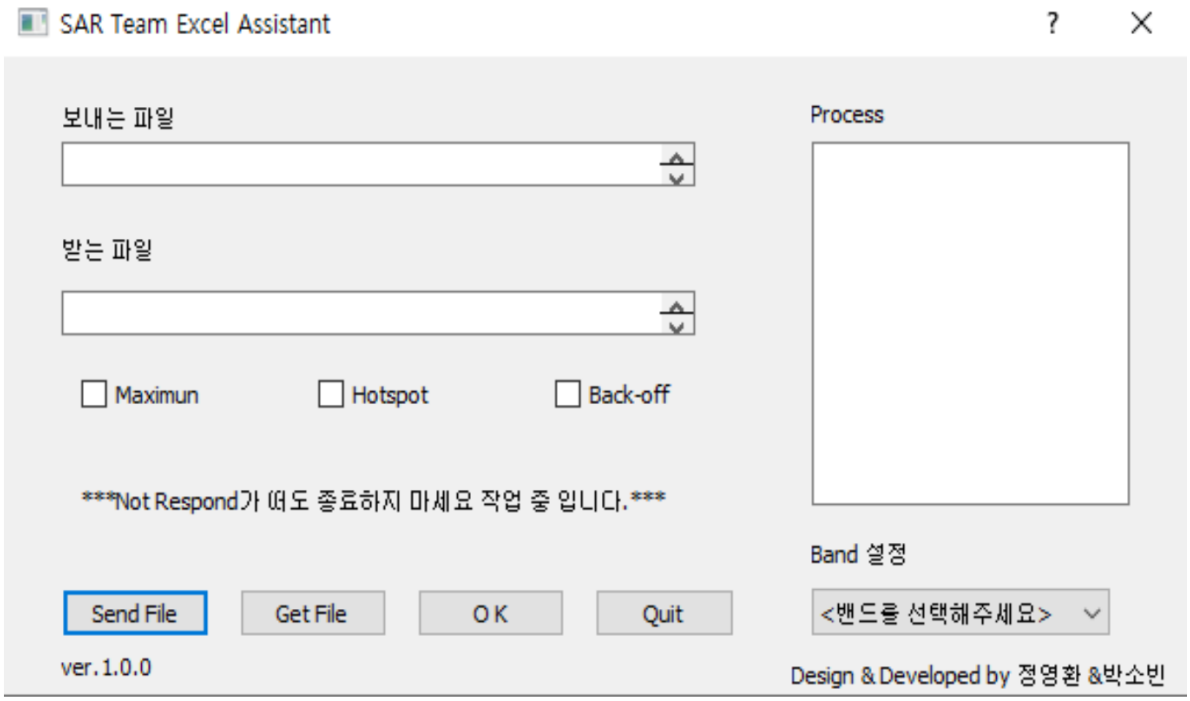
위 6개의 변수 이외에 다른 값은 파워 측정에 사용되지 않는다.

위 값들이 파라미터에 해당하고 계측수치를보정할지 말지 결정하는 결정변수가 된다.

이외의 데이터는 디바이스에 대한 정보를 담고 있기 때문에 사내 보안상 보고서에 작성할수 없다.

GUI 구성 및 객체 할당

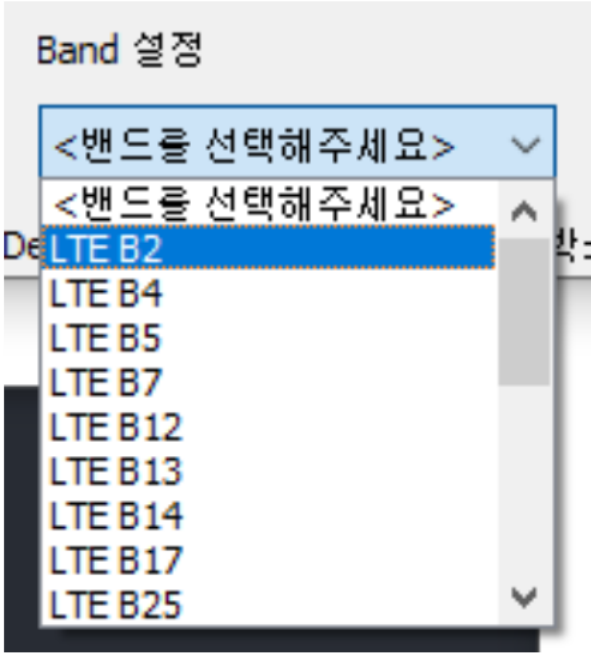
PyQt5 Building UI를 통해서 사용자가 사용할 UI를 구성한다.



사용성을 위해서 불러오는 파일과 타켓 파일을 구성했고 받는 파일

의 경우에는 올바른 위치에 들어갈 경우에 Process에 결과 보고를 할 수 있도록 객체를 할당했다.

추가적으로 측정한 부분이 Max인지 Hot-spot 인지 Reduc 인지 수치만 보고는 알수 없으므로 사원 분들이 직접 선택할수 있도록 체크박스 형태로 화면을 구성했다.



오른쪽 아래를 보면 밴드가 있다. Power 측정 device에서 나오는 결과 값에는 밴드 명이 출력되지 않는다.

따라서 사원 분들이 측정한 밴드가 어디값인지 직접적으로 할당해 주어야 하는 부분이므로 드롭박스 형태로 결과창을 띄우게 했다.

화면 크기

실행 프로그램은 일반적으로 타 특정 프로그램과 동시성을 가지고 사용된다 따라서 화면 레이아웃과 크기를 수용해 사원분들의 의견을 받아 제작하였다.

코드 구성

```
class MyApp(UI.Ui_Dialog,QDialog):
    global send_name
    global get_name

    global Max_power
    global Hotspot_power
    global back_off_power
#######################################################
```

클래스를 선언하고 들어갈 변수 명을 할당한다.

Sequencial 하게 움직이는 코드인 만큼 한번에 하나의 데이터를 옮기게 설정한다.

위치가 안맞으면 아예 다른 값이 들어가기 때문에 속도와 정확성을 처리 할수 있어야 한다.

```
#이니셜라이즈 및 셋업 과정
    def __init__(self):
        super().__init__()
        QDialog.__init__(self, None, Qt.WindowStaysOnTopH
        self.setupUi(self)
        self.date = QDate.currentDate()
        self.initUI()
```

프로그램 실행시 가장 먼저 실행될 initialize 함수이다.

기본 세팅이 되고나서 프로그램이 실행되는 순서이기 때문에 init에는 많은 수치를 할당하지 않는다.

그러므로 해당 함수에는 기초 ui와 전체 함수 대기 형태로 구성했다.

```
면 출력에 관해서
def initUI(self):
    QToolTip.setFont(QFont('SansSerif', 10))
    self.center()

    #버튼과 ui 상호작용
    self.send_file.clicked.connect(self.OnOpenDocument)#
    self.send_file.setToolTip('파일 불러오기 입니다.')

    self.get_file.clicked.connect(self.OnOpenDocument2)
    self.get_file.setToolTip('파일 불러오기 입니다.')

    self.quit.clicked.connect(QCoreApplication.instance(
    self.quit.setToolTip('종료하기')

    self.ok.clicked.connect(self.okFunction)
    self.ok.setToolTip('확인')

    #체크박스
    self.max_chk.stateChanged.connect(self.chkFunction)
    self.hot_chk.stateChanged.connect(self.chkFunction)
    self.back_chk.stateChanged.connect(self.chkFunction)
    #콤보박스
    self.combo.activated[str].connect(self.comboEvent)
##########################################################
```

유저 인터페이스에 객체를 할당해 준다. 객체를 할당해준다는 의미는 함수와 버튼을 연결하고 겨로가값을 화면에 표시 할수 있도록 객체를 활성화 시킨다는 것을 의미한다.

콤보 박스, 버튼 ,텍스트 박스등을 코드와 연결한다.

여기서 변수가 겹치게 되면 같은 값이 다른 버튼에 할당 되므로 정확한 값이 할당될수 있도록 주의 해야 한다.

또한 메모리 접근이나 배열의 주소 allocate 문제도 존재 한다.

따라서 하나의 변수를 재활용하기 보다는 다른 변수를 만들고 기존의 변수를 순차적으로 폐기하는 방식을 사용하는 것이 좋다.

가장 중요한 부분은 ok 버튼을 눌렀을 때다.

```
def okFunction(self):
    global Row
    send = pd.read_csv(send_name)
    target = op.load_workbook(get_name)
    Row=len(send.index)
    sheet = target[Band]
    sheet_row = len(sheet['A'])
    print('time start')
    time_a = time.time()
r문은 1번만 사용할것
```

따라서 버튼을 누른 다음 python 내부 적으로 일어나는 함수의 변화를 보자

누른 다음에 와서야 방금 할당했던 데이터를 메모리에 올린다.

추가 적인 속도를 위해서 라면 할당하자 마자 메모리에 올리는것이

함수의 운용시간을 위해서 좋다.

추가적인 요구 사항이 생겨서 ver 2.0을 만들게 된다면 해당 부분을 수정해야겠다.

또한 불러온 csv 파일을 할당하는 과정을 거친다.

그후에는 반복문을 찾아 움직이면서 문제를 해결한다.

```
for i in range(Row):
나 데이터 프레임을 만들면 메모리 낭비가 심함
    x3 = send[' RBO'][i]
    x4 = send[' RBS'][i]
    x5 = send[' CHN'][i]
    self.textBox.append('Processing....'+str(i)+'\n')

    if (Band =='LTE B41') or (Band == 'LTE B41(HPUE)') or (Band == 'LTE B41(IC)'):
        if (Max_power == 1) and (Hotspot_power == 0) and (back_off_power == 0):
            if x5 == 39750:
                y5 = 'E'
            elif x5 == 40185:
                y5 = 'F'
            elif x5 == 40620:
                y5 = 'G'
            elif x5 == 41055:
                y5 = 'H'
            elif x5 == 41490:
                y5 = 'I'
```

여기서는 Pandas 기능을 통해 column명으로 설정되어 있는 변수 들 중 핵심 변수인 6개 항목(Frequency band, Bandwidth, Channel, RBS, RBO, UE)을 골라 사용할 수 있다.

send[‘ CHN’][i]에 해당하는 데이터를 검색하여 변수 x5에 저장한다. UI 화면에서 콤보 박 스를 통해 선택한 Band 종류에 따라 해당 Excel

Sheet로 이동하고, 체크 박스를 통해 선택한 power value type에 따라 해당 열에 맞춰 data가 입력된다.

```
#여기서 부터 모드별로 시작과 끝을 정함
if send[' Mode'][i]== ' QPSK ':
    fin=st+6
elif send[' Mode'][i]== ' 16QAM ':
    st = st + 7
    fin= st + 6
elif send[' Mode'][i]== ' 64QAM ':
    st = st + 14
    fin= st + 6
elif send[' Mode'][i]== ' 256QAM ':
    st = st + 21
    fin= st + 6
```

데이터를 비교하면서 Transmission Mode (QPSK, 16QAM 등)를 설정하고, 설정 된 모드를 기반으로 채널을 할당해 변수(st, fin)를 조율한다. .csv 파일 (보내고자 하는 파일)에서 각 mode 별로 (RBO, RBS) 쌍의 파라미터가 순서대로 정렬되어 있다.

따라서 해당 mode의 데이터 시작점(st)과 끝점(fin)을 안다면 알고리즘이 간단 해진다.

```
#시작 값을 정하고
#이게 각 밴드마다 전부 다름 그러니까 위치를 하나하나 따는게 좋음
if send[' BW'][i]== ' 5MHz ':
    st = 105

elif send[' BW'][i]== ' 10MHz ':
    st = 74

elif send[' BW'][i]== ' 15MHz ':
    st = 43

elif send[' BW'][i]== ' 20MHz ':
    st = 12
```

각 Frequency band 별로 지니고 있는 Bandwidth의 종류가 다양하다.

따라서 value가 [받는 파일(.xlsl)]에서 위치에 있어야 하므로 시작점(st)를 미리 선언한다.

여기서 몇가지 문제가 발생한다.

계측 회사의 장비와 유엘의 내부 기준이 다른것이 바로 그것이다.

따라서 실제 측정 값을 적절하게 보정해서 올바른 값으로 만들어줄필요가 있다.

일단 먼저 어떤 밴드에서 문제가 발생하는지 정리 하고 코드에 반영해서 수치 보정을 한다.

```
#결측값 보정 해주는 코드 임 다른 밴드에서 고장난 부분이 있으면 주
for j in range(st,fin+1):
    if send[' BW'][i]== ' 5MHz ' and x3==12 and x4 == 6:
        y3 = sheet['C'][j].value
        y4 = sheet['D'][j].value-1
    elif send[' BW'][i]== ' 5MHz ' and x3==12 and x4==11:
        y3 = sheet['C'][j].value
        y4 = sheet['D'][j].value-2
    elif send[' BW'][i]== ' 10MHz ' and x3==1 and x4==24:
        y3 = sheet['C'][j].value
        y4 = sheet['D'][j].value-1
    elif send[' BW'][i]== ' 10MHz ' and x3==25 and x4==24:
        y3 = sheet['C'][j].value
        y4 = sheet['D'][j].value-1
    elif send[' BW'][i]== ' 15MHz ' and x3==36 and x4==18:
        y3 = sheet['C'][j].value
        y4 = sheet['D'][j].value-2
    elif send[' BW'][i]== ' 15MHz ' and x3==36 and x4==35:
```

이 과정에서 오류 값을 줄이고 데이터를 한번더 스캔 하기 때문에 문제가 일어날 확률을 눈에 띄게 줄일수 있다.

BW (MHz)	Mode	RB Allocation	RB offset	Measured Pwr (dBm)		
				20407	20525	20643
				824.7 MHz	836.5 MHz	848.3 MHz
1.4 MHz	QPSK	1	0	23.2	23.3	23.7
		1	3	23.1	23.3	23.6
		1	5	23.1	23.3	23.7
		3	0	23.1	23.3	23.6
		3	1	23.1	23.3	23.6
		3	3	23.1	23.3	23.6
		6	0	22.0	22.2	22.6
	16QAM	1	0	21.7	22.1	22.4
		1	3	21.7	21.9	22.4
		1	5	21.7	21.9	22.5
		3	0	22.0	22.2	22.5
		3	1	21.9	22.2	22.5
		3	3	21.9	22.2	22.5
		6	0	21.1	21.1	21.5
	64QAM	1	0	20.8	21.1	21.8
		1	3	20.7	20.6	21.7
		1	5	20.7	21.3	21.5
		3	0	20.9	21.1	21.5
		3	1	20.9	21.1	21.4
		3	3	20.9	21.1	21.5
		6	0	20.0	20.1	20.4
	256QAM	1	0			
		1	3			
		1	5			
		3	0			
		3	1			
		3	3			
		6	0			

결과 파일을 뽑아서 확인하면 다음과 같다.

만약 빈칸이 존재한다면 그것대로 문제가 되는데 깔끔하고 오차 없이 올바른 값이 들어간것을 볼수 있다.

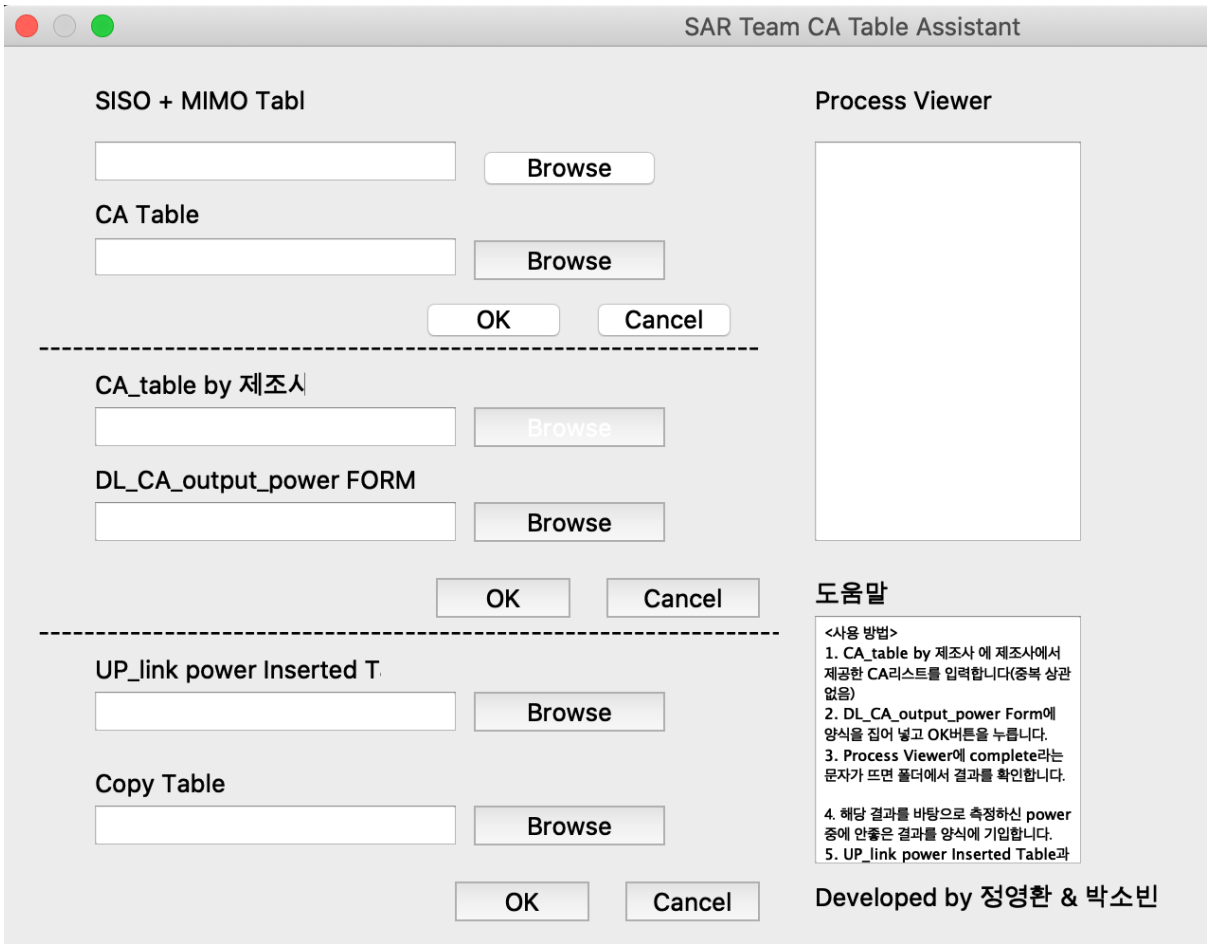
어려운 프로젝트는 아니였지만 빠르게 문제를 해결할수 있다는 점에서 굉장히 고무적인 프로젝트에 해당 했다.

6 Project.2 SAR Team CA Combination Program

CA combination에 관한 자세한 설명은 CA Combination and ReconFiguration Automation Process 를 참조한다.

여기서는 간단한 코드 정리와 함께 사원분들의 피드백을 받아 구현된 추가적인 내용과 ver.2.0 의 4*4 MIMO 내용을 포함한 몇가지 내용을 소개하고자 한다.

실제 제작되고 배포된 CA program의 레이아웃은 다음과 같다 .



프로그램의 흐름은 다음과 같다.

- 제조사에서 제공한 CA 조합을 입력한다.
- 지정된 양식을 프로그램에 업로드 한다.
- 지정한 포멧이 맞다면 MIMO와 non-MIMO를 구분한다.
- MIMO내부에서 중복 조합을 제거 한다.
- non-MIMO 내부에서 중복 조합을 제거 한다.
- 제거된 유일 조합을 분산 처리 하여 새로운 DataFrame을 만든다.
- 해당 프레임은 중복되지 않는 선에서 유일한 키 값을 갖는다.
- “측정된 워스트 케이스를 직접 작성해서 넣는다.” : 사원분들이 하셔야 할 유일한 행동
- 워스트 케이스 기반으로 inter 와 intra를 구분하고 contiguous와 non-contiguous를 구분하여 모든 케이스에 대한 CA 조합의 밴드 위치를 파악해 자동으로 결과를 출력한다.

추가 구현 사항

- 사원분들의 UL 내부 정렬 방식과 맞지 않아 정렬 문제 해결
- 임의 밴드의 범위가 지정 테이블과 맞지 않는 경우 해결
- 4*4 MIMO 분석 및 해결 문제 추가
- 레이아웃 재 구성

코드 구성

```
import openpyxl as op
from openpyxl.styles import  PatternFill,Color
from collections import Counter
import sys
import copy
```

외부라이브러리를 import 한다.

중복 수치들만 빠르게 확인 할수 있어야 하므로 styles . Patternfill 을 추가적으로 import 한다.

엑셀 파일을 읽어오고 들어온 결과를 새오운 메모리에 할당해 얹어놓는다. 중간 연산을 가능하게 해서 추가적인 파일 생성을 가능하게 만들도록 한다. 해당 코드는 다음과 같다.

```
#original = re_test
send =  op.load_workbook('sample1.xlsx')
target = op.load_workbook('DL_CA_output_cpy.xlsx')
sheet = send['Sheet1']
tar_table = target['Sheet1']

index_len=len(sheet['A'])

index_len=len(sheet['A'])

blank_list=[]

index_list=['A','F','K','P']
cc_list=['B','G','L','Q'] #2cc,3cc,4cc,5cc
rev_list=['E','J','O','T']
cover_list=['D','I','N','S']
res_list=['C','H','M','R']

st = 3
ok_list=[]
no_list=[]
ch_cnt=0
CA_cnt=3
```

구성 된 변수는 각 테이블의 몇개의 탭에 나뉘어 분산되어 있다.

따라서

A F K P , B G L Q , E J O T 와 같이 같은 기능을 하는 연산 탭을 묶어 한번에 처리 하도록 한다.

겹침 문제를 해결하기 위해 성공한 리스트를 만들기 위해 메모리를 할당한다.

```python
def Band_cal(a):
    data = []
    if int(a) == 2:
        data.append(20)
        data.append(900)
        data.append(1960)
    elif int(a) == 4:
        data.append(20)
        data.append(2175)
        data.append(2132.5)

    elif int(a) == 5:
        data.append(10)
        data.append(2525)
        data.append(881.5)
    elif int(a) == 7:
        data.append(20)
        data.append(3100)
        data.append(2655)
    elif int(a) == 12:
        data.append(10)
        data.append(5095)
        data.append(737.5)
```

코드 중 일부 이다. 밴드별로 mid channel이 다르다. 따라서 각 밴드에 걸맞는 위치를 구분해서 할당해 주어야 한다.

위치나 밴드에 맞지 않으면 주파수 값이나 밴드 넓이 등이 잘못 출력되는 문제가 있다면 일반적으로 이곳에서 발생한다.

제조사에서 제공한 테이블 안에는 중복되는 조합이 굉장히 많이 분포되어있다. 이를 수작업으로 고르는 작업은 굉장히 답답한 일이다.

해당 작업을 자동으로 처리하는 코드의 일부이다.

```python
for x in range(len(cc_list)):
    cc_len = len(sheet[cc_list[x]])
    for i in range(3,cc_len):
        if sheet[cc_list[x]][i].value == None or sheet[cc_list[x]][i].value == 'Same':
            continue
        if sheet[res_list[x]][i].value != None:
            continue
        split_x = sheet[cc_list[x]][i].value.split('-')
        for j in range(i+1,cc_len):
            if sheet[cc_list[x]][j].value == None or sheet[res_list[x]][j].value != None or sheet[cc_list[x]][j].value
                continue
            split_y = sheet[cc_list[x]][j].value.split('-')
            cnt=0

            p= Counter(split_x)
            q= Counter(split_y)
            for key in p:
                for B_key in q:
                    if key == B_key:
                        if p[key]==q[B_key]:
                            cnt+=1

            if cnt == len(p):
                sheet[cc_list[x]][j].value = 'Same'
                sheet[cc_list[x]][j].fill = PatternFill(patternType='solid', fgColor=Color('ffff00'))
                sheet[cover_list[x]][j].value = sheet[index_list[x]][i].value
                sheet[cover_list[x]][j].fill = PatternFill(patternType='solid',fgColor=Color('ffff00'))
```

해당 코드를 통해서 각 조합이 상위 조합에서 유일한지 확인하고 분류 하는 작업을 갖는다.

추가적으로 분류된 조합은 유일하기 때문에 해당값을 제외하고 남은 모든 부분은 중복 형태로 잘라내도록 한다.

*** 일괄적으로 처리하므로 입력값에 실수가 없도록 한다. ***

같은 CC내에서 중복을 처리 하고 나면 다른 CC간 cover가 이루어진다. 예를 들어 3CC 내에서 2CC와 커버가 되었는데 같은 값이 5CC에서도 cover된다면 가장 큰 바운더리인 5CC가 할당되는것이 바로 그것이다.

다음 코드를 통해서 해당 기능을 구현한다.

```
for se_n in range(len(cc_list)-1):#2cc부터
    for ta_n in range(se_n+1,len(cc_list)):#끝cc까지
        se = cc_list[se_n]
        ta = cc_list[ta_n]
        r_se = rev_list[se_n]
        r_ta = rev_list[ta_n]

        for i in range(st,index_len):#첫 cc의 출발 범위
            if sheet[se][i].value == None:
                continue
            if sheet[se][i].value == 'Same':
                continue

            split_x = sheet[se][i].value.split('-')#있으면 하나씩

            for j in range(st,index_len):
                if sheet[ta][j].value == None:
                    continue
                if sheet[ta][j].value == 'Same':
                    continue

                if sheet[r_ta][j].value != 'No':
                    split_y = sheet[ta][j].value.split('-')
                    cnt=0

                    p= Counter(split_x)
                    q= Counter(split_y)
                    for key in p:
                        for B_key in q:
                            if key == B_key:
                                if p[key]<=q[B_key]:
                                    cnt+=1
```

값이 중복되지 않게 메모리를 나누어 처리 하고 결과가 올바르게 할당될수 있도록 이전에 작업 되었던 중복 조합은 체크 하지 않고 넘어가게 한다.

몇가지 의존성이 존재 하고 해당 값들이 혼선 되지 않게 한다.

```
                elif sheet[r_ta][j].value =='No' and sheet[r_se][i].value=='No':
                    split_y = sheet[ta][j].value.split('-')
                    if split_x[0]==split_y[0]:
                        split_x.pop(0)
                        split_y.pop(0)
                        p=len(split_x)
                        cnt = 0
                        for x in split_x:
                            if x in split_y:
                                cnt+=1
                        if cnt==p:
                            if sheet[se][i].value not in ok_list:
                                ok_list.append(sheet[se][i].value)
                                sheet[cover_list[se_n]][i].value=sheet[index_list[ta_n]][j].value
                                sheet[cover_list[se_n]][i].fill = PatternFill(patternType='solid', fgColor=Color('ff000
                                sheet[cc_list[se_n]][i].fill = PatternFill(patternType='solid', fgColor=Color('ff0000')
                                ch_cnt+=1
```

```python
for x in no_list:
    if x =='//':
        continue
    split_text = x.split('-')
    set_text=list(set(split_text))

    for p in set_text:
        min_data = list(p)
        if 'B' in min_data or 'C' in min_data:
            split_text.append(p)
        elif 'D' in min_data:
            split_text.append(p)
            split_text.append(p)
        elif 'E' in min_data:
            split_text.append(p)
            split_text.append(p)
            split_text.append(p)

    while(len(split_text)!=4):
        split_text.append(' ')

    for p in range(len(set_text)):
        tar_table['B'][CA_cnt].value = x
        tar_table['C'][CA_cnt].value = set_text[p]
        num=0
        split_text_cpy = copy.deepcopy(split_text)
        for de in range(len(split_text_cpy)):
            if set_text[p]==split_text_cpy[de]:
                del split_text_cpy[de]
                break

        for q in range(len(split_text_cpy)):
            if  num==0:
                tar_table['D'][CA_cnt].value = split_text_cpy[q]
            elif num==1:
                tar_table['E'][CA_cnt].value = split_text_cpy[q]
            elif num==2:
                tar_table['F'][CA_cnt].value = split_text_cpy[q]
            num+=1
```

새로운 틀을 구성해 새 파일을 만들고 다운로드 할수 있어야 한다.

그러므로 새로운 폼에 겹치지 않고 오류 없게 각 탭을 돌면서 다른 부분을 처리하도록 한다.

결과 정리본

제조사에서 제공한 기본 리스트를 입력한후 프로그램을 동작 시키면 다음과 같은 결과가 출력된다.

Index.0	CC.0	estriction.	Cover.0	Reverse.0	Index.1	CC.1	estriction.	Cover.1	Reverse.1	Index.2	CC.2	estriction.	Cover.2	Reverse.2	Index.3	CC.3	estriction.	Cover.3	Reverse.3
Index	2CC	Restrictior	Cover	Reverse	Index	3CC	Restrictior	Cover	Reverse	Index	4CC	Restrictior	Cover	Reverse	Index	5CC	Restrictior	Cover	Reverse
2CC #1	2A-2A			Yes	3CC #1	2A-2A-5A			Yes	4CC #1	2A-2A-5A-30A			Yes	5CC #1	2A-2A-46D	346 SCC only		No
2CC #2	2A-4A			Yes	3CC #2	2A-2A-12A			Yes	4CC #2	2A-2A-5A-66A			Yes	5CC #2	A-5B-30A-66A			Yes
2CC #3	2A-5A			Yes	3CC #3	2A-2A-13A			Yes	4CC #3	A-2A-12A-66A			Yes	5CC #3	A-5B-66A-66A			Yes
2CC #4	2A-12A			Yes	3CC #4	2A-2A-29A	329 SCC only		Yes	4CC #4	A-2A-13A-66A			Yes	5CC #4	2A-46D-66A	346 SCC only		Yes
2CC #5	2A-13A			Yes	3CC #5	2A-2A-30A			Yes	4CC #5	A-2A-29A-30	329 SCC only		Yes	5CC #5	B-30A-66A-66A			Yes
2CC #6	2A-29A	329 SCC only		Yes	3CC #6	2A-2A-66A			Yes	4CC #6	A-2A-66A-66A			Yes	5CC #6	41C-41D			Yes
2CC #7	2A-30A			Yes	3CC #7	2A-4A-4A			Yes	4CC #7	A-5A-30A-66A			Yes	5CC #7	66A-66A-46D	346 SCC only		No
2CC #8	2A-46A	346 SCC only		Yes	3CC #8	2A-4A-5A			Yes	4CC #8	A-5A-66A-66A			Yes					
2CC #9	2A-66A			Yes	3CC #9	2A-4A-13A			Yes	4CC #9	2A-5A-66B			Yes					
2CC #10	4A-4A			Yes	3CC #10	2A-5A-30A			Yes	4CC #10	2A-5A-66C			Yes					
2CC #11	4A-5A			Yes	3CC #11	2A-5A-66A			Yes	4CC #11	2A-5B-30A			Yes					
2CC #12	4A-13A			Yes	3CC #12	2A-5B			Yes	4CC #12	2A-5B-66A			Yes					
2CC #13	4A-46A	346 SCC only		Yes	3CC #13	2A-12A-30A			Yes	4CC #13	A-12A-30A-66A			Yes					
2CC #14	5A-30A			Yes	3CC #14	2A-12A-66A			Yes	4CC #14	A-12A-66A-66A			Yes					
2CC #15	5A-66A			Yes	3CC #15	2A-13A-66A			Yes	4CC #15	A-13A-66A-66A			Yes					
2CC #16	5B			Yes	3CC #16	2A-29A-30A	329 SCC only		Yes	4CC #16	2A-13A-66B			Yes					
2CC #17	12A-30A			Yes	3CC #17	2A-29A-66A	329 SCC only		Yes	4CC #17	2A-13A-66C			Yes					
2CC #18	12A-66A			No	3CC #18	2A-30A-66A			Yes	4CC #18	A-29A-30A-66	329 SCC only		Yes					
2CC #19	30A-29A	329 SCC only		Yes	3CC #19	2A-46A-46A	346 SCC only		Yes	4CC #19	2A-46A-46C	346 SCC only		Yes					
2CC #20	30A-66A			Yes	3CC #20	2A-46A-66A	346 SCC only		Yes	4CC #20	2A-46C-66A	346 SCC only		Yes					
2CC #21	13A-46A	346 SCC only		Yes	3CC #21	2A-46C	346 SCC only		Yes	4CC #21	2A-46D	346 SCC only		Yes					

또한 분류 이후에 같은 커버인지 확인 하는 시트도 생성된다.

2CC #19	30A-29A	329 SCC onl	4CC #5	Yes	3CC #19	2A-46A-46A	346 SCC only		Yes	4CC #19	2A-46A-46C	346 SCC only		Yes
2CC #20	30A-66A		5CC #2	Yes	3CC #20	2A-46A-66A	346 SCC only		Yes	4CC #20	2A-46C-66A	346 SCC only		Yes
2CC #21	13A-46A	346 SCC only		Yes	3CC #21	2A-46C	346 SCC onl	4CC #19	Yes	4CC #21	2A-46D	346 SCC onl	5CC #1	Yes
2CC #22	13A-66A		4CC #4	Yes	3CC #22	2A-66A-66A		5CC #3	Yes	4CC #22	4A-46A-46C	346 SCC only		Yes
2CC #23	25A-25A		3CC #40	Yes	3CC #23	2A-66B		4CC #9	Yes	4CC #23	4A-46D	346 SCC only		Yes
2CC #24	25A-26A		3CC #40	Yes	3CC #24	2A-66C		4CC #10	Yes	4CC #24	A-30A-66A-66A			Yes
2CC #25	25A-41A			Yes	3CC #25	4A-4A-5A			Yes	4CC #25	5B-30A-66A		5CC #2	Yes
2CC #26	26A-41A			Yes	3CC #26	4A-4A-13A			Yes	4CC #26	5B-66A-66A		5CC #3	Yes
2CC #27	41A-41A			Yes	3CC #27	4A-46A-46A	346 SCC only		Yes	4CC #27	Same		4CC #26	Yes
2CC #28	41C		5CC #6	Yes	3CC #28	4A-46C	346 SCC onl	4CC #22	Yes	4CC #28	A-30A-66A-66A			Yes
2CC #29	66A-29A	329 SCC onl	4CC #18	Yes	3CC #29	5A-30A-66A		4CC #7	Yes	4CC #29	13A-46D	346 SCC only		Yes
2CC #30	66A-46A	346 SCC onl	4CC #34	Yes	3CC #30	5A-66A-66A		4CC #8	Yes	4CC #30	25A-41D			Yes
2CC #31	66A-66A		5CC #3	Yes	3CC #31	5A-66B		4CC #9	Yes	4CC #31	41A-41D			Yes

보라색 : 다른CC로 인해 cover 가능한 CA 조합

녹색 : 같은CC간의 same인 CA 조합

베이지색 : 실제 측정해야할 CA 조합

그후 작성된 자동 시트에 결과 파일을 넣고 프로그램을 실행시킨다.

DL CA output power results

E-UTRA CA congiguration (BCS)	Bands					UL				
	PCC	SCC1	SCC2	SCC3	SCC4	PCC				
	1st	2nd	3rd	4th	5th	Mode	BW (MHz)	Channel	Freq (MHz)	RB/Offset
13A-46A	13A	46A				16QAM	10	23230	782	25/0
	13A	46A				16QAM	10	23230	782	25/0
	13A	46A				16QAM	10	23230	782	25/0
	13A	46A				16QAM	10	23230	782	25/0
25A-41A	25A	41A				16QAM	20	26590	1905	1/0
26A-41A	26A	41A				16QAM	15	26865	831.5	1/0
2A-4A-4A	2A	4A	4A			16QAM	20	19100	1900	1/49
	4A	2A	4A			16QAM	20	20175	1732.5	1/0
2A-4A-5A	2A	4A	5A			16QAM	20	19100	1900	1/49
	4A	2A	5A			16QAM	20	20175	1732.5	1/0
	5A	2A	4A			16QAM	10	20525	836.5	1/0
2A-4A-13A	13A	2A	4A			16QAM	10	23230	782	25/0
	2A	4A	13A			16QAM	20	19100	1900	1/49
	4A	2A	13A			16QAM	20	20175	1732.5	1/0
2A-46A-46A	2A	46A	46A			16QAM	20	19100	1900	1/49
2A-46A-66A	2A	46A	66A			16QAM	20	19100	1900	1/49
	2A	46A	66A			16QAM	20	19100	1900	1/49
	2A	46A	66A			16QAM	20	19100	1900	1/49
	2A	46A	66A			16QAM	20	19100	1900	1/49
	66A	2A	46A			16QAM	20	132072	1720	1/0

이렇게 업링크를 넣어주고 프로그램을 실행하면 결과가 출력되게 된다.

이전 Power Project 에서는 간단하게 측정 값 보정에 그치는 작업에 불과했지만 해당 프로젝트 에서는 결과치를 여러번 받아서 재 구성하고 다시 정렬 하고 값을 털어내고 연산하고 high냐 low냐에 따라서 가장 멀리 떨어진 위치에 값을 배치 하면서도 연속이냐 아니냐를 고려해 주어야 해서 조금더 머리를 많이 쓰는 작업에 해당했다.

실제 프로그램 실행시간은 최대 5분 이내이며

결과 수치는 은지과장님과 성훈과장님을 비롯한 사원 분들에게 검증을 받아 오류가 없는한 100%에 가까운 정확도를 보여준다.

7 Project.3 : G-EMC Team Web Base : Task Scheduler

해당 프로젝트는 G-EMC내부에서 절차의 시각화와 함께 성적서 자동화를 목적으로 구성된 프로젝트이다. 사내 와이파이망을 기반으로 동작하는 CAN(Company Area Network)를 구성하고 팀 내부 자체 웹페이지를 탑재하여 팀원 전원이 공유하는 팀 스케쥴러를 만든다.

만들어진 스케쥴러 기반 데이터 베이스는 현재 진행된 업무 진행 상황과 결과를 공유해 일괄적으로 성적서를 출력할수있게 설계한다.

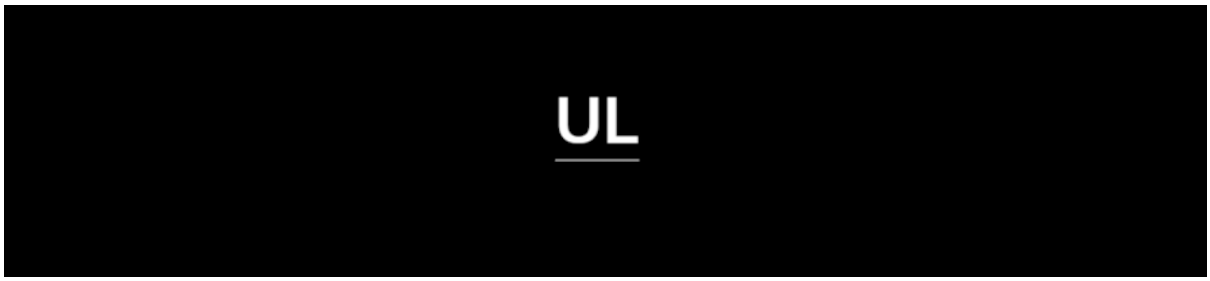
또한 팀 책임자의 업무가 매우 바빠 주기적으로 스케쥴러를 확인하지 못할것을 고려해 업무 진행 정도 예측 프로그램을 머신러닝 기반 회귀분석 모델을 설계해 시험 완료 일자를 예측하고 해당 날짜만료 2~3일전 팀 책임자에게 일괄 메일 발송시스템을 구축한다.

또한 팀 내부 프로젝트 문제는 보안적으로도 굉장히 민감한 문제를 가지고 있어 일괄적으로 웹 페이지 형태로 보여주는데 보안 이슈를 갖는다.

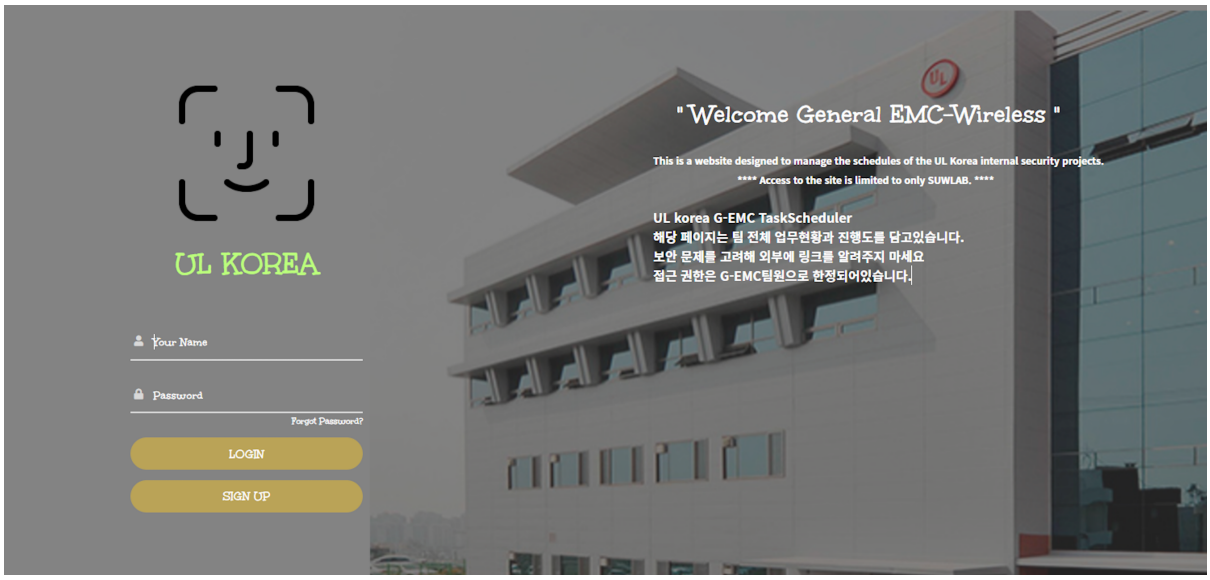
따라서 G-EMC 팀원 얼굴을 인식하는 딥러닝 모델을 구출해 웹페이지에 탑재하고 확인된 인원만 데이터 베이스에 접근 할수 있게 한다.

구현 화면 샘플

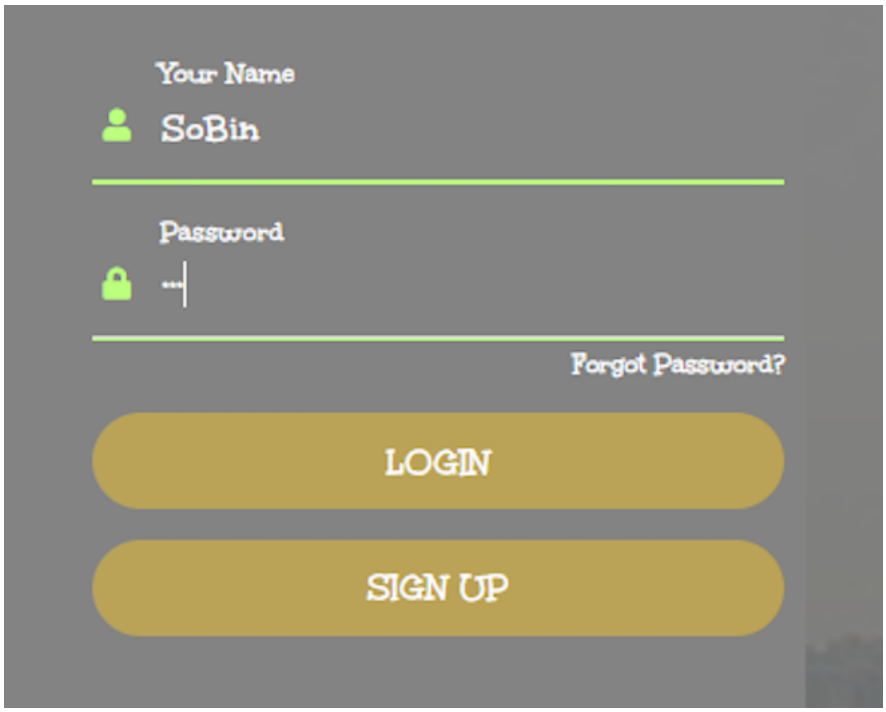
인트로 대기화면



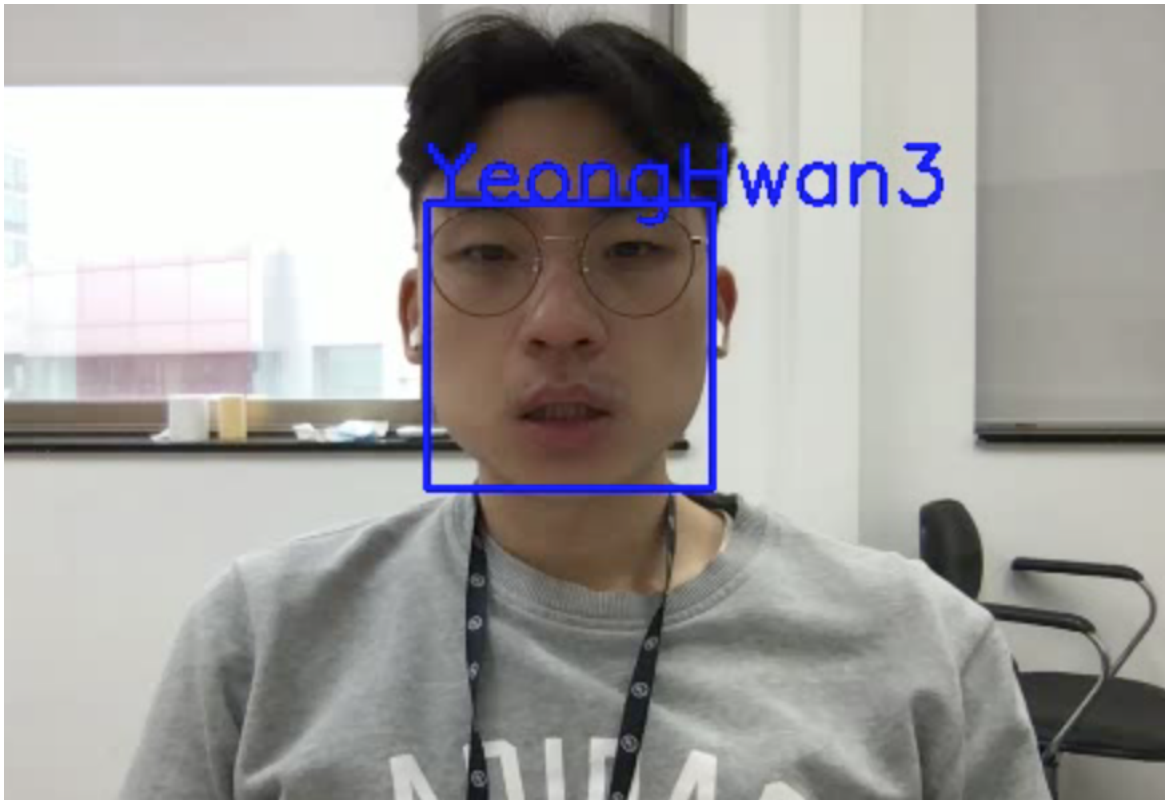
메인 로그인 화면



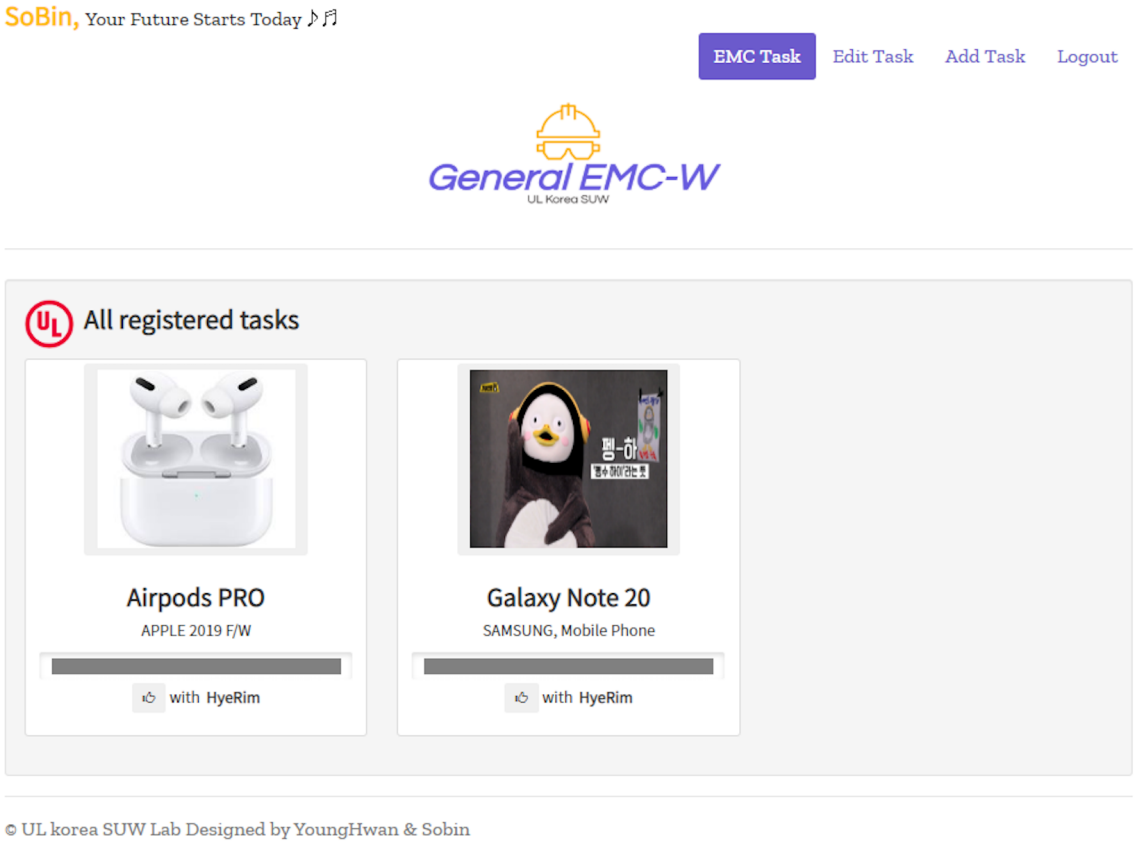
로그인예시

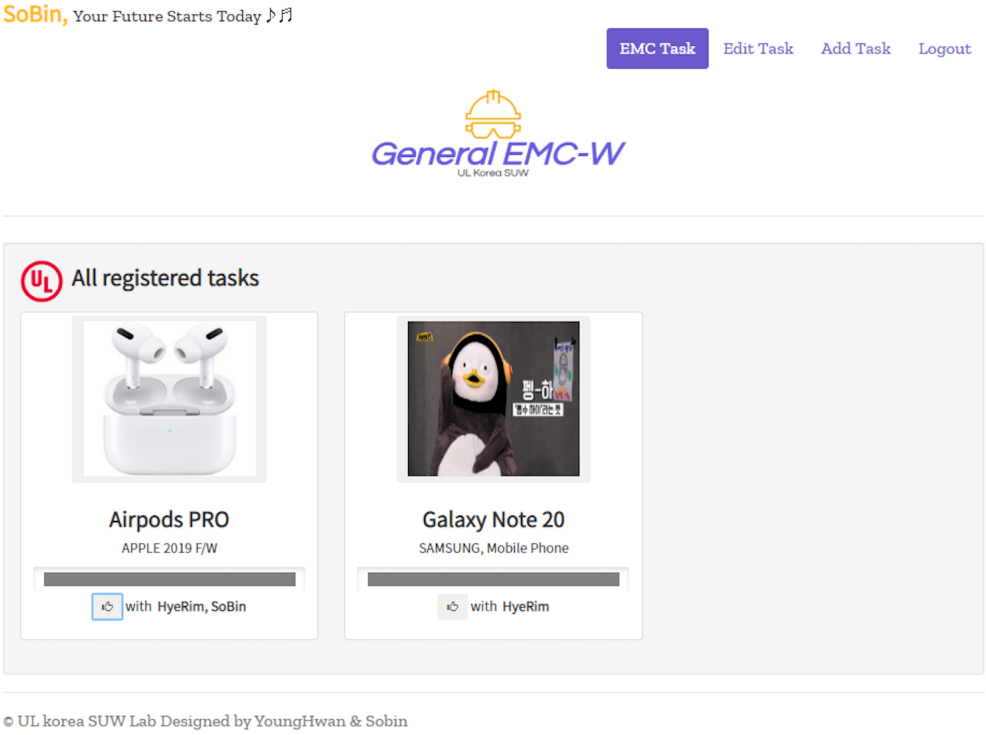


얼굴인식 예시



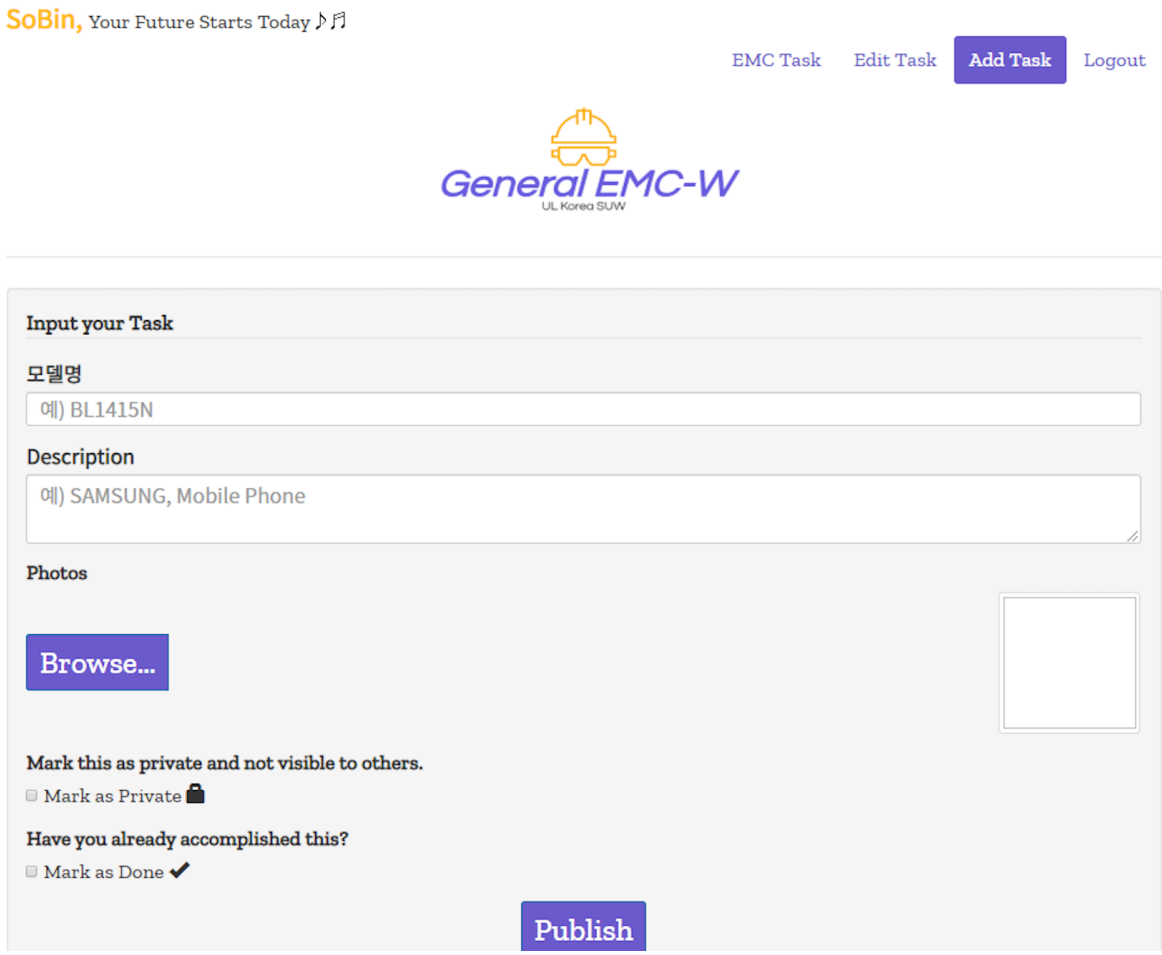
로그인 성공 후 메인화면


SoBin, Your Future Starts Today ♪♬
EMC Task
Edit Task
Add Task
Logout
General EMC-W
UL Korea SUW
All registered tasks
Airpods PRO
APPLE 2019 F/W
with HyeRim
Galaxy Note 20
SAMSUNG, Mobile Phone
with HyeRim
© UL korea SUW Lab Designed by YoungHwan & Sobin


SoBin, Your Future Starts Today ♪♬
EMC Task
Edit Task
Add Task
Logout
General EMC-W
UL Korea SUW
All registered tasks
Airpods PRO
APPLE 2019 F/W
with HyeRim, SoBin
Galaxy Note 20
SAMSUNG, Mobile Phone
with HyeRim
© UL korea SUW Lab Designed by YoungHwan & Sobin

해당 화면에서 전체 업무 진행 현황을 파악 할수 있다. 또한 해당 시험을 함께 진행하고 있는 인원의 목록을 아래 함께 하기 버튼을 통해 할 수 있다. 구체적인 인원에게 연락하고 싶을때 진행 인원확인이 가능하다.

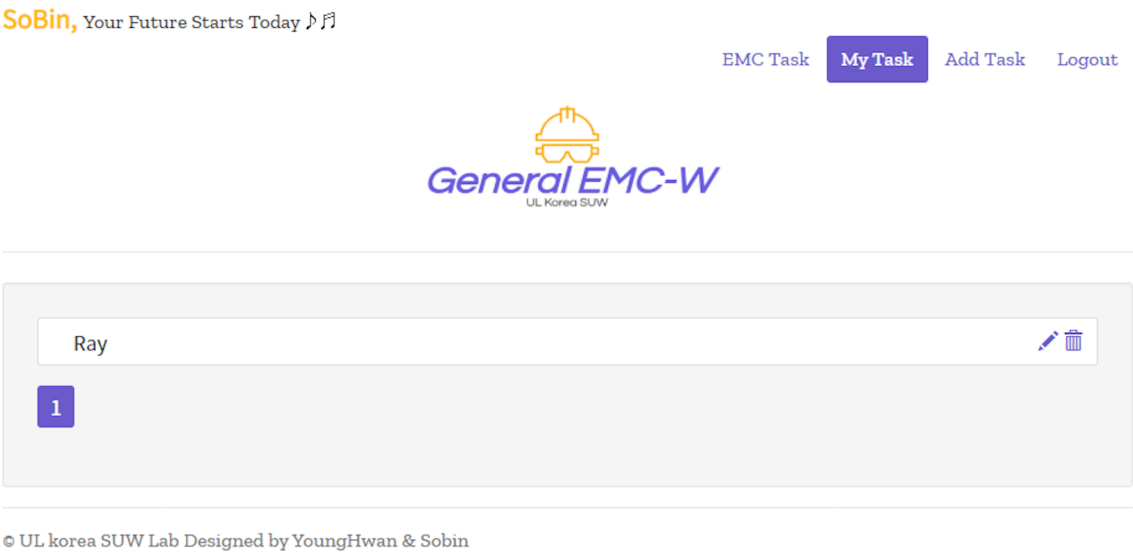
업무 추가



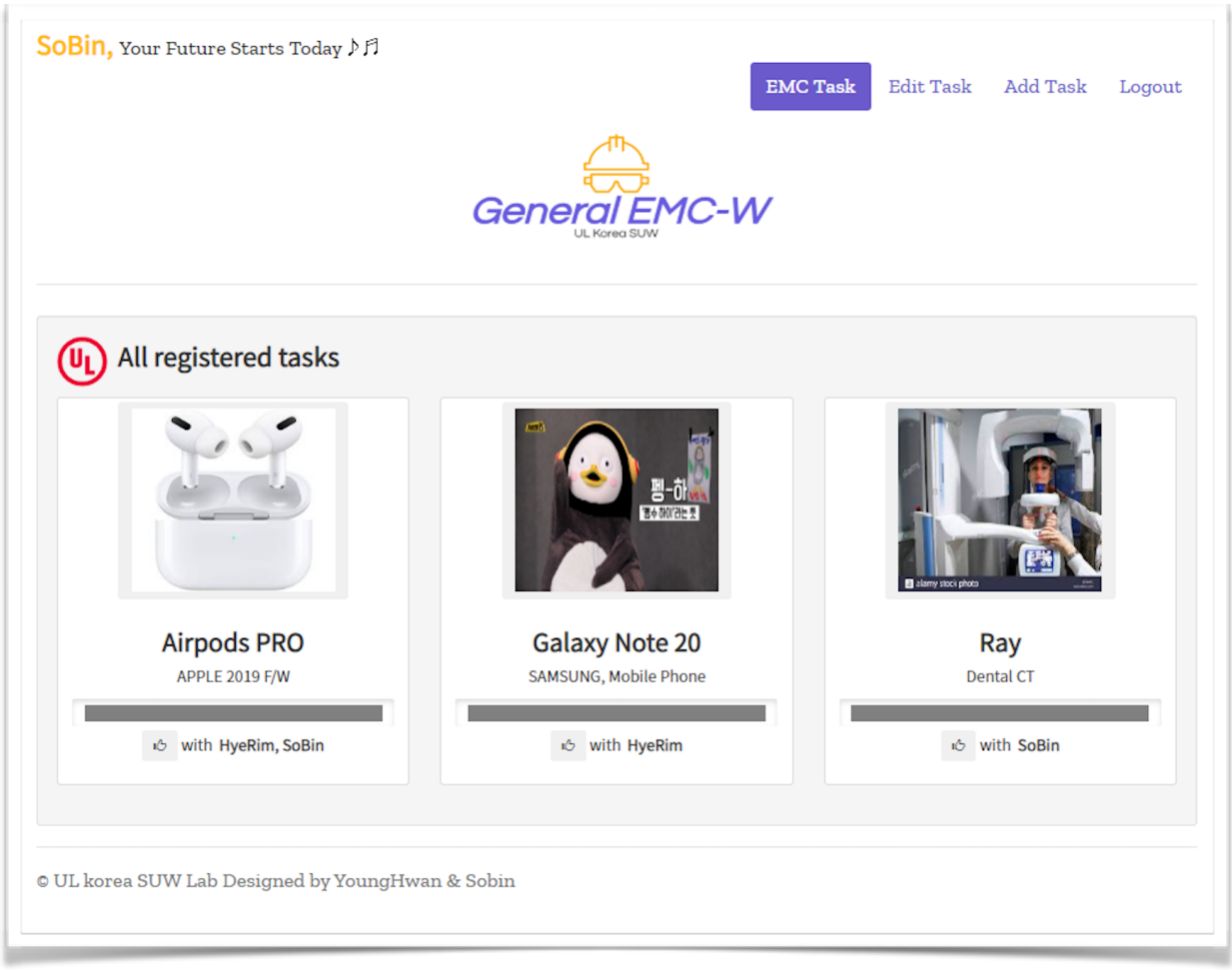
업무 추가 탭 이다.

타 사원과 공유 가능한 내용이 아니라면 아래 private를 추가 하게 하여 업무의 생산성과 개인 프로젝트 간 구분을 짓게 한다.

제품에 관한 내용을 작성하고 publish를 통해서 웹에 등록한다.



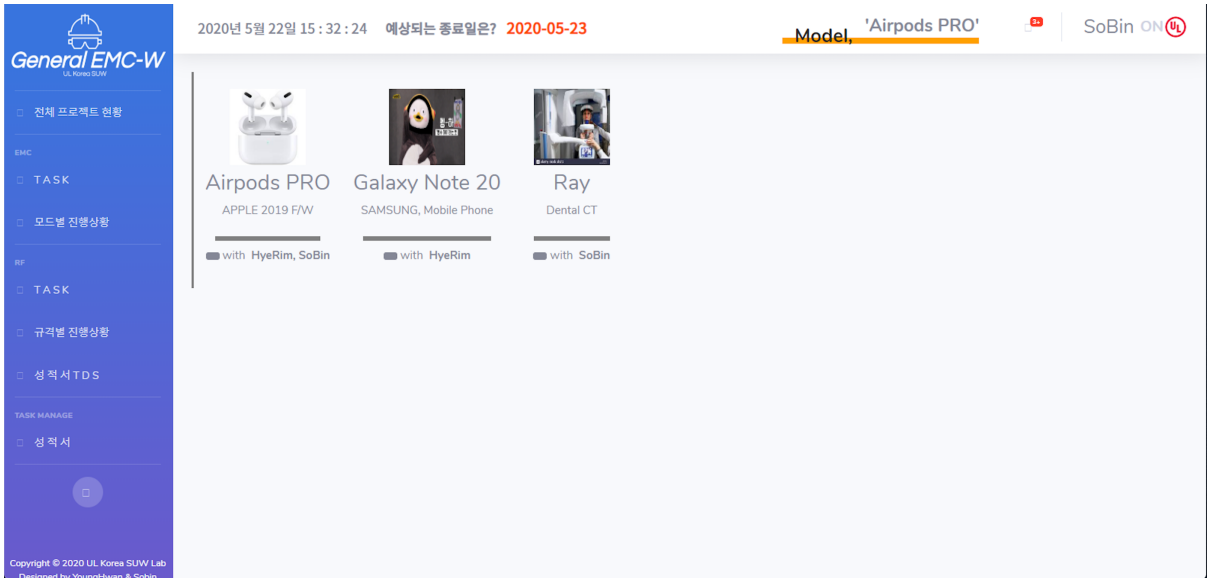
등록된 업무는 개인화 탭에 보여지고 private를 체크 하지 않았다면 모든 사원에게 공개된다.



성공적으로 업무 탭에 추가된 모습이다.

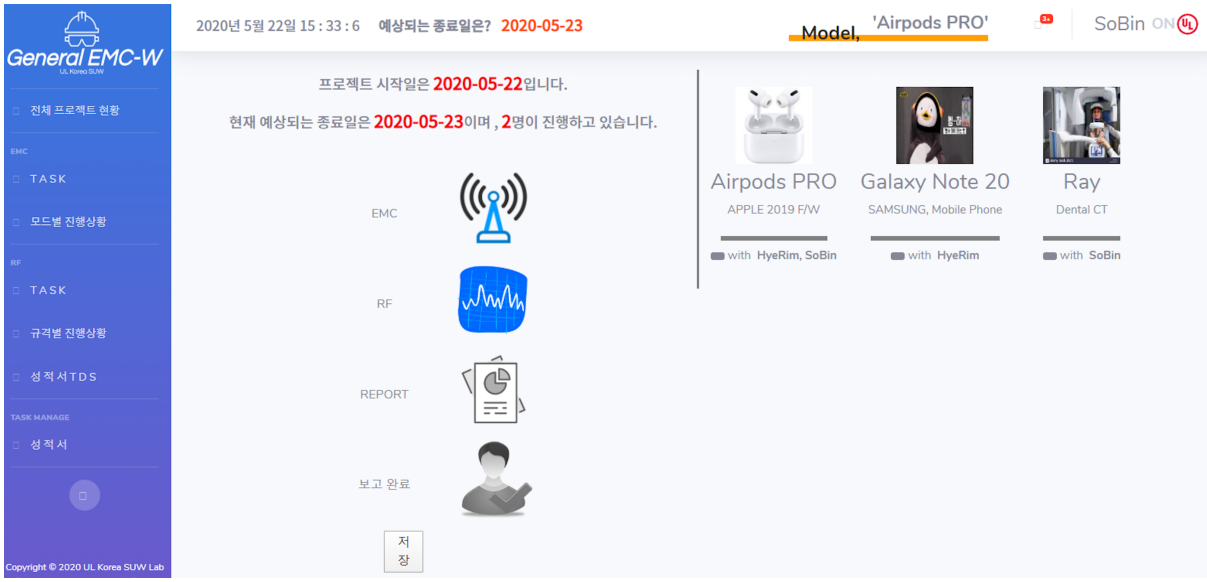
추가 된 업무는 팀원과 공유되어 일괄적으로 확인 가능하다.

추가된 해당 탭을 클릭 하면 전체 프로젝트 현황을 볼수 있다.



전체 프로세스를 관리 하는 형태의 홈으로 구성되어있고

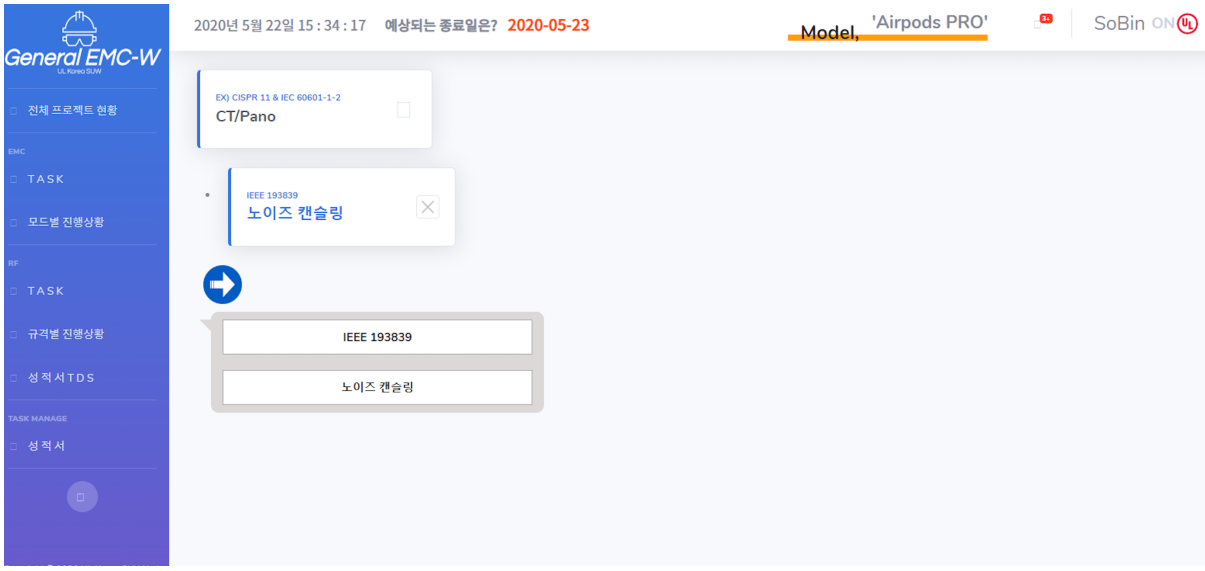
이 페이지 레이아웃 내에서 전체 진행도를 확인하고 성적서를 포함한 내용을 일괄적으로 처리 가능하다.



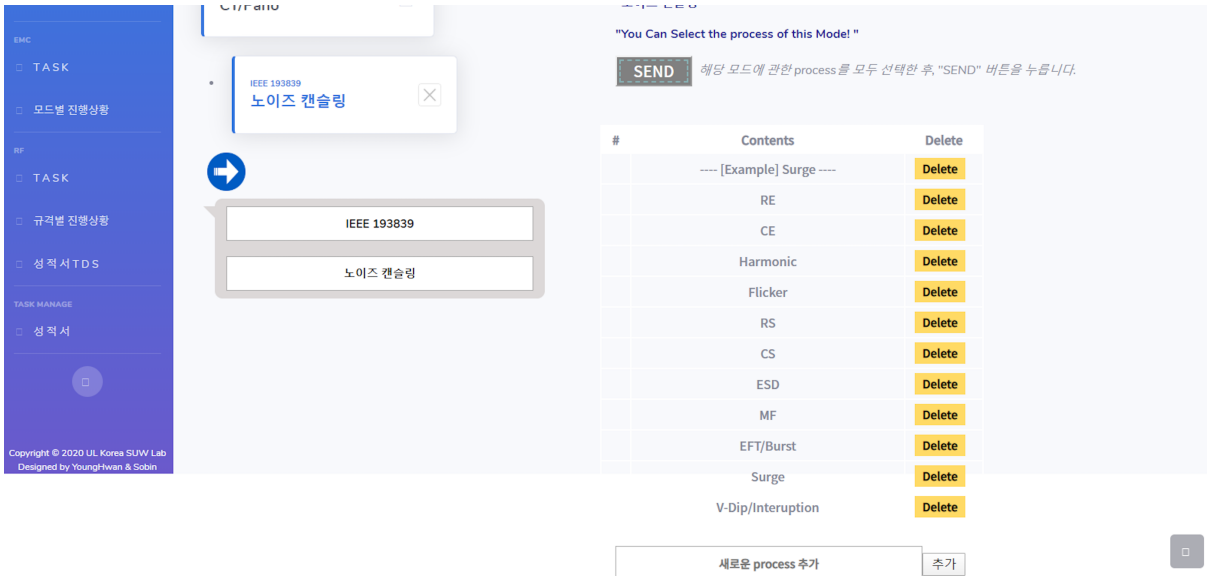
상단의 날짜를 기준으로 예상되는 종료일을 산정할수 있다.

제품 디버깅이나 다른 이슈에 관해서는 일괄적으로 처리 하기 어렵지만 어느정도의 진척도와 결과물을 확인 할수 있는 지표로서 작용할 것이라고 생각된다.

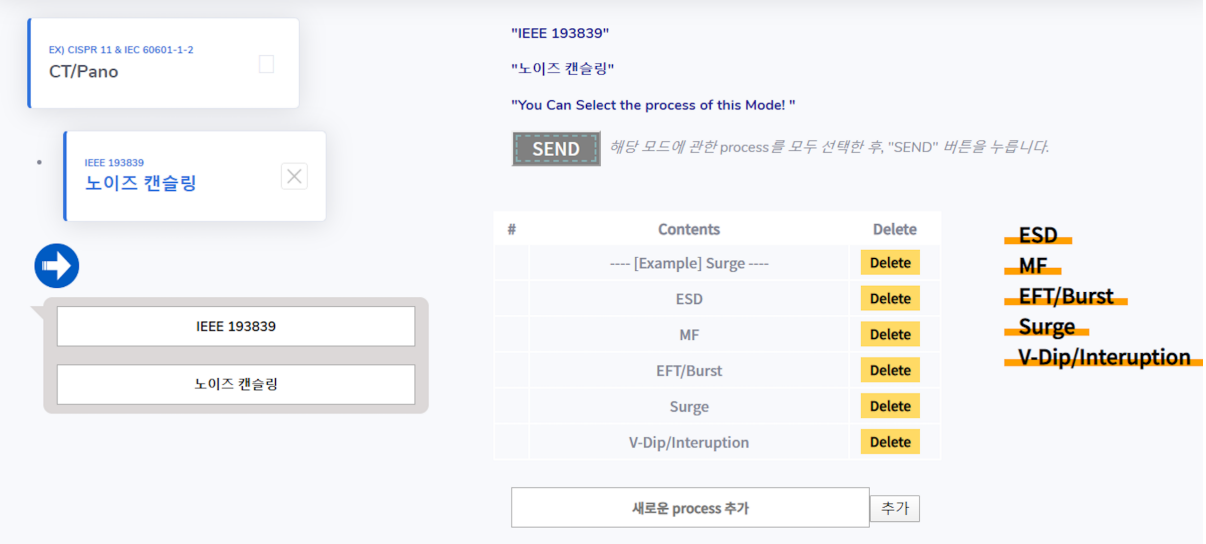
모드와 규격을 설정하는 탭이다. 모드와 규격을 설정하고 시험이 진행되었는지 등을 체크 하게 되면 자동으로 날짜를 계산할수 있게끔 했다.



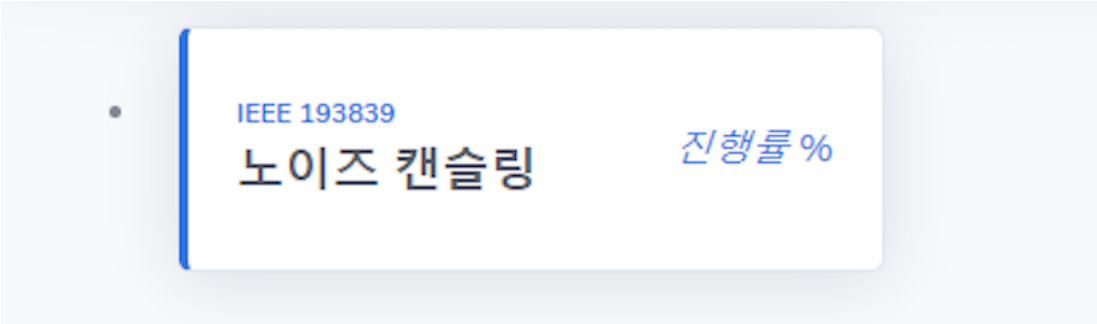
위처럼 모드를 추가 하고 여러개의 모드가 합쳐져 만들어 지는 경우에는 몇가지 다른 모드를 선택적으로 추가할수 있게끔 설계했다.



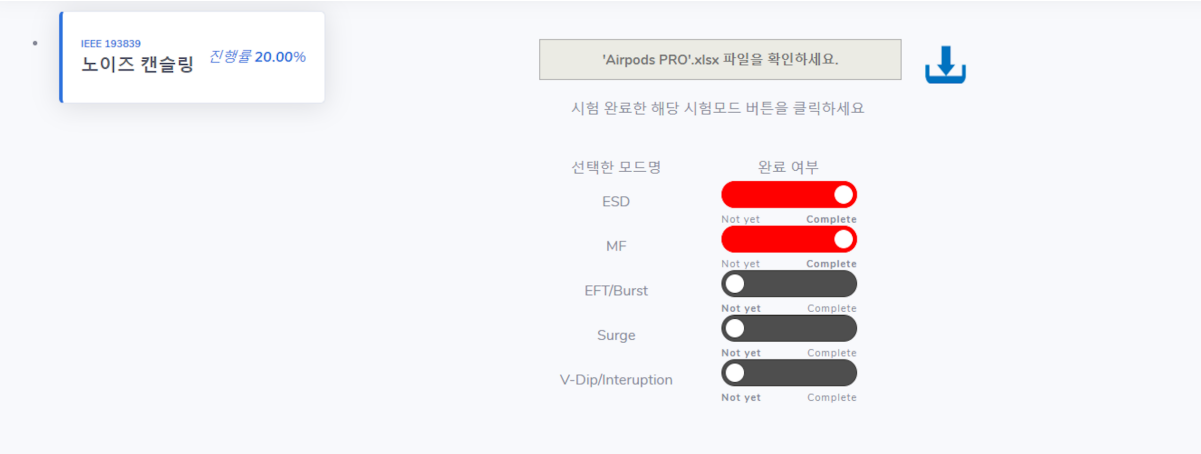
위처럼 일괄적인 시험케이스 추가도 가능하고 목록에 있지 않은 문제를 해결해야 하는 상황도 고려해 시험 케이스를 유동적으로 추가할 수 있게 한다.



위처럼 해당 제품에 걸맞는 테스트 플랜을 직접 선정 가능하고 일괄적으로 엑셀로 다운로드 해 문서화 작업도 가능하게 한다.

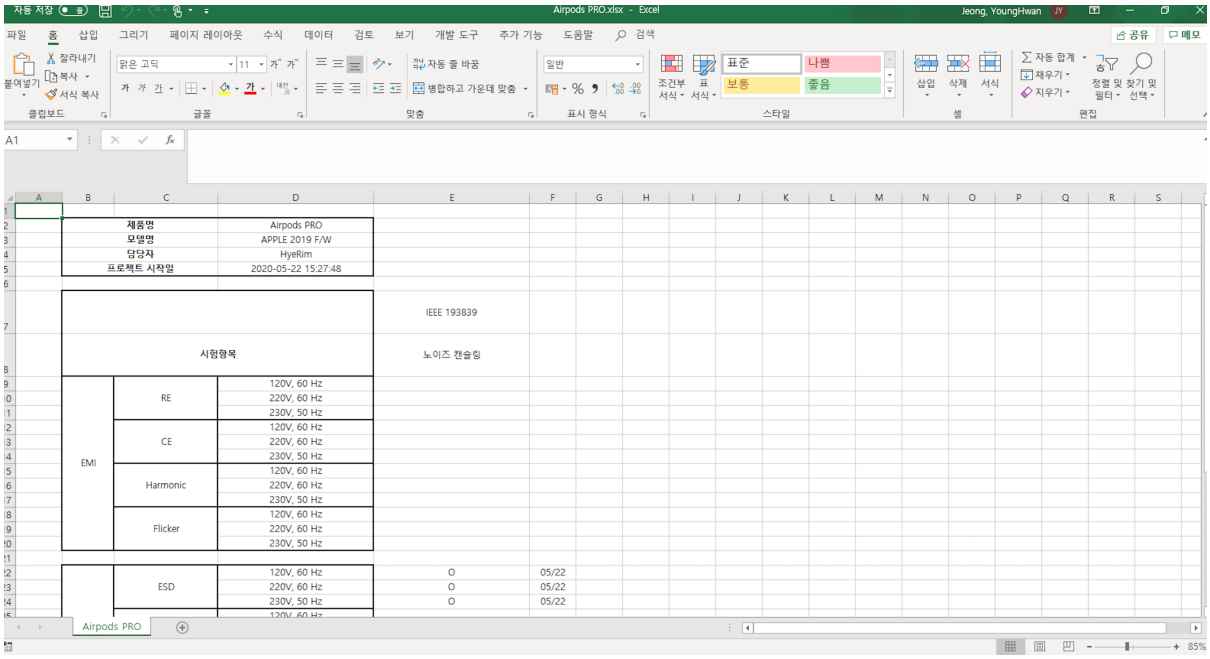


최종적으로 선택을 완료 하고 나면 다음과 같이 진행률을 확인할수 있게끔 객체가 생성된다.



만들어진 플랜의 경우에는 테스트가 완료된 경우 다음처럼 버튼을 활성화 시켜서 진행률을 표시 할수 있게 한다.

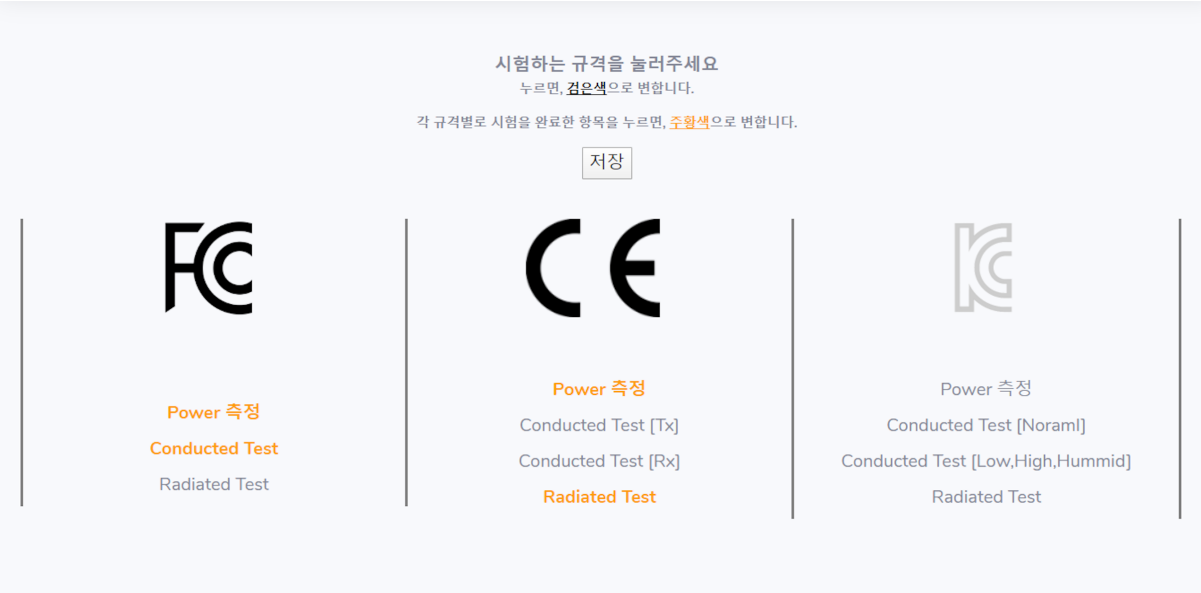
표시된 진행률은 모든 화면에 공시됨으로서 모든 사원들이 업무 진행 정도 파악에 용이 하게 한다.



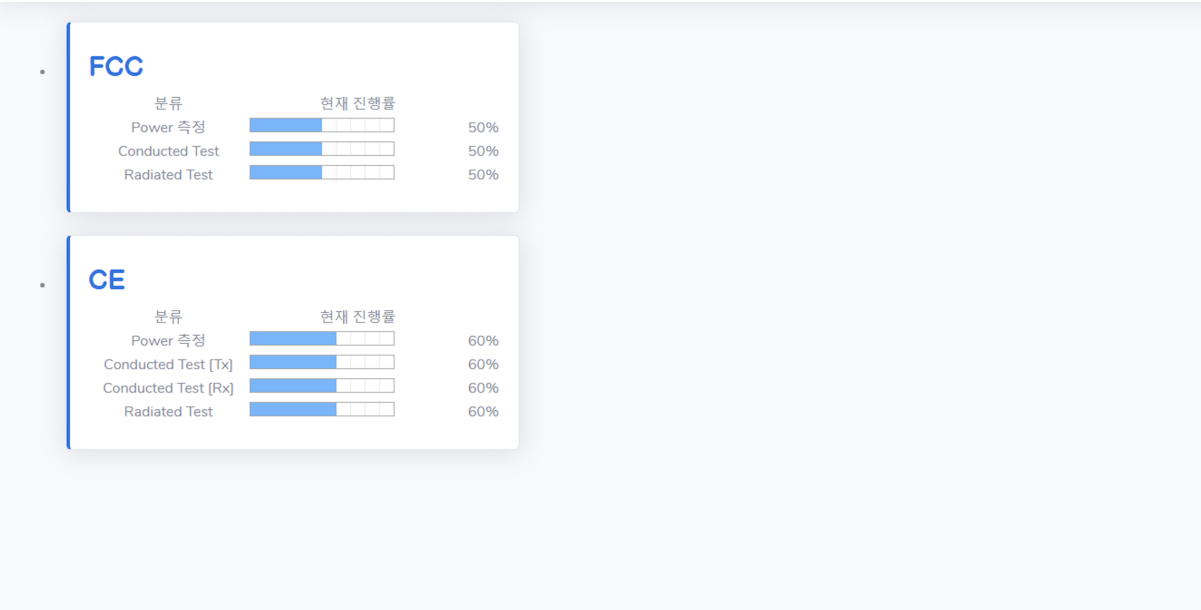
제품명	Airpods PRO
모델명	APPLE 2019 F/W
담당자	HyeRim
프로젝트 시작일	2020-05-22 15:27:48

시험항목			IEEE 193839	
			노이즈 캔슬링	
EMI	RE	120V, 60 Hz		
		220V, 60 Hz		
		230V, 50 Hz		
	CE	120V, 60 Hz		
		220V, 60 Hz		
		230V, 50 Hz		
	Harmonic	120V, 60 Hz		
		220V, 60 Hz		
		230V, 50 Hz		
	Flicker	120V, 60 Hz		
		220V, 60 Hz		
		230V, 50 Hz		
	ESD	120V, 60 Hz	O	05/22
		220V, 60 Hz	O	05/22
		230V, 50 Hz	O	05/22

그리고 웹 페이지에서 자동으로 이런 포멧이 생성되고 다운로드 될 수 있게 했다. 문서화 하기도 용이해 사원분들의 편의성을 도모했다.



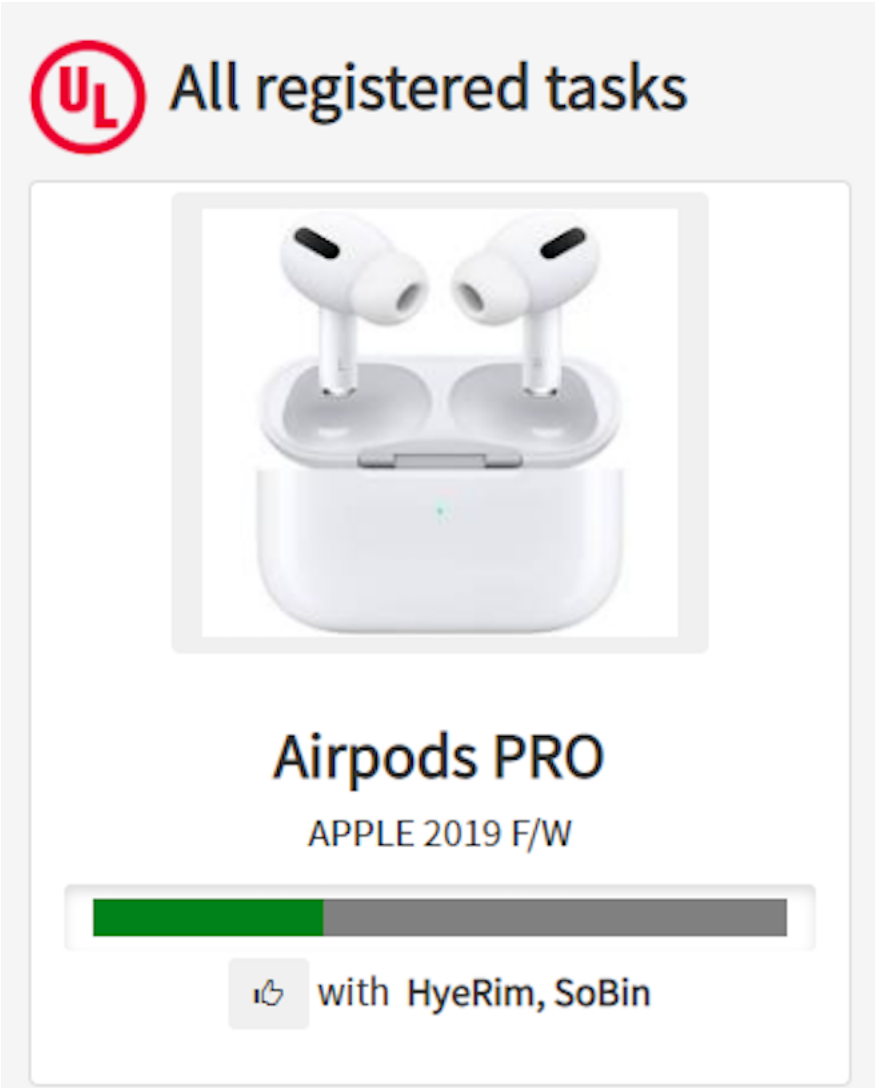
RF는 다음과 같다. 각 규격과 시험 항목에 관해서는 제품마다 달라 가장 큰틀을 잡고 테스트 항복을 선택하게끔 정리한다.



선택하게 되면 다음과 같은 창을 호출할수 있다. 테스트 항목별 데이터를 일목요연하게 정리해서 시각화 할수 있게 했다.

시각화된 자료는 자동적으로 보고서나 성적서 형태로 저장되어 업무의 편의성을 도모한다.

성적서 자동화 탭은 다음과 같다. 전체 성적서를 만들어야 할 파일을 올리고 저장을 하면 일괄적으로 처리해 저장할수 있게 한다.



저장된 진행률에 관한 정보는 위처럼 시각화 되어 얼마나 되고 있고 얼마나 남았는지를 직관적으로 알수 있게 구성했다.

추가적으로 해당 업무의 진행정도를 업무의 책임자가 단번에 알수 있게 하는것은 위 프로젝트의 핵심 목표이다.

따라서 책임자가 업무의 진행정도를 파악하는것을 굳이 현 페이지에 들어오지 않고도 알수 있게 하기 위해 책임자에게 지정 기일 3일전에 일괄적으로 메일을 발송하게 한다.

서버 시간을 기준으로 3일전에 일괄적으로 발송 처리해 책임자가 화면을 주기적으로 보지 않더라도 해당 내용을 보고 아 이정도 되고 있구나 하는 내용을 감각적으로 파악 할수있게 한다.

업무가 획일화 되어 있지 않고 모든 업무에 맞춤형 시스템을 만든다는것은 현실적으로 불가해 큰 틀에서 진행률에 맞추어 정도를 파악하는데서 그쳤다.

하지만 추가적으로 인원이 보충되고 업무 관련해서 많은 분들이 관심을 가지고 협조가 가능한 상황이 온다면 더욱 논리적이고 의미 있는 시스템 개발을 할수 있을것이라 생각한다.

8 Reference

[Deep Learning from skratch - sati kogi 2017 / 한빛미디어]

[SW developing Process - 특허청 2012 / 특허청]

[Wes Mckinney,Julie Steele and Meghan Blanchette, Teresa Exley,"Python for Data Analysis" proceedings 2012.Oct O'REILLY pp.111-152]

[Vivian Siahaan-Rismon Hasiholan Sianipar "LEARNING PyQt5",first-edition proceedings 2019 MY_SQL ,pp 89-221]

[L. Benini, A. Bogliolo, S. Cavallucci and B. Ricco, "Monitoring system activity for OS-directed dynamic power management," Proceedings. 1998 International Symposium on Low Power Electronics and Design (IEEE Cat. No.98TH8379), Monterey, CA, USA, 1998, pp. 185-190.]

[T. E. Oliphant, "Python for Scientific Computing," in Computing in Science & Engineering, vol. 9, no. 3, pp. 10-20, May-June 2007.]

[python cookbook 3rd_edition - O'Relly 2015 /한빛미디어]

[Deep learning with python - Francois Chollet / 한빛 미디어]

[Unified, Real-Time Object Detection - Joseph Redmon / Google AI team]

[H. Cai, Q. Wu, T. Corradi, and P. Hall. The cross- depiction problem: Computer vision algorithms for recog- nising objects in artwork and in photographs. arXiv preprint arXiv:1505.00110, 2015.]

[M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual ob- ject classes challenge: A

retrospective. International Journal of Computer Vision, 111(1):98-136, Jan. 2015.]

[S. Gidaris and N. Komodakis. Object detection via a multi- region & semantic segmentation-aware CNN model. CoRR, abs/1505.01749, 2015.]

[P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localiza- tion and detection using convolutional networks. CoRR, abs/1312.6229, 2013.]

[J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. International journal of computer vision, 104(2):154-171, 2013.]

도움을 주신분들

Power 자동화 관련 - 양은희 사원님

CA 자동화 관련 - 김성훈 과장님 , 양은희 사원님

G-EMC task Scheduler 관련 - 최창영부장님 , 송혜림사원님

감사합니다.