|  |
| --- |
| [project Final report] |
| 최종 보고서 |
|  |
|  |
|  |

|  |  |  |
| --- | --- | --- |
| **Team name** | R.U.Robot | |
| **Professor** | 손태식 | |
| **Project Manager** | 김형호 | 김형호 |
| **Team Member** | 이상민 | 이상민 |
| 조영준 | 조영준 |
| 송연주 | 송연주 |
|  |  |

The table of contents

[**Abstract** 3](#_Toc10982341)

[**1.1** **Background / Trend** 4](#_Toc10982342)

[**1.2** **Motivation of this work** 4](#_Toc10982343)

[**1.3** **Contribution of this work** 4](#_Toc10982344)

[**2.** **Related work** 5](#_Toc10982345)

[**2.1.1** **uncaptcha** 5](#_Toc10982346)

[**3.** **Proposed System** 6](#_Toc10982347)

[**3.1** **System description** 6](#_Toc10982348)

[**3.1.1** **System Context Diagram** 6](#_Toc10982349)

[**3.1.2** **Class Diagram** 7](#_Toc10982350)

[**3.1.3** **Subsystem Model** 13](#_Toc10982351)

[**3.1.4** **Sequence Diagram** 14](#_Toc10982352)

[**3.2.1** **Crawler** 17](#_Toc10982353)

[**3.2.2** **Preprocessor** 17](#_Toc10982354)

[**3.2.3** **Machine learning** 17](#_Toc10982355)

[**3.2.4** **GUI** 18](#_Toc10982356)

[**3.3** **Algorithm or Method** 18](#_Toc10982357)

[**3.3.1** **MyWindow** 18](#_Toc10982358)

[**3.3.2** **LogInDialog** 19](#_Toc10982359)

[**3.3.4** **CaptchaAudio** 24](#_Toc10982360)

[**3.3.5** **CaptchaImage** 26](#_Toc10982361)

[**3.3.6** **LearnedMachine** 28](#_Toc10982362)

[**3.3.7** **ImageMachine(LearnedMachine)** 29](#_Toc10982363)

[**3.3.8** **SoundMachine(LearnedMachine)** 29](#_Toc10982364)

[**4.** **Implementation Results** 31](#_Toc10982365)

[**4.1** **Test environment** 31](#_Toc10982366)

[**4.2** **Test scenario** 31](#_Toc10982367)

[**4.2.1** **Test scenario by functions** 32](#_Toc10982368)

[**4.2.2** **Total test scenario** 32](#_Toc10982369)

[**4.3** **Results** 32](#_Toc10982370)

[**5.** **Conclusion** 33](#_Toc10982371)

[**5.1** **역할 분담** 33](#_Toc10982372)

[**5.2** **개발 일정** 33](#_Toc10982373)

[**5.3** **개발 비용** 34](#_Toc10982374)

[**5.4** **위험요소 및 해결방법** 34](#_Toc10982375)

[**5.5** **사회적 영향** 34](#_Toc10982376)

[**5.6** **발전 방향** 34](#_Toc10982377)

[**References** 36](#_Toc10982378)

# **Abstract**

A CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a type of challenge-response test used in computing to determine whether the user is human. However, as a machine learning evolves, this system is being threatened by image recognition and audio recognition. The purpose of Project is showing that advances in artificial intelligence, especially machine learning, can easily make CAPTCHA vulnerable. This project examines the reliability of Captcha by an Anti-Captcha automation AI (Artificial Intelligence) programme. . The project is expected to demonstrate dangers of Artificial Intelligence and also gives vision on the improvement of Captcha.

The system components and functions are as follows. The system of the project consists of crawler, Pre-processor and machine learning model. First, the crawler, a technique employed to extract large amounts of data from websites, extracts Chrome's web information and CAPTCHA data. And It also automatically enters user information and predicts results. Second, the Pre-processor processes the CAPTCHA data so that it can be used by machine learning. processing performed on raw data to prepare it for machine learning, to delete noise and edit image and audio data and turn data in readable numpy array. Third, the Machine Learning model analysis downloaded image and audio array based on train set and predicts processed CAPTCHA data.

For Gmarket and Auction, the following process has been fully implemented. Saving the user's information, tracking the user's web surfing, extracting the data from the web page if CAPTCHA window is opened, pre-processing extracted data and appling to machine learning models. Finally, the value predicted through the model is automatically entered with the user information.

The future direction of development of our project is as follows. The project is needed to be updated learning model to match the various CAPTCHA data and to increase accuracy. Also, valid sites will be extended.

1. **Introduction**

## **Background / Trend**

Captcha(Completely Automated Public Turing test to tell Computers and Humans Apart, 이하 캡챠)는 HIP기술의 일종으로, 어떠한 사용자가 실제 사람인지 컴퓨터 프로그램인지를 구별하기 위해 사용되는 방법이다. 사람은 구별할 수 있지만 컴퓨터는 구별하기 힘들게 의도적으로 비틀거나 덧칠한 그림이나 숫자나 텍스트를 읽어주는 오디오를 주고 그 내용을 물어보는 방법이 사용된다. 현재 로그인시나 본인인증시에 널리 사용되고있다.

ELKSZ

[그림 01] 캡챠 이미지

하지만 이러한 사용자인증방식은 인공지능의 발전이 대두됨에 따라 신뢰성을 위협받고 있다. 2018년 8월 한국정보화진흥원에서 발표한 인공지능 악용에 따른 위협과 대응 방안 리포트에 의하면, 인공지능은 범죄 목적으로 개발, 악용되는 사례가 늘어나고 있다. 특히 인공지능은 범죄성 사이버 공격 프로세스 작업 자동화가 가능하고, 인간행위와 인간인지능력을 모방할 수 있다는 점에서 디지털보안 분야에서 높은 파괴력을 보일 수 있다.

## **Motivation of this work**

본 프로젝트는 위와 같은 문제점을 인식하고 국내의 본인인증 서비스의 신뢰성을 조사하여 사용자로부터 보다 안전한 본인인증 서비스를 제공하고자 시작하게 되었다. 본 프로젝트에서는 인공지능을 활용한 안티캡챠(Anti-captcha) 프로그램을 개발하여 국내 사이트의 캡챠 기능을 활용하는 휴대폰 본인인증 서비스의 신뢰성을 검증한다.

## **Contribution of this work**

본 프로젝트에서는 인공지능을 활용한 안티캡챠(Anti-captcha) 프로그램을 개발하여 국내 사이트의 캡챠 기능을 활용하는 휴대폰 본인인증 서비스의 신뢰성을 검증한다. 이를 이용해 국내 사이트에서 사용되는 캡챠의 보안성을 검증해봄으로써 인공지능의 위험성을 알리고 국내 사이트의 본인인증체계의 개선방안을 제고하여 국내 캡챠 관련 연구에 시발점이 되기를 기대하고 있다. 또한, 마지막으로 캡챠의 신뢰도를 높일 수 있는 해결방안을 제시하여 본인인증 서비스를 한 단계 더 발전시키고 사용자가 국내 본인인증 서비스를 보다 안심하고 사용하는데 기여한다.

# **Related work**

* 본 프로젝트에 관련된 연구들에 대해서 정리합니다.
* 프로젝트 결과물과 비슷한 결과물이 있다면 그것의 장단점에 대해서 서술합니다.
* 본 프로젝트에 있어서 현실적 제한 요소(기술적 제약, 경제성, 안정성, 신뢰성, 미학, 윤리, 사회적 영향)에 대해서 기술합니다.

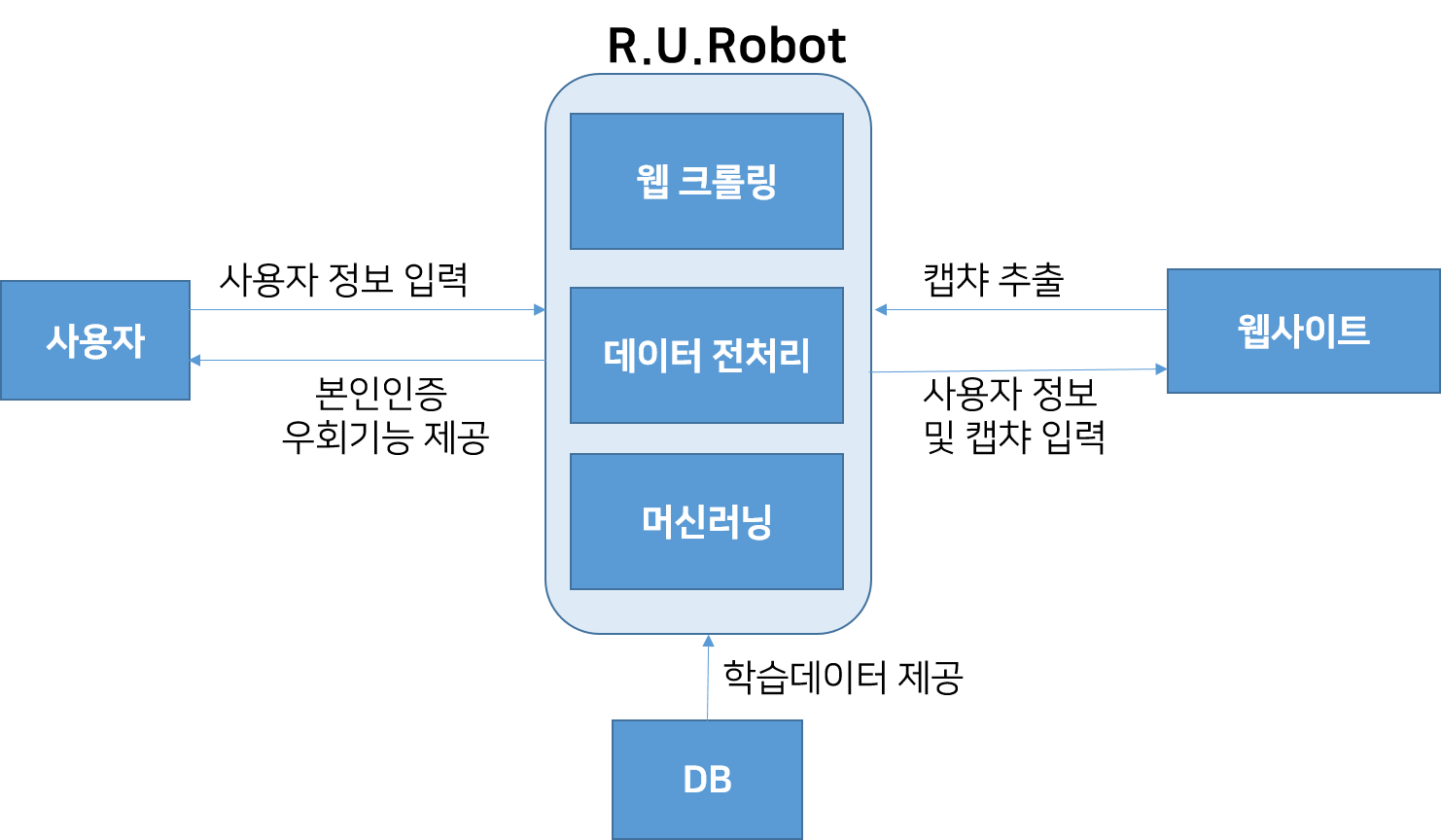
1. 캡챠의 기원
2. 캡챠의 발전
   1. 인비지블 리캡챠
   2. 이미지-텍스트 퓨전
3. 언캡챠
   * 1. **uncaptcha**

uncaptcha(이하 언캡챠)는 메릴랜드대학의 컴퓨터과학 연구원들에 의해 개발된 프로그램으로써 인공지능의 이미지/오디오 인식기능을 활용하여 구글의 recaptcha(이하 리캡챠) 혹은 그와 비슷한 보안 시스템을 무력화시키는 프로그램이다. 2017년 4월에 처음 언캡챠가 발표된 이후 구글은 리캡챠의 브라우저 정보 탐지 기능을 강화하였고, 음성의 단위를 ‘문장’으로 바꾸었다. 하지만 2018년 6월 언캡챠는 스크린 클릭커를 업데이트시켜 특정 픽셀 커서를 옮길 때 사람처럼 움직이도록 하는 기능을 추가하여 다시 한번 구글의 리캡챠를 무력화시키는데 성공했다. 또한, 언캡챠팀은 깃허브를 통해 개념증명용 코드(<https://github.com/ecthros/uncaptcha2>)를 공개하였다. 이후 구글은 2018년 10월 맞춤형 리스크 분석을 배경에서 실시하게 함으로써 사용자 경험을 향상시킨 리캡챠3를 발표했다.

# **Proposed System**

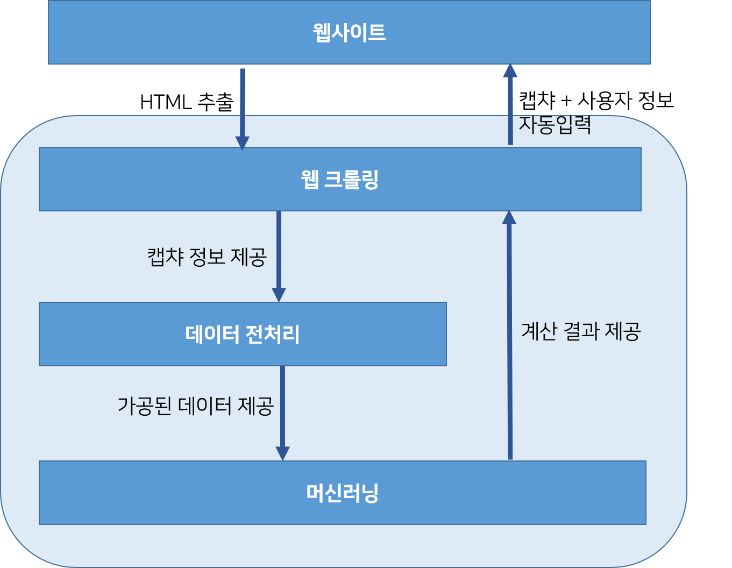
## **System description**

* + 1. **System Context Diagram**



[그림 02] System Context Diagram

본 프로그램의 전제 구조는 위 그림과 같다. 본 프로그램의 기능은 크게 웹 크롤링, 데이터 전처리, 머신러닝으로 나뉘어져 있으며, 사용자와 웹사이트 사이에서 인터렉션한다. 사용자가 프로그램을 사용할 때에는 먼저 사용자는 본 프로젝트의 실행 프로그램을 실행 시킨 후 사용자의 정보를 입력시킨다. 이후 프로그램을 통해 크롬을 실행하면 프로그램은 크롬 드라이버를 통해 HTML elements를 수집하고 수집된 정보를 통해 사용자가 본인인증 서비스에 접근했는지를 판단한다. 만약 사용자가 본인인증 서비스에 접근했다면 웹사이트로부터 캡챠 이미지, 오디오 데이터를 추출하고 데이터 전처리 과정을 거친 데이터와 기존에 저장되어있던 학습 데이터와의 비교를 통해 예측 값을 도출한다. 도출해낸 값과 사용자정보를 웹사이트에 자동으로 입력한다. 이 과정은 사용자가 크롬을 종료할 때까지 반복된다.

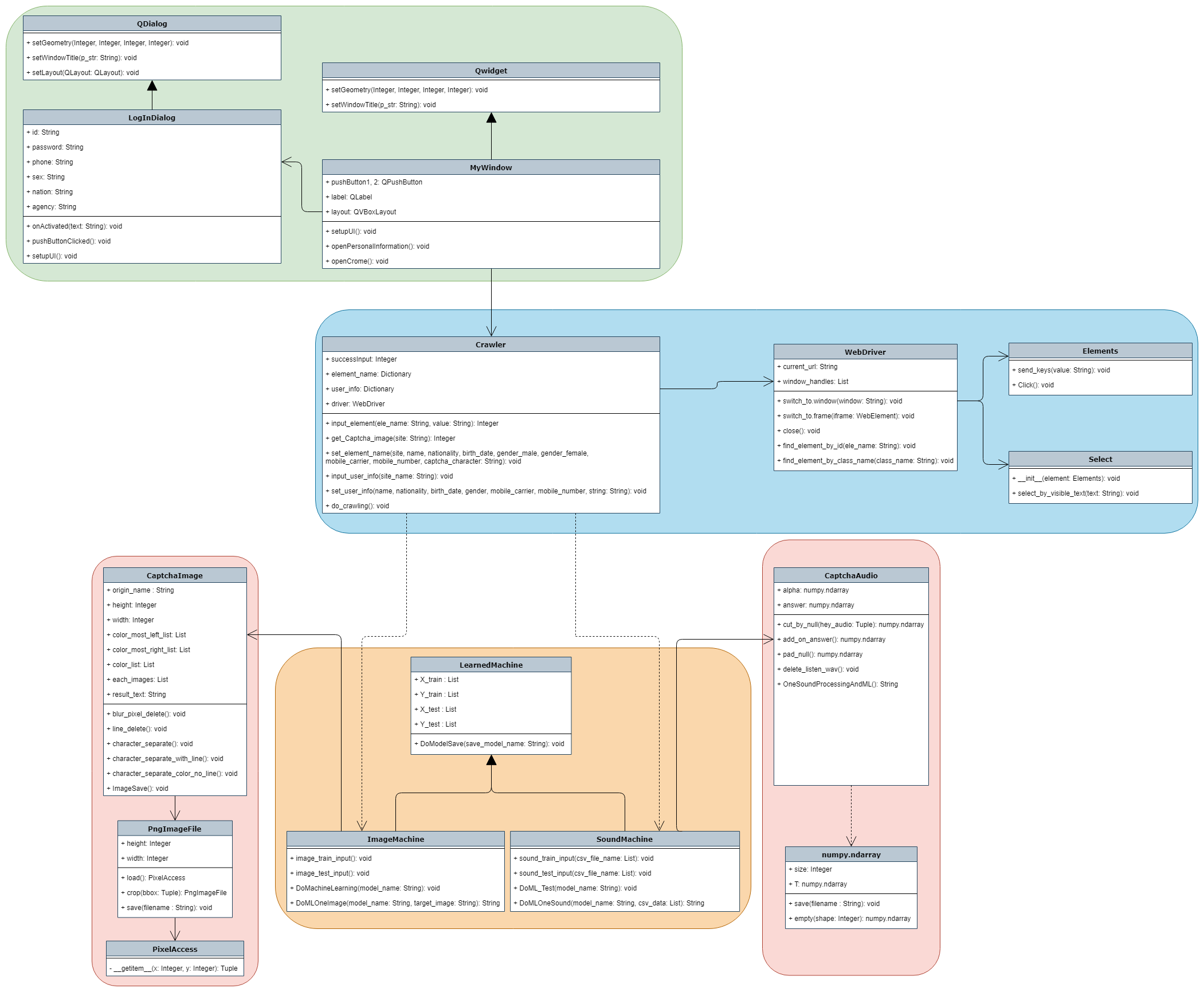


[그림 03] 프로그램 내부 구조 context diagram

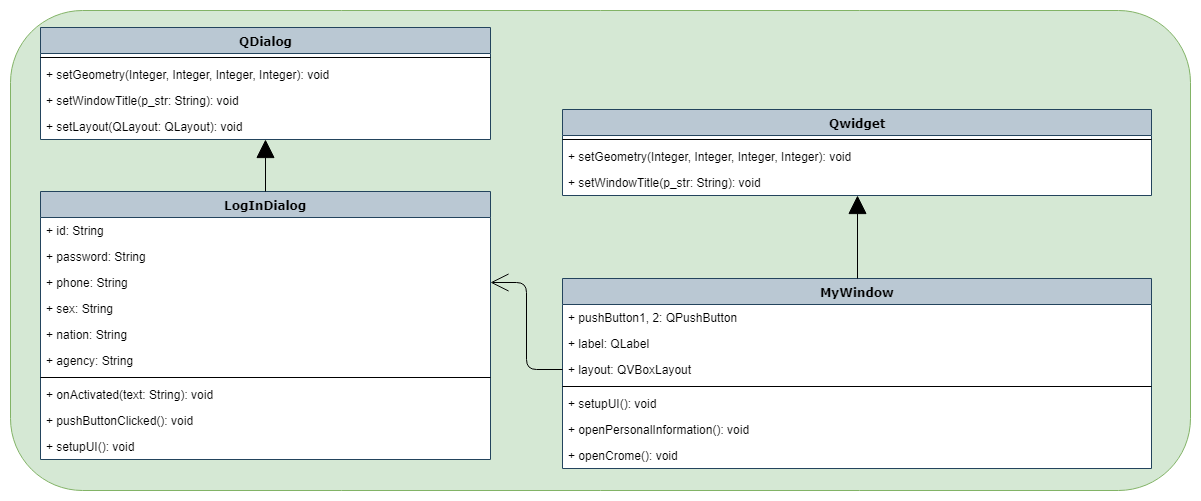
프로그램의 내부 구조에 대해 더 자세하게 살펴보자면, 프로젝트는 크게 웹 크롤링, 데이터 전처리, 머신러닝 세가지 주요 기능을 가지고 있으며, 사용자가 본인인증 사이트에 접근하였을 때 웹 크롤링, 데이터 전처리, 머신러닝, 그리고 다시 웹 크롤링의 순서로 동작한다. 먼저, 사용자가 본인인증 사이트에 접근하였을 때 HTML elements를 탐색하여 캡챠 이미지 데이터와 오디오 데이터를 추출한다. 추출한 데이터는 데이터 전처리로 넘겨지게 되는데 데이터 전처리 단계에서는 제공받은 데이터들의 노이즈를 제거하고 머신러닝이 접근할 수 있는 객체 형태로 가공한다. 이후 전처리된 데이터들은 머신러닝으로 넘겨지게 되고 머신러닝은 기존에 가지고 있던 학습된 데이터와 넘겨받은 데이터들의 비교를 통해 값을 예측한다. 이렇게 계산된 결과들은 다시 웹 크롤러에게 넘겨지게 되고 웹 크롤러는 넘겨받은 캡챠 값과 프로그램 동작 전에 사용자로부터 입력받은 사용자 정보를 웹페이지 양식에 맞게 자동 입력한다.

* + 1. **Class Diagram**

본 프로젝트의 실행 프로그램을 실행하면 MyWindow 클래스로부터 시작되며 사용자 정보 입력 창을 클릭하면 사용자 정보 입력을 받는 LogInDialog 클래스가 사용된다. 크롬 실행을 클릭하면 크롬의 내용을 수집, 처리 및 자동입력하는 Crawler 클래스가 사용되며 Crawler 클래스는 크롬의 내용을 수집하기 위해 WebDriver 클래스를 사용한다. Crawler 클래스는 캡챠 데이터의 예측 결과를 알아내기 위해 이미지의 경우 ImageMachine 클래스를, 사운드의 경우 SoundMachine 클래스를 이용한다. ImageMachine 클래스는 이미지를 머신러닝에 적용하기 위해 이미지를 전처리해주는 CaptchaImage 클래스를 이용하며 SoundMachine 클래스는 사운드를 머신러닝에 적용하기 위해 사운드를 전처리해주는 CaptchaAudio 클래스를 이용한다. 전체적인 클래스 다이어그램을 보여주는 다이어그램은 하기와 같다.

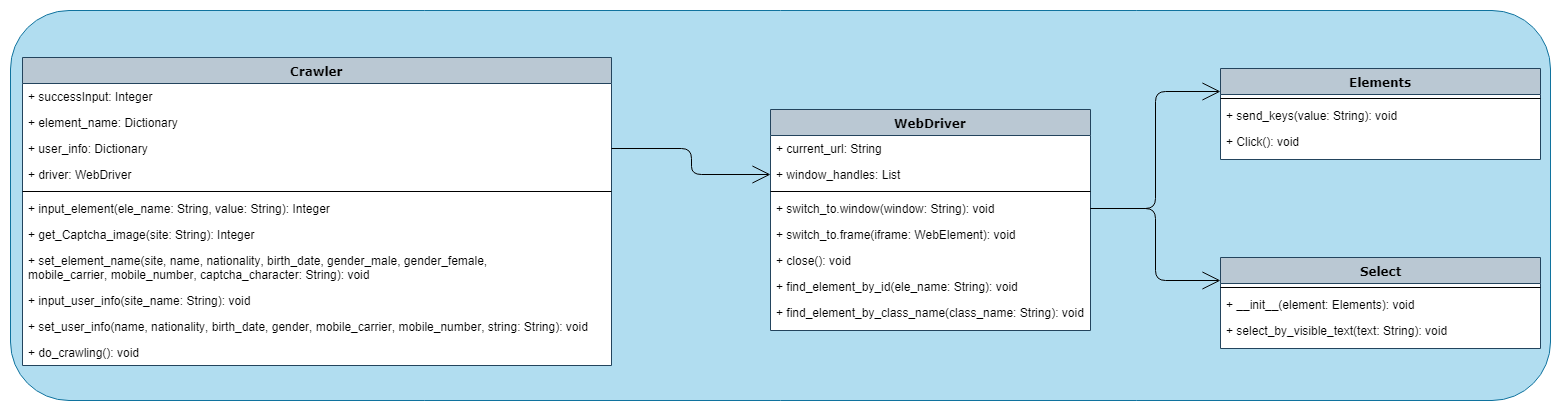


[그림 09] class diagram



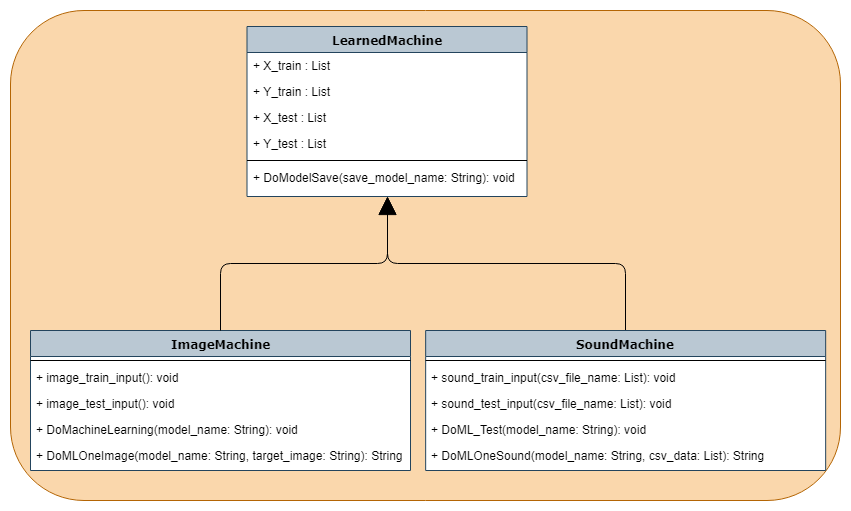
[그림 10] GUI class diagram

* MyWindow 클래스 : Qwidget의 자식 클래스로 메인창을 생성해주는 역할
  + 변수 설명
    - pushButton1, 2 : “개인정보 입력”, “크롬 열기” 버튼
    - label : 사용자가 입력한 정보를 메인창에서 표시해주는 라벨
    - layout : 메인창의 레이아웃 생성
  + 함수 설명
    - setupUI : 메인창을 생성
    - openPersonalInformation : 사용자가 입력한 개인정보를 불러와 메인창의 라벨에 복사하고 “test.txt”파일에 개인정보를 저장.
    - openChrome : “크롬 열기” 버튼이 클릭된 경우 크롤러 실행
* LogInDialog 클래스 : QDialog의 자식 클래스로 “개인정보 입력” 버튼이 클릭된 경우 새 창에서 개인정보를 입력받고 저장하는 역할
  + 변수 설명
    - id, password, phone, sex, nation, agency : 사용자의 정보를 의미한다.
  + 함수 설명
    - pushButtonClicked : 입력받은 사용자 정보를 개인정보 변수에 저장해줌으로써 메인창에 표시하고 크롤링에서 입력 가능하게 함
    - seupUI : 사용자 정보를 입력받는 팝업창 생성



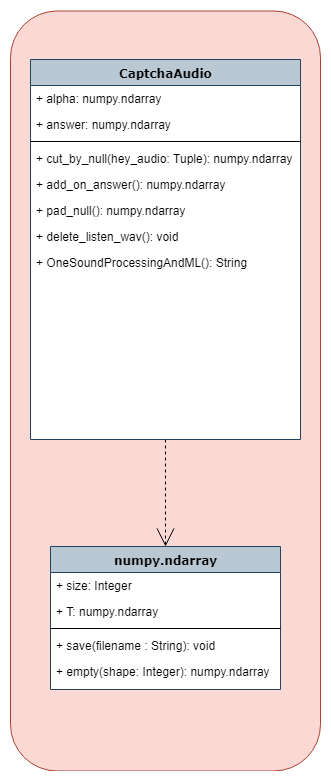
[그림 11] crawler class diagram

* Crawler 클래스 : 사용자가 크롬을 이용한 웹서핑 중 웹페이지 데이터를 수집 및 처리 후 자동입력하는 역할
  + 변수 설명
    - successInput : 사용자 정보 자동입력의 완수여부를 의미
    - element\_name : 캡챠 페이지별로 다른 웹 element의 태그를 나타냄
    - user\_info : 사용자 정보를 나타냄
    - driver : Webdriver의 객체로 웹페이지 데이터 수집과 자동입력에 사용
  + 함수 설명
    - input\_element : 현재 driver의 웹페이지에서 ele\_name에 해당하는 태그에 value 값을 입력
    - get\_Captcha\_image : site 종류에 따라 다른 HTML에 맞춰 캡챠 이미지와 음성 데이터를 다운받는 역할
    - set\_element\_name : element\_name의 값을 설정
    - input\_user\_info : 사용자 정보와 자동입력방지문자 예측결과를 크롬드라이버를 통해 입력하여 자동화
    - set\_user\_info : user\_info의 값을 설정
    - do\_crawling : 크롬이 종료될 때까지 반복하여 웹페이지 데이터 수집
* WebDriver 클래스 : Crawler 클래스에서 사용되며 크롤러와 크롬의 중간다리 역할
  + 변수 설명
    - current\_url : 현재 사용자가 서핑 중인 URL
    - window\_handles : 사용자가 크롬으로 열어둔 창들을 의미, 그 중 window\_handles[-1]은 현재 이용중인 창을 의미
  + 함수 설명
    - switch\_to.window : 파라미터로 할당된 윈도우로 driver를 switch
    - switch\_to.frame : 파라미터로 할당된 frame으로 driver를 switch
    - close : 웹드라이버를 종료
    - find\_element\_by\_id : 파라미터와 일치하는 id element 찾아냄
    - find\_element\_by\_class\_name : 파라미터와 일치하는 class name element 찾아냄



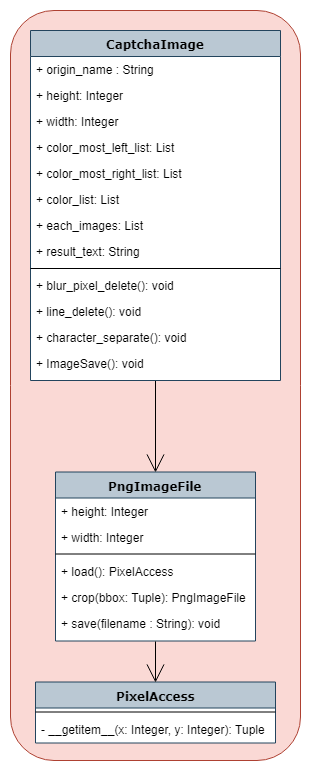
[그림 12] machine model class diagram

* LearnedMachine 클래스 : ImageMachine, SoundMachine의 부모 클래스로 머신러닝 모델이 작동하기 위해 필요한 train, test set을 설정하는 기능
  + 변수 설명
    - X\_train : 모델 생성시 학습 시킬 데이터들의 집합
    - Y\_train : 학습 시킬 데이터들의 정답 집합
    - X\_test : 학습된 모델에 적용시킬 데이터들의 집합
    - Y\_test : 학습된 모델에 적용시킬 데이터들의 정답 집합
  + 함수 설명
    - DoModelSave : 학습시킨 모델을 파라미터 값으로 저장함
* ImageMachine 클래스 : 전처리된 이미지 데이터를 입력하고 학습된 데이터를 적용하는 역할
  + 함수 설명
    - image\_train\_input : X\_train, Y\_train에 특정 폴더의 데이터들을 삽입
    - image\_test\_input : X\_test, Y\_test에 특정 폴더의 데이터들을 삽입
    - DoMachineLearning : 설정되어있는 X\_test를 모델에 적용해 결과 확인
    - DoMLOneImage : 문자 1개 이미지를 모델에 적용해 결과 확인
* SoundMachine 클래스 : 전처리된 음성 데이터를 입력하고 학습된 데이터를 적용하는 역할
  + 함수 설명
    - sound\_train\_input : train.csv의 데이터들을 삽입
    - sound\_test\_input : test.csv의 데이터들을 삽입
    - DoML\_Test : 설정되어있는 X\_test를 모델에 적용해 결과 확인
    - DoMLOneSound : 파라미터 음성 csv 데이터를 모델에 적용해 결과 확인



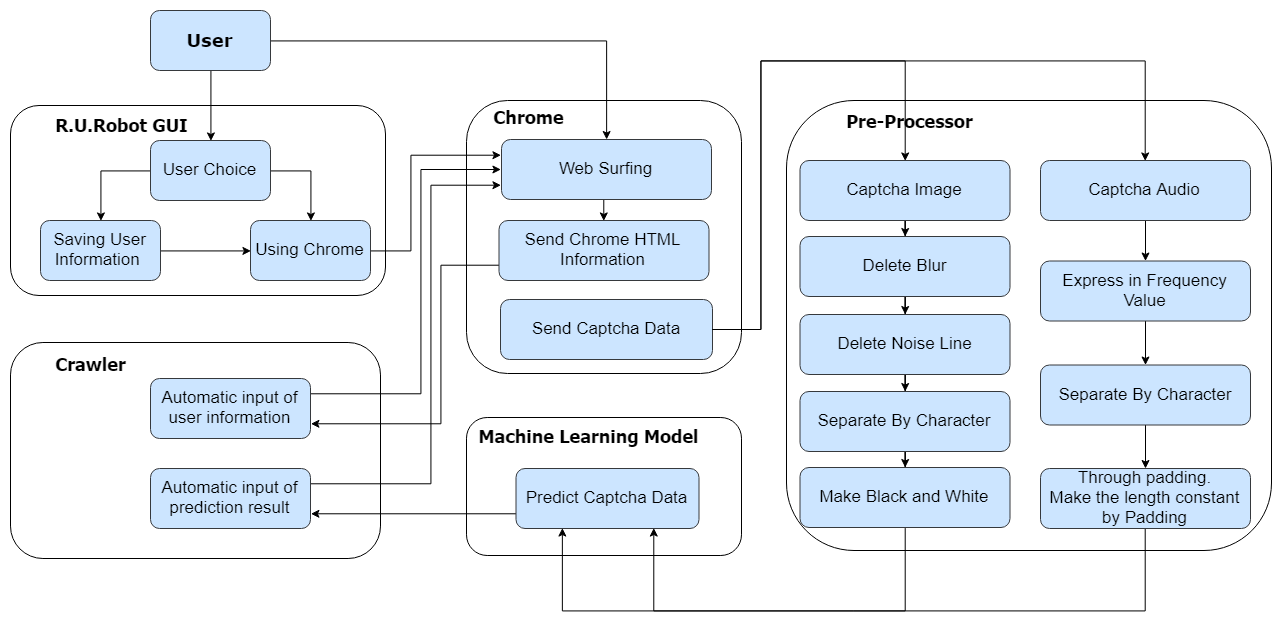
[그림 13] captcha audio class diagram

* CaptchaAudio 클래스 : 캡챠 의 음성 데이터로부터 머신러닝에 적용할 수 있도록 데이터를 처리하는 역할
  + 변수 설명
    - alpha : raw audio으로부터 추출한 각 알파벳 array
    - answer : 추출한 알파벳들을 행렬형태로 정리한 array (5,7000)
  + 함수 설명
    - cut\_by\_null : raw audio로부터 음성단위로 알파벳을 추출해내는 함수.
    - add\_on\_answer : 추출한 알파벳을 answer 행렬에 추가하는 함수
    - pad\_null : 각 음성의 길이를 동일하게 맞춰주기위해 alpha의 빈부분(MAX\_SIZE - alpha의 크기에 해당) 0으로 채워주는 함수
    - delete\_listen\_wav : 추출 이후 오디오 파일 삭제하는 함수
    - OneSoundProcessingAndML : CaptchaAudio 객체를 생성하여 전처리 후 머신러닝에 적용하여 예측결과를 반환
* numpy.ndarray 클래스 : CaptchaAudio 클래스에서 사용되며 데이터 처리를 용이하게 하는 역할
  + 변수 설명
    - size : array의 크기를 반환
    - T : 행과 열을 transpose
  + 함수 설명
    - empty : 더미값이 들어있는 array 생성



[그림 14] captcha image class diagram

* CaptchaImage 클래스 : 캡챠의 이미지 데이터로부터 머신러닝에 적용할 수 있도록 데이터를 처리하는 역할을 함
  + 변수 설명
    - origin\_name : 원본 이미지 파일의 이름을 의미
    - height, width : 원본 이미지 파일의 높이와 넓이를 의미
    - color\_most\_left\_list, color\_most\_right\_list : 원본 이미지 내 5글자의 가장 왼쪽(오른쪽)의 x좌표를 의미. 문자별로 자를때 사용됨
    - color\_list : 원본 이미지 내에서 사용된 색상의 리스트
    - each\_images : 원본 이미지 내 5글자를 자른 객체 리스트
    - result\_text : 머신러닝 모델을 적용해 예측되는 결과
  + 함수 설명
    - blur\_pixel\_delete : 원본 이미지의 blur를 제거함
    - line\_delete : 이미지의 noise line을 제거함
    - character\_separate : 이미지의 문자 5글자를 각각 자름
    - ImageSave : 이미지를 규칙에 맞게 정리하여 저장함
* PngImageFile 클래스 : CaptchaImage 클래스에서 사용되며 PNG확장자 이미지 객체의 수정을 가능하게 함
  + 변수 설명
    - height, width : 해당 객체의 높이와 넓이
  + 함수 설명
    - load : 픽셀 단위를 접근하기위해 PixelAccess 객체를 불러옴
    - crop : 파라미터 튜플 값에 맞춰 이미지를 자름
    - save : 이미지를 저장함
* PixelAccess 클래스 : PngImageFile 클래스에서 사용되며 PNG확장자 이미지 객체의 픽셀단위 접근을 가능하게 함
  + 함수 설명
    - \_\_getitem\_\_ : 이미지 내의 픽셀 값 하나의 튜플(RGBA)를 획득
    1. **Subsystem Model**

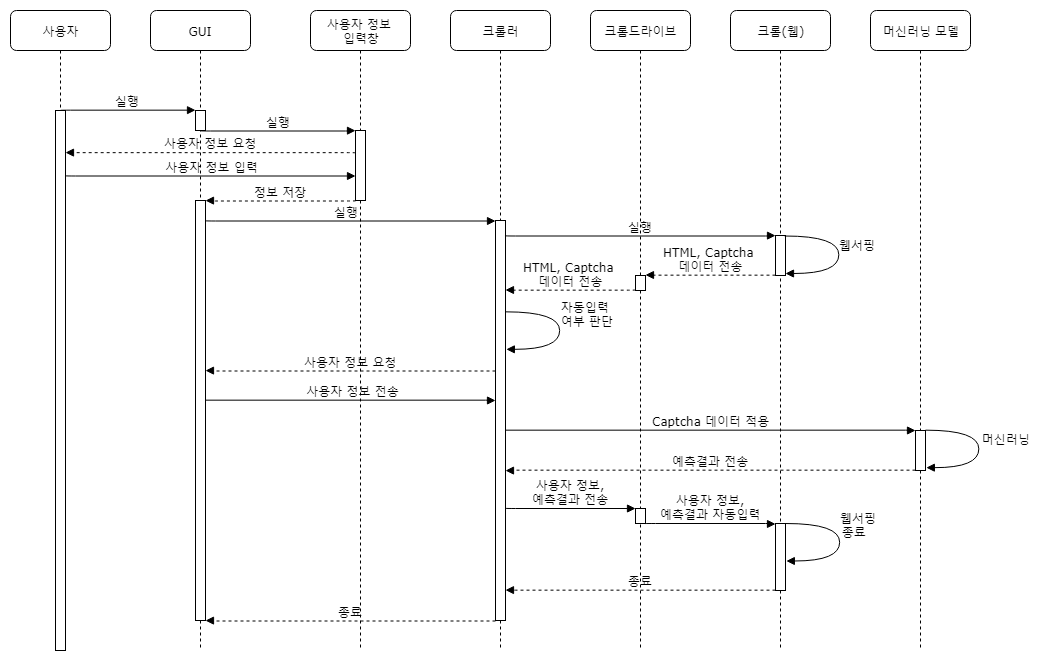


[그림 04] subsystem model

* R.U.Robot GUI
  + User Choice : 사용자는 “크롬실행”과 “사용자 정보입력”을 선택
  + Saving User Information : 사용자 정보를 입력받아 저장
  + Using Chrome : 사용자를 위해 크롬을 열며 Crawler, Chrome driver 설정
* Chrome
  + Web Surfing : 사용자는 웹서핑을 즐김
  + Send Chrome HTML Information : 현재 웹페이지의 정보를 Crawler가 수집
  + Send Captch Data : 캡챠 이미지, 음성 데이터를 pre-processor가 수집
* Crawler
  + Automatic input of user information : 저장한 사용자 정보를 자동입력
  + Automatic input of prediction result : 캡챠 머신러닝의 결과를 자동입력
* Image Pre-Processor
  + Captcha Image : 원본 이미지를 접근 가능한 객체(PngImageFile)로 변경
  + Delete Blur : 이미지의 Blur를 제거
  + Delete Noise Line : 이미지의 Noise Line을 제거
  + Separate By Character : 이미지를 문자 개별 이미지로 나눔
  + Make Black and White : 나눈 문자 이미지를 흑백화
* Audio Pre-Processor
  + Captcha Audio : 원본 음성 파일을 불러옴
  + Express in Frequency Value :음성을 접근 가능한 객체(numpy.ndarray)로 변경
  + Separate By Character : 음성을 문자 개별 음성으로 나눔
  + Through padding Make the length constant by Padding :

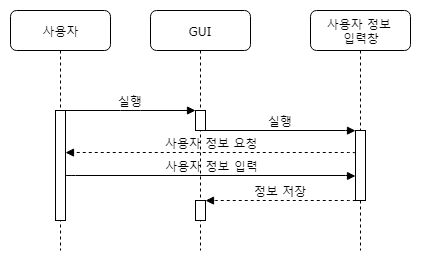
패딩을 통해서 모든 개별 음성의 길이를 동일하게 만듬

* Machine Learning Model
  + Predict Captcha Data : 전처리된 데이터를 기존의 모델에 적용하여 결과 예측
    1. **Sequence Diagram**



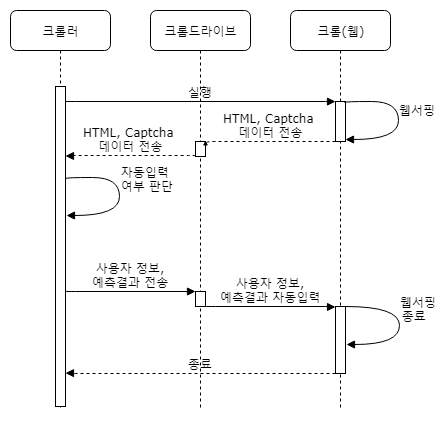
[그림 05] 전체 sequence diagram

사용자는 본 프로젝트의 실행프로그램을 실행 시켜 GUI를 이용할 수 있게 되며 GUI에서 사용자 정보 입력창이 열려 사용자는 자신의 정보를 입력시킬 수 있다. 그 후, 사용자는 크롬을 이용하여 웹서핑을 한다. 이때 크롤러와 크롬드라이브는 사용자의 현재 웹의 정보들을 수집하여 자동입력 실행 여부를 판단하며, 캡챠 데이터가 있는 경우 캡챠 데이터를 수집해 머신러닝에 적용하여 예측 결과를 사용자 정보와 함께 크롬에 자동입력 시킨다.



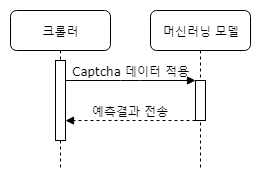
[그림 06] GUI와 사용자의 sequence diagram

사용자가 R.U.Robot.exe(본 프로젝트의 실행 프로그램)을 실행시키면 GUI가 표시된다. 사용자는 크롬을 이용하기 전, 자신의 정보를 사용자 정보 입력창에 입력해 사용자 정보를 프로그램에 저장시킨다. 해당 정보는 메모리 상에서 저장되어 사용자가 본 프로그램이 지원하는 캡챠 웹 페이지에 접속하였을 때, 자동으로 입력된다.



[그림 07] 크롤러와 웹 페이지 간의 sequence diagram

GUI에서 실행시킨 크롤러는 크롬을 실행시켜 사용자가 웹 서핑을 가능하게 한다. 웹서핑을 하는 동안 크롬드라이버를 통해 크롤러가 웹 페이지의 정보를 수집한다. 수집된 정보를 판별하여 자동입력 기능 수행 여부를 판단한다. 만약, 캡챠 페이지라면, 정보를 수집할 때 캡챠 데이터 또한 수집하여 머신러닝에 입력한다. 그 후, 사용자 정보와 예측 결과를 크롬드라이버를 통해 크롬 상에서 자동으로 입력되도록 한다. 이 과정을 사용자가 웹 서핑하는 동안 반복하며, 크롬을 종료할 경우 이를 크롤러가 알게되어 자동으로 크롤링 과정이 종료된다.



[그림 08] 크롤러와 머신러닝 모델 간의 sequence diagram

캡챠 페이지로부터 획득한 캡챠 데이터(음성, 문자)를 기존에 만들어둔 머신러닝 모델에 적용시켜 해당 데이터에 대한 예측 결과를 획득한다.

* 1. **Functionality**
     1. **Crawler**

크롤러는 웹페이지에서 필요한 정보를 추출하고, 사용자 정보를 입력하는 기능을 담당한다. 이때 크롤러는 웹페이지의 HTML 코드를 읽어드려 본인인증 페이지인지 아닌지를 판단한다. 이때 대상으로 하고 있는 웹 사이트마다 상이한 HTML elements를 가지고 있기때문에 각각에 대응하는 코드를 케이스 별로 분류하여 탐색한다. 이후 사용자가 본인인증 서비스에 접근하게 되면 크롤러는 자동으로 캡챠 이미지와 오디오를 추출한다. 추출된 데이터들은 데이터 전처리 단계로 넘겨지게 되고, 이후 머신러닝을 통해 예측된 값을 다시 넘겨받게 되면, 예측된 값과 미리 저장되어 있던 사용자 정보를 양식에 맞게 정렬하여 본인인증 웹페이지에 입력한다. 결과적으로 사용자는 본인인증 서비스에 접근만하면 모든 것이 자동적으로 수행되고, 사용자는 본인인증 코드를 본인의 핸드폰으로 전달받을 수 있다.

* + 1. **Preprocessor**

데이터 전처리(preprocessor)의 기능으로는 크게 이미지 전처리와 오디오 전처리 두 가지로 나뉜다. 먼저 이미지 전처리의 절차로는, 크롤러로부터 전달받은 캡챠 이미지를 접근이 가능한 객체(PngImageFile)로 변경한다. 이후 이미지의 픽셀을 읽어들여 알파값이 255가 아니면 흐린 이미지라고 판단, 픽셀을 색상 값을 제거하는 방식으로 이미지의 blur를 제거한다. 이미지의 배경 색상(Noise Line)은 픽셀을 횡으로 한 줄씩 읽어들여 5개의 색만 나타나는 지점을 찾아 알파벳에 해당하는 색상이라 판다, 색상을 저장해두고, 이후 그 이외의 색이 나타나면 배경 색상이라고 판단하고 색을 제거한다. 이 과정을 모두 거치고 나면 알파벳 이미지만 남게되고, 각 알파벳 좌상측 끝을 기준으로 알파벳을 분리하여 각각 흑백화한다. 각각의 단계가 수행될 때마다 이미지는 프로젝트 내에 저장되어 추후 머신러닝 알고리즘 정확도 계산에 사용된다. 다음으로 오디오 전처리 절차로는, 크롤러로부터 전달받은 캡챠 오디오를 접근이 가능한 배열 객체(numpy array)로 변경한다. 오디오를 Numpy array로 변경하게 되면 오디오의 음성을 수치화 할 수 있게 되는데 이를 통해 노이즈를 제거하고, 음성과 음성 사이의 무음이 존재하는 부분을 제거하,.여 음성을 알파벳 개별 음성으로 나눈다. 이후 패딩을 입혀 모든 음성 데이터의 길이를 동일하게 만들고, 전처리된 음성 데이터를 한 배열에 순서에 맞게 저장한다. 이렇게 저장된 이미지, 오디오 데이터들은 머신러닝 단계로 넘겨진다.

* + 1. **Machine learning**

머신러닝(machine learning)은 학습 데이터(train set) 생성과 실사용(test set)으로 나뉜다. 학습 데이터 생성 프로그램 동작에 필요한 머신 러닝 모델 생성에 필요한 과정이므로 프로그램 실행 전에 선행되어야 한다. 학습 데이터 생성은 전처리된 데이터들을 넘겨받아 패턴을 탐색하고 로지스틱 회귀 알고리즘을 기반으로 모델을 생성한다. 학습이 완료되면 각 웹사이트의 캡챠에 대한 학습 데이터들을 파이썬 객체 파일(\*.pkl)로 프로젝트 폴더 내에 저장한다. 이후 사용자에 의해 요청된 캡챠 데이터(test set)가 들어오면 생성해둔 머신러닝 모델을 통해 값을 예측한다.

* + 1. **GUI**

GUI 기능은 크게 사용자 정보 입력 및 저장과 크롬 실행 기능, 크롬 드라이버 경로 설정 세 가지로 나뉘게 된다. 첫번째로 사용자 정보 입력 및 저장 기능은 크롬 실행 전에 이행되어야하는 단계로, 추후 본인인증시에 빠르게 자동 입력이 될 수 있도록 자신의 사용자 정보를 미리 프로그램 내에 저장해두는 단계이다. 사용자 정보는 프로그램 파일 내에 텍스트 파일 형식으로 저장되게 된다. 두 번째로 크롬 실행 기능이다. 프로그램에서 크롬을 열게 되면 평상시에 크롬을 실행하는 것과는 다르게 크롬 드라이버를 통해 크롬을 디버깅 모드로 실행하게 되어 웹 크롤링에 필요한 프레임워크인 Selenium을 사용할 수 있게 된다. 세 번째 기능은 크롬 드라이버 경로 설정이다. 크롬 드라이버의 경로가 알맞게 설정되어 있어야 프로그램이 정상적으로 작동하기 때문에 사용자가 직접 크롬 드라이버의 경로를 설정할 수 있도록 해 프로그램의 안전성을 높이기 위해 추가된 기능이다.

## **Algorithm or Method**

각 세부 기능별 pseudo code는 아래와 같다.

* + 1. **MyWindow**

|  |
| --- |
| Function **setupUI**()  화면상 1000, 200 위치에 300, 300 크기의 창으로 UI를 생성  창의 제목을 “R U Robot”으로 설정  pushButton1 = “개인정보 입력”이라 적힌 버튼  pushButton1이 클릭되면 openPersonalInformation 함수 실행  pushButton2 = “크롬 열기”라 적힌 버튼  pushButton2가 클릭되면 openChrome 함수 실행  label = 라벨  layout = 레이아웃을 “QVBoxLayout()”(수직으로 나열)으로 설정  “layout”에 “pushButton1”을 추가  “Layout”에 “pushButton2”를 추가  “Layout”에 “label”을 추가  Function **openPersonalInformation**()  dlg = “LoginDialog()” 클래스 지정  “exec()”로 프로세스를 불러옴  id = “LoginDialog()”의 “id” 값  sex = “LoginDialog()”의 “sex” 값  nation = “LoginDialog()”의 “nation” 값  agency = “LoginDialog()”의 “agency” 값  password = “LoginDialog()”의 “password” 값  phone = “LoginDialog()”의 “phone” 값  chrome = “LoginDialog()”의 “chorme” 값  driver = “LoginDialog()”의 “driver” 값  f = “test.txt” 파일을 쓰기전용으로 “utf-8”로 인코딩하여 불러옴  “label”의 텍스트를 “  name: “id”  nationality : “nation”  birth: “password”  sex : “sex”  agency : “agency”  phone: “phone”  chrome.exe : “chrome”  chromedriver.exe : “driver”  ”으로 저장  “f”에 “id”, “nation”, “password”, “sex”, “agency”, “phone”, “chrome”, “driver”을 저장  Function **openChrome**()  t\_crawler = crawler.do\_crawling를 하는 쓰레드 생성  t\_crawler 쓰레드 시작 |

* + 1. **LogInDialog**

|  |
| --- |
| Function **setupUI**()  화면상 1000, 200 위치에 300, 300 크기의 창으로 UI를 생성  창 제목을 “개인정보 입력”으로 지정  self.label1 = “이름: ”을 내용으로 하는 label  self.label2 = “생년월일: ”을 내용으로 하는 label  self.label3 = “휴대폰번호: ”을 내용으로 하는 label  self.label4 = “Chrome.exe 경로: ”을 내용으로 하는 label  self.label5 = “Chromedriver.exe 경로: ”을 내용으로 하는 label  self.label6 = “통신사”을 내용으로 하는 label  self.label7 = “성별”을 내용으로 하는 label  self.label8 = “국적”을 내용으로 하는 label  self.lineEdit1 = 텍스트 입력칸 생성  self.lineEdit2 = 텍스트 입력칸 생성  self.lineEdit3 = 텍스트 입력칸 생성  self.lineEdit4 = 텍스트 입력칸 생성  self.lineEdit5 = 텍스트 입력칸 생성  self.rbtn1 = “내국인”이라 적힌 버튼 생성  self.rbtn2 = “외국인”이라 적힌 버튼 생성  self.rbtn3 = “SKT”라 적힌 버튼 생성  self.rbtn4 = “KT”라 적힌 버튼 생성  self.rbtn5 = “LGT”라 적힌 버튼 생성  self.rbtn6 = “알뜰폰”이라 적힌 버튼 생성  self.rbtn7 = “남자”라 적힌 버튼 생성  self.rbtn8 = “여자”라 적힌 버튼 생성  self.btg1 = 버튼그룹 생성  “btg1”에 “rbtn1”과 “rbtn2”를 포함  self.btg2 = 버튼그룹 생성  “btg2”에 “rbtn7”과 “rbtn8”을 포함  self.pushButton1 = “확인”이라 적힌 버튼 생성  “self.pushButton1”이 클릭된 경우 “self.pushButtonClicked”를 실행  layout = 레이아웃 생성  “label1”을 (0, 0)에 위치  “lineEdit1”을 (0, 1)에 위치  “label8”을 (0, 2)에 위치  “label7”을 (2, 2)에 위치  “label5”를 (5, 0)에 위치  “rbtn7”을 (3, 2)에 위치  “rbtn8”을 (3, 3)에 위치  “label2”를 (1, 0)에 위치  “lineEdit2”를 (1, 1)에 위치  “rbtn1”을 (1, 2)에, “rbtn2”를 (1, 4)에 위치  “label3”를 (2, 0)에 위치  “lineEdit3”를 (2, 1)에 위치  “rbtn3”을 (6, 0)에, “rbtn4”를 (6, 1)에 위치  “label4”를 (3, 0)에 위치  “lineEdit4”를 (3, 1)에 위치  “rbtn5”을 (6, 2)에, “rbtn6”를 (6, 4)에 위치  “label5”를 (4, 0)에 위치  “lineEdit2”를 (4, 1)에 위치  “pushButton1”을 (7, 2)에 위치  레이아웃 세팅 완료  Function **pushButtonClicked**()  self.id = “self.lineEdit1”에 입력된 텍스트  self.password = “self.lineEdit2”에 입력된 텍스트  self.phone = “self.lineEdit3”에 입력된 텍스트  self.chrome = “self.lineEdit4”에 입력된 텍스트  self.driver = “self.lineEdit5”에 입력된 텍스트  IF self.rbtn1 is checked  self.nation = “self.rbtn1”의 텍스트  ELSE IF self.rbtn2 is checked  self.nation = “self.rbtn2”의 텍스트  IF self.rbtn3 is checked  self.agency = “self.rbtn3”의 텍스트  ELSE IF self.rbtn4 is checked  self.agency = “self.rbtn4”의 텍스트  ELSE IF self.rbtn5 is checked  self.agency = “self.rbtn5”의 텍스트  ELSE IF self.rbtn6 is checked  self.agency = “self.rbtn6”의 텍스트  IF self.rbtn1 is checked  self.sex = “self.rbtn7”의 텍스트  ELSE IF self.rbtn2 is checked  self.sex = “self.rbtn8”의 텍스트    crawler.set\_user\_info(self.id, self.nation, self.password, self.sex, self.agency, self.phone, 0)  self.close() |

* + 1. **Crawler**

|  |
| --- |
| Function **input\_elment**(ele\_name: String, value: String)  ele = ele\_name과 일치하는 element id  ele에 value값을 입력  highlight(ele)  IF ele\_name == “captchaCode” THEN  ele를 클릭  RETURN 1  Fuction **get\_Captcha\_image**(site: String)  CASE site OF  “Gmarket” :  “captcha\_img”라는 요소를 찾아 “captcha.png”로 저장  captcha = “captcha”인 element id  value = captcha의 “value” 속성 값  url = "https://sslmember2.gmarket.co.kr/GCaptcha/CurrentSound?encValue="+value  url의 내용을 request해서 “captcha.wav”로 저장  “Auction” :  “gCapImage”라는 요소를 찾아 “captcha.png”로 저장  captcha = “hidCaptcha”인 element id  value = captcha의 “value” 속성 값  url = "https://memberssl.auction.co.kr/GCaptcha/CurrentSound?encValue=" + value  url의 내용을 request해서 “captcha.wav”로 저장  “PASS” :  넘어감  ELSE CASE :  print 오류표시  END CASE  Fuction **set\_element\_name**(site, name, nationality, birth\_date, gender\_male,  gender\_female, mobile\_carrier, mobile\_number, captcha\_character: String)  IF site == self.element\_name[“사이트”] THEN # 기존 사이트와 동일하다면  RETURN  element\_name[“사이트”] = site  element\_name[“이름”] = name  ...  element\_name[“자동입력방지문자”] = captcha\_character  Fuction **input\_user\_info**(site\_name: String)  CASE site OF  “Gmarket” :  iframe = “popLayerIframe1”과 일치하는 element id  driver를 iframe으로 switch  set\_elment\_name("Gmarket", "u\_name", "naSelect", "birth\_date",  "gender\_male", "gender\_female", "carrier\_sel",  "cellphone\_num", "captchaCode")  “Auction” :  “bt\_sound”와 일치하는 element class 찾아 click  set\_element\_name("Auction", "name", "naSelect", "ssnLeft8",  "sexSelect1", "sexSelect2", "carrierType", "cpNo", "captchaCode")  “PASS” :  set\_element\_name("PASS", "smsName", "native", "smsBirth", "m\_sel",  "fm\_sel", None, "smsMobileNum", "s\_secureText")  ELSE CASE :  print 오류표시  END CASE  IF successInput is False THEN  IF get\_Captcha\_image(site\_name) is False THEN  print 이미지 다운로드 실패  user\_info[“자동입력방지문자”] = OneSoundProcessingAndML()  IF input\_element(element\_name[“이름”], user\_info[“이름”] is False THEN  print 이름 입력 실패  return  select = Select(element\_name[“국적”]과 일치하는 element id)  select에서 user\_info[“국적”]에 일치하는 콤보박스 선택  IF input\_element(self.element\_name["생년월일"], self.user\_info["생년월일"])  is False THEN  print 생년월일 입력 실패  return  IF element\_name[“성별남”] is not None and element\_name[“성별여”] is not  None THEN  TRY  IF user\_info[“성별”] == “남자” THEN  element\_name[“성별남”]에 일치하는 element id click  ELSE THEN  element\_name[“성별여”]에 일치하는 element id click  EXCEPT NoSuchElementException THEN  IF user\_info[“성별”] == “남자” THEN  element\_name[“성별남”]에 일치하는 element class click  ELSE THEN  element\_name[“성별여”]에 일치하는 element class click  IF element\_name[“통신사”] is not None THEN  select = Select(element\_name[“통신사”]과 일치하는 element id)  select에서 user\_info[“통신사”]에 일치하는 콤보박스 선택  IF input\_element(element\_name[“휴대폰번호”], user\_info[“휴대폰번호”] is not  False THEN  print 휴대폰 번호 입력 실패  return  IF input\_element(element\_name[“자동입력방지문자”],  user\_info[“자동입력방지문자”] is not False THEN  print 자동입력방지문자 입력 실패  return  successInput = True    Function **set\_user\_info**(name, nationality, birth\_date, gender, mobile\_carrier,  mobile\_number, string: String)  IF name THEN  user\_info[“이름”] = name  IF nationality THEN  user\_info[“국적”] = nationality  ...  IF mobile\_number THEN  user\_info[“휴대폰번호”] = mobile\_number  Function **do\_crawling**()  chrome\_driver = “크롬드라이버 위치 지정”  driver = webdriver.Chrome(chrome\_driver)  WHILE True  time.sleep(1)  current\_url = driver.current\_url  driver의 window를 현재 window로 switch  IF "sslmember2.gmarket.co.kr/FindID" in current\_url THEN  input\_user\_info(“Gmarket”)  ELSE IF "Common/CustomizedVerification" in current\_url THEN  input\_user\_info(“Auction”)  ELSE IF "mobile-ok.com/SimplePop" in current\_url THEN  input\_user\_info(“PASS”)  END WHILE |

* + 1. **CaptchaAudio**

|  |
| --- |
| Function **cut\_by\_null**(hey\_audio: Tuple)  anp = “오디오 파일 numpy array로 저장”  cnt = -1  i = 0  null\_count = “빈 numpy array”  alpha = “빈 numpy array”  answer = “크기가 7000인 numpy array”  WHILE i > anp.size  IF “해당 셀의 값이 0” THEN  WHILE i< anp.size and anp[i] ==0  “0을 null\_count에 추가”  i ++  END WHILE  IF “0의 개수가 10미만” THEN  “alpha에 append” #음성 중간의 0  ELSE #빈 공간의 0  alpha = “pad 추가”  IF cnt in range(0,4) THEN  answer = “답 array 에 추가”  alpha = “initialize”  cnt += 1  null\_count = “initialize”  ELSE  “anp[i]를 alpha에 추가”  i += 1  END WHILE  “alpha에 0 패딩”  “answer 에 alpha 추가”  “answer에 첫번째행 제거”  self.delete\_listen\_wav()  Function **add\_on\_answer**(alpha: 추가할 답, answer: 기존 답)  new = “기존 답에 추가할 답 추가”  return new  Function **pad\_null**(without\_null)  pad = np.zeros(최대크기 - without\_null.size)  with\_null = “기존답과 pad를 append”  return with\_null  Function **delete\_listen\_wav**()  file = “디렉토리 경로”  IF “디렉토리 경로에 파일이 존재” THEN  “해당파일 삭제”  ELSE  print “file doesn’t exist”  Function **OneSoundProcessingAndML**()  CaptchaAudio 객체 pre 선언  audio = pre.cut\_by\_null(hey\_audio=”오디오 경로”)  audio = audio.T  return result[0] + result[1] + result[2] + result[3] + result[4]  return result 결과 리턴 |

* + 1. **CaptchaImage**

|  |
| --- |
| Function **blur\_pixel\_delete**()  FOR y = 0 TO height  FOR x = 0 TO width  IF origin\_pixel[x, y][3] != 255 THEN  origin\_pixel[x, y] = 흰색  NEXT x  NEXT y  Function **line\_delete**()  FOR y = 5 TO height  color\_lists = []  FOR x = 0 TO width  IF origin\_pixel[x, y] not in color\_lists THEN  IF origin\_pixel[x, y] != 흰색 and  (origin\_pixel[x, y] == origin\_pixel[x+2, y] or  origin\_pixel[x, y] == origin\_pixel[x-2, y]) THEN  color\_lists에 origin\_pixel[x, y] append  IF length of color\_lists == 5 THEN  BREAK  NEXT x  NEXT y    FOR y = 0 TO height  FOR x = 0 TO width  IF origin\_pixel[x, y] not in color\_lists THEN  origin\_pixel[x, y] = 흰색  ELSE IF origin\_pixel[x, y] in color\_lists THEN  FOR i = 0 to length of color\_lists  IF origin\_pixel[x, y] == color\_lists[i] THEN  color\_most\_left\_list[i]=min(color\_most\_left\_list[i], x)  color\_most\_right\_list[i]=max[color\_most\_right\_list[i], x)  NEXT i  NEXT x  NEXT y  Function **character\_separate**()  FOR i = 0 in 5  bbox=(self.color\_most\_left\_list[i], 0, self.color\_most\_right\_list[i], 45)  new\_image = origin\_image를 bbox로 자른 이미지  new\_pixel = new\_image.load()  bg\_image = MAX\_WIDTH, new\_image.height 크기만큼 흰색의 이미지  this\_image\_color = ()  new\_color\_dicts = {}  FOR y = 0 TO new\_image.height  FOR x = new\_image.width-1 to 0  IF new\_pixel[x, y] != 투명색 THEN  IF new\_pixel[x, y] not in new\_color\_dicts.keys() THEN  new\_color\_dicts[new\_pixel[x, y]] = 1  ELSE THEN  new\_color\_dicts[new\_pixel[x, y]] += 1  PREVIOUS x  NEXT y  max\_value = max(new\_color\_dicts.values())  FOR color\_dict in new\_color\_dicts  IF new\_color\_dicts[color\_dict] == max\_value THEN  this\_image\_color = color\_dict  BREAK  NEXT color\_dict    FOR y = 0 TO new\_image.height  FOR x = 0 TO new\_image.width  IF new\_pixel[x, y] != this\_image\_color THEN  new\_pixel[x, y] = 흰색  ELSE THEN  new\_pixel[x, y] = 검정색  NEXT x  NEXT y  bg\_image에 new\_image 덮어씌우기(paste)  new\_image = bg\_image  each\_images[i] = new\_image  NEXT i  Function **ImageSave**(save\_dir\_name: String, counter: Integer)  IF save\_dir\_name이 접근 불가능 THEN  save\_dir\_name의 디렉토리 생성  IF save\_dir\_name + “/train”이 접근 불가능 THEN  save\_dir\_name + “/train”의 디렉토리 생성  IF save\_dir\_name + “/test”이 접근 불가능 THEN  save\_dir\_name + “/test”의 디렉토리 생성  i = 0  FOR char in origin\_name[7:12]  IF save\_dir\_name + “/train” + char이 접근 불가능 THEN  save\_dir\_name + “/train” + char의 디렉토리 생성  IF save\_dir\_name + “/test” + char이 접근 불가능 THEN  save\_dir\_name + “/test” + char의 디렉토리 생성  IF counter[ord(char)-65] == TEST\_NUM(default: 2) THEN  save\_dir\_name+"/train/"+char+"/"+file\_name[:5]+"\_result\_"+str(i)+".png"  의 파일명으로 each\_images[i] 저장  ELSE THEN  save\_dir\_name+"/test/"+char+"/"+file\_name[:5]+"\_result\_"+str(i)+".png"  의 파일명으로 each\_images[i] 저장  NEXT char, i |

* + 1. **LearnedMachine**

|  |
| --- |
| Function **DoModelSave**(save\_model\_name: String)  logreg = LogisticRegression 클래스 생성  logreg.fit(X\_train, Y\_train)  “./save\_model\_name.pkl”의 파일명으로 선언한 logreg 모델 저장 |

* + 1. **ImageMachine(LearnedMachine)**

|  |
| --- |
| Function **image\_train\_input**()  FOR char in “ABCDEFGHIJKLMNOPQRSTUVWXYZ” THEN  train\_file\_list = "./train\_test\_image/train/" + char + "/"의 내부 리스트  FOR file\_name in train\_file\_list  X\_train에 XinputRGBPlusWithName("./train\_test\_image/train/"+char+"/"  +file\_name) APPEND  Y\_train에 char APPEND  NEXT file\_name  NEXT char  Function **image\_test\_input**()  FOR char in “ABCDEFGHIJKLMNOPQRSTUVWXYZ” THEN  test\_file\_list = "./train\_test\_image/test/" + char + "/"의 내부 리스트  FOR file\_name in test\_file\_list  X\_test에 XinputRGBPlusWithName("./train\_test\_image/test/"+char+"/"  +file\_name) APPEND  Y\_test에 char APPEND  NEXT file\_name  NEXT char  Function **DoMachineLearning**(model\_name: String)  logreg = model\_name의 모델 LOAD  print 정확도 : logreg.score(X\_test, Y\_test)  @staticmethod  Function **DoMLOneImage**(model\_name: String, target\_image: Image)  test = []  test에 XinputRGBPlusWithImage(target\_image) APPEND  logreg = model\_name의 모델 LOAD  RETURN logreg.predict(test) |

* + 1. **SoundMachine(LearnedMachine)**

|  |
| --- |
| Function **sound\_train\_input**(csv\_file\_name: List)  csv\_data = csv\_file\_name READ wih pandas  header = list(csv\_data.columns)  FOR i in header  Y\_train에 i[2:3] APPEND  NEXT i  row\_count = length of csv\_data  column\_count = length of Y\_train  csv\_array = csv\_data.values  FOR j = 0 TO column\_count  temp\_X = []  FOR i = 0 TO row\_count  temp\_X에 csv\_array[i][j] APPEND  NEXT i  X\_train에 temp\_X APPEND  NEXT j  Function **sound\_test\_input**(csv\_file\_name: List)  csv\_data = csv\_file\_name READ wih pandas  header = list(csv\_data.columns)  FOR i in header  Y\_test에 i[2:3] APPEND  NEXT i  row\_count = length of csv\_data  column\_count = length of Y\_test  csv\_array = csv\_data.values  FOR j = 0 TO column\_count  temp\_X = []  FOR i = 0 TO row\_count  temp\_X에 csv\_array[i][j] APPEND  NEXT i  X\_test에 temp\_X APPEND  NEXT j  Function **DoML\_Test**(model\_name: String)  logreg = model\_name의 모델 LOAD  RETURN logreg.predict(X\_TEST)  @staticmethod  Function **DoMLOneSound**(model\_name: String, csv\_data: List)  row\_count = length of csv\_data  csv\_array = csv\_data  test = []  FOR j = 0 TO 5  temp = []  FOR i = 0 TO row\_count  temp에 csv\_array[i][j] APPEDN  NEXT i  test에 temp APPEND  NEXT j    logreg = model\_name의 모델 LOAD  RETURN logreg.predict(test) |

# **Implementation Results**

## **Test environment**

테스트에 적용된 환경 설정은 하기와 같다.

|  |  |
| --- | --- |
| 환경 | 비고 |
| 운영체제 | Windows 10 |
| 크롬 버전 | 74.0.3729.108 (x64) |
| 크롬드라이버 | 74.0.3729.6 |

[표 ] 테스트 환경

## **Test scenario**

### **Test scenario by functions**

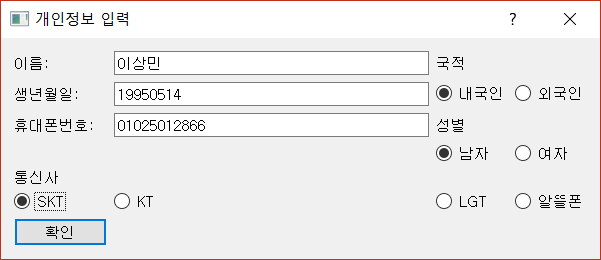
1. 프로그램 실행



[그림 09] GUI 메인창 띄우기

프로그램을 실행시키면 [그림 09]와 같이 GUI 메인창이 정상적으로 표시된다.

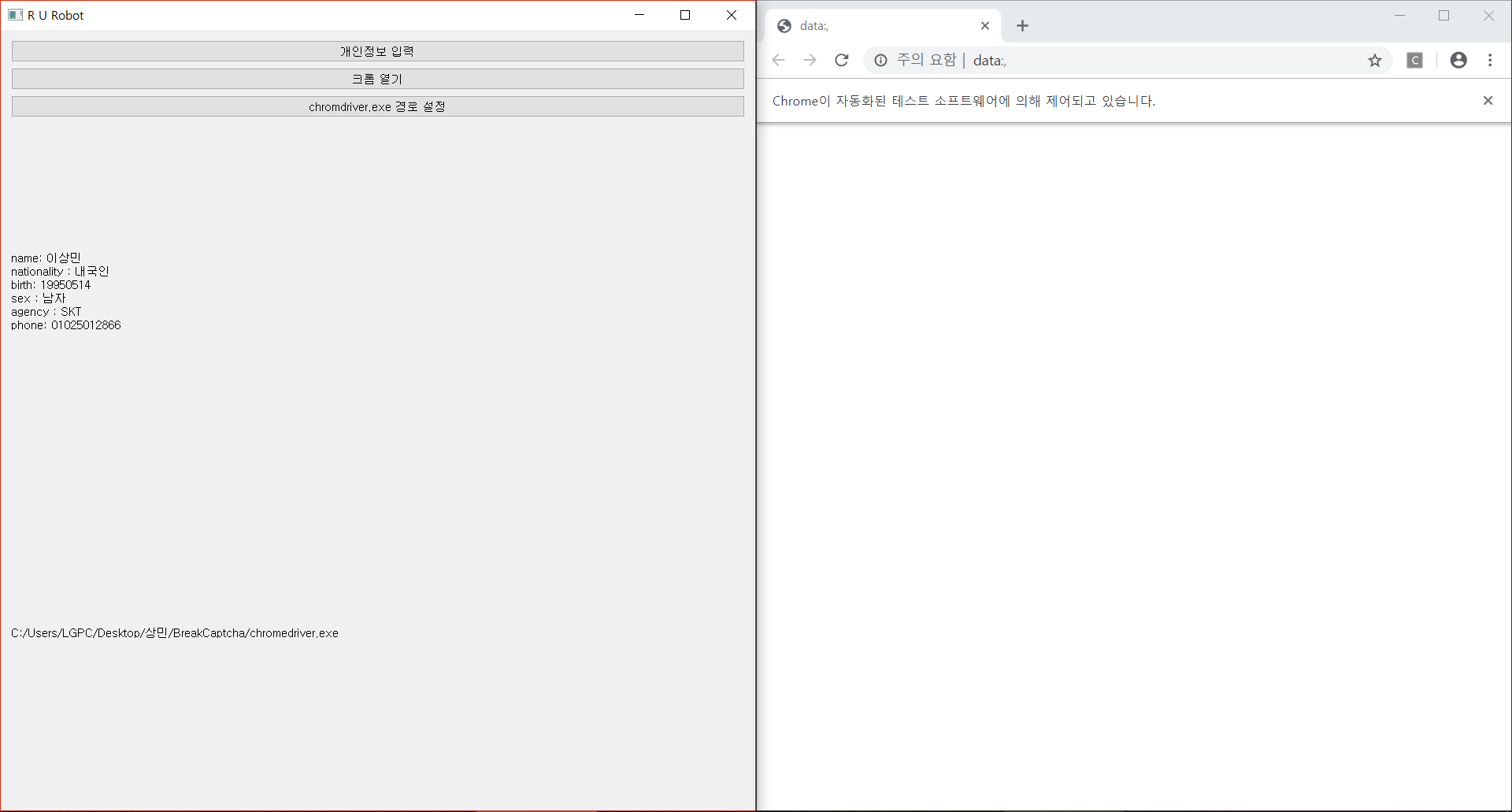
1. 사용자 정보 입력



[그림 10] 사용자 정보 입력

GUI 메인창에서 개인정보 입력 버튼을 클릭하면 사용자 정보 입력창이 열린다. 사용자는 자신의 정보를 넣고 확인을 누르면 메모리에 사용자의 정보가 저장된다.

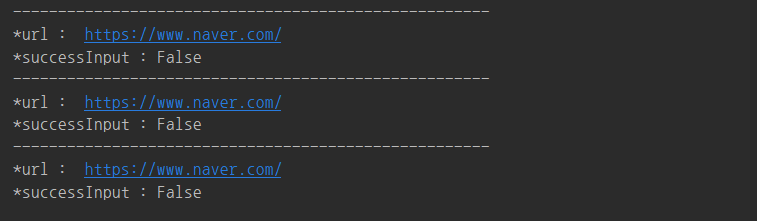
1. 프로그램을 통한 크롬 실행



[그림 11] “크롬 열기” 버튼으로 실행한 크롬 브라우저

GUI 메인창에서 크롬 열기 버튼을 클릭하면 크롬 브라우저가 열린다. 만약 사용자 정보를 입력하지 않고 바로 크롬 열기 버튼을 클릭한다면, default 값의 사용자 정보가 이용된다.

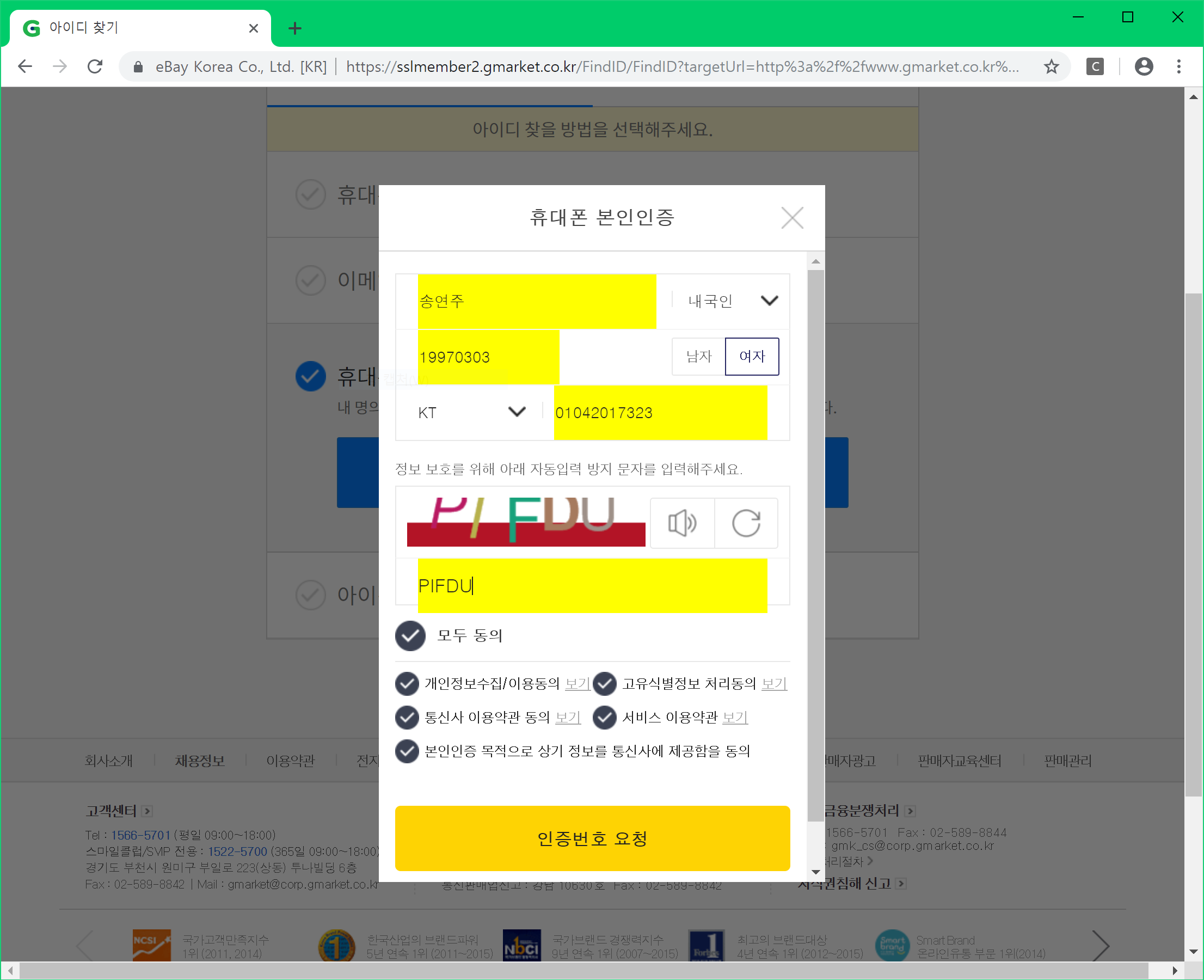
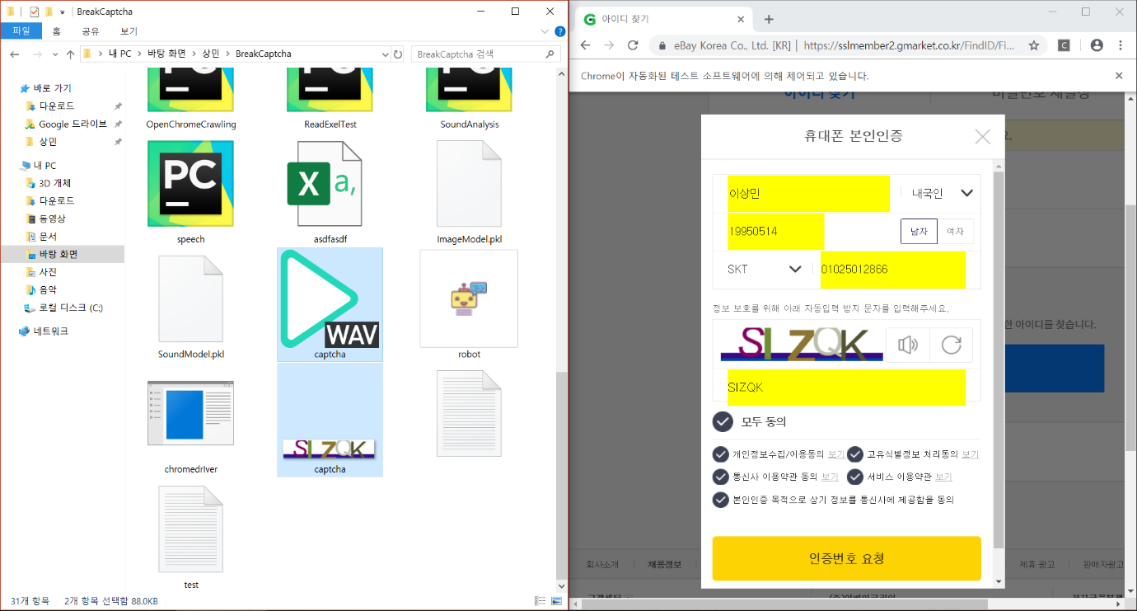
1. 프로그램을 통한 웹 서핑



[그림 12] 사용자 웹 서핑 정보(HTML, url 등) 수집

사용자는 열린 크롬 브라우저에서 자유롭게 웹서핑을 하면 본 프로젝트의 프로그램은 사용자가 웹서핑할 때의 정보(HTML, url 등)을 수집한다. [그림 12]에서 URL만 수집되듯이 보이지만 HTML과 driver 정보 모두 수집하고 있다. 만약 탭이 여러 개라면 가장 우측에 있는 탭에 대하여 작동을 한다. 창이 여러 개라면 가장 마지막에 열린 창에 대해 작동을 한다.

1. 보인인증 서비스 접근



[그림 13] 캡챠 데이터 수집

[그림 13]의 오른쪽 페이지처럼, 캡챠 페이지를 접근한다면 머신러닝으로 캡챠 데이터를 예측하기 위해 음성파일 wav와 이미지파일 png를 받아온다. [그림 13]의 좌측 폴더 내 파일들을 보아 정상적으로 추출한 것을 확인할 수 있다.

1. 캡챠 이미지 전처리





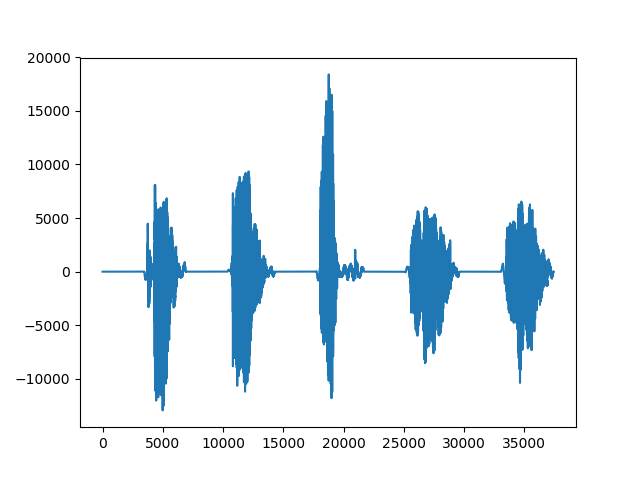


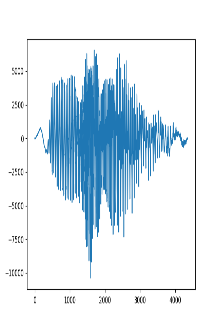
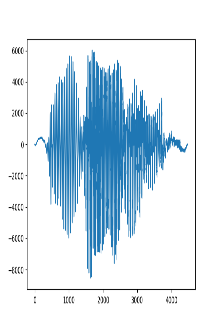
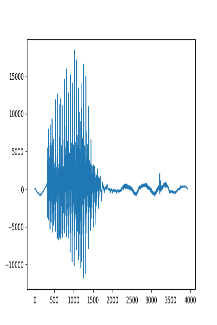
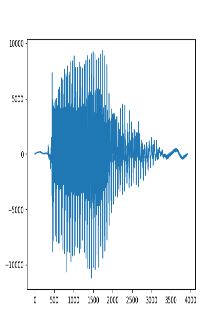
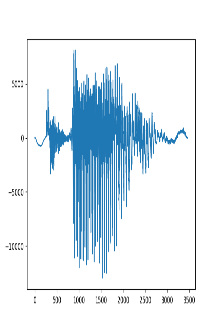
    

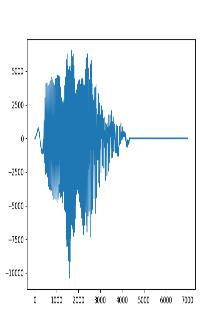
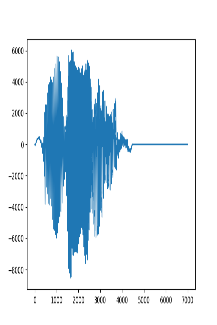
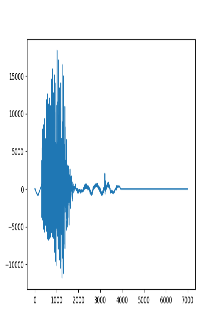
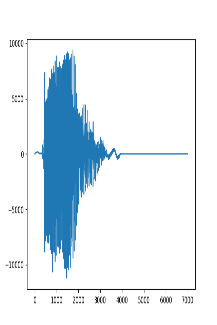
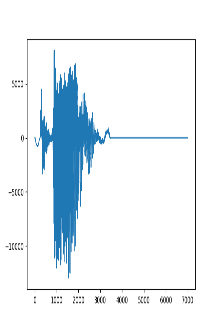
[그림 14] 이미지 전처리 과정

추출한 이미지를 머신러닝에 적용하기 전, 학습된 데이터 모델에 맞추어 정확도 향상을 위해 전처리 과정을 거친다. [그림 14]를 통해 원본파일이 전처리 되어 각각의 결과가 나타나는 것을 확인 할 수 있다. 이 전처리 과정은 높이가 일정해야 오류가 나지 않으며, 이 전처리 과정을 따르는 G마켓, 옥션의 경우 높이가 동일하다. 150개가 넘는 데이터에 대해 전처리 과정을 거쳤을 때, 모두 정상적으로 전처리가 되었지만 예외상황이 있을 가능성이 있다.

1. 캡챠 오디오 전처리



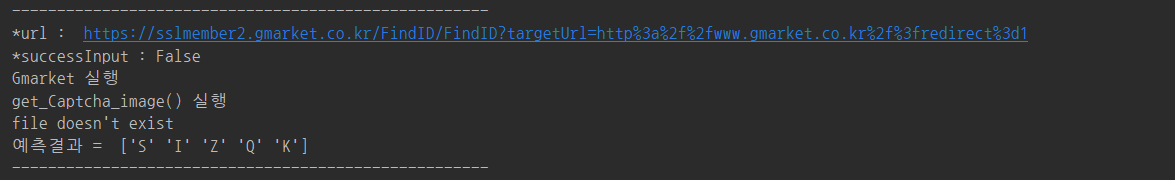




[그림 15] 오디오 전처리 과정

마찬가지로 추출한 음성파일을 머신러닝에 적용하기 전, 학습된 데이터 모델에 맞추어 정확도 향상을 위해 전처리 과정을 거친다. [그림 15]를 통해 원본파일이 전처리 되어 각각의 결과가 나타나는 것을 확인 할 수 있다.

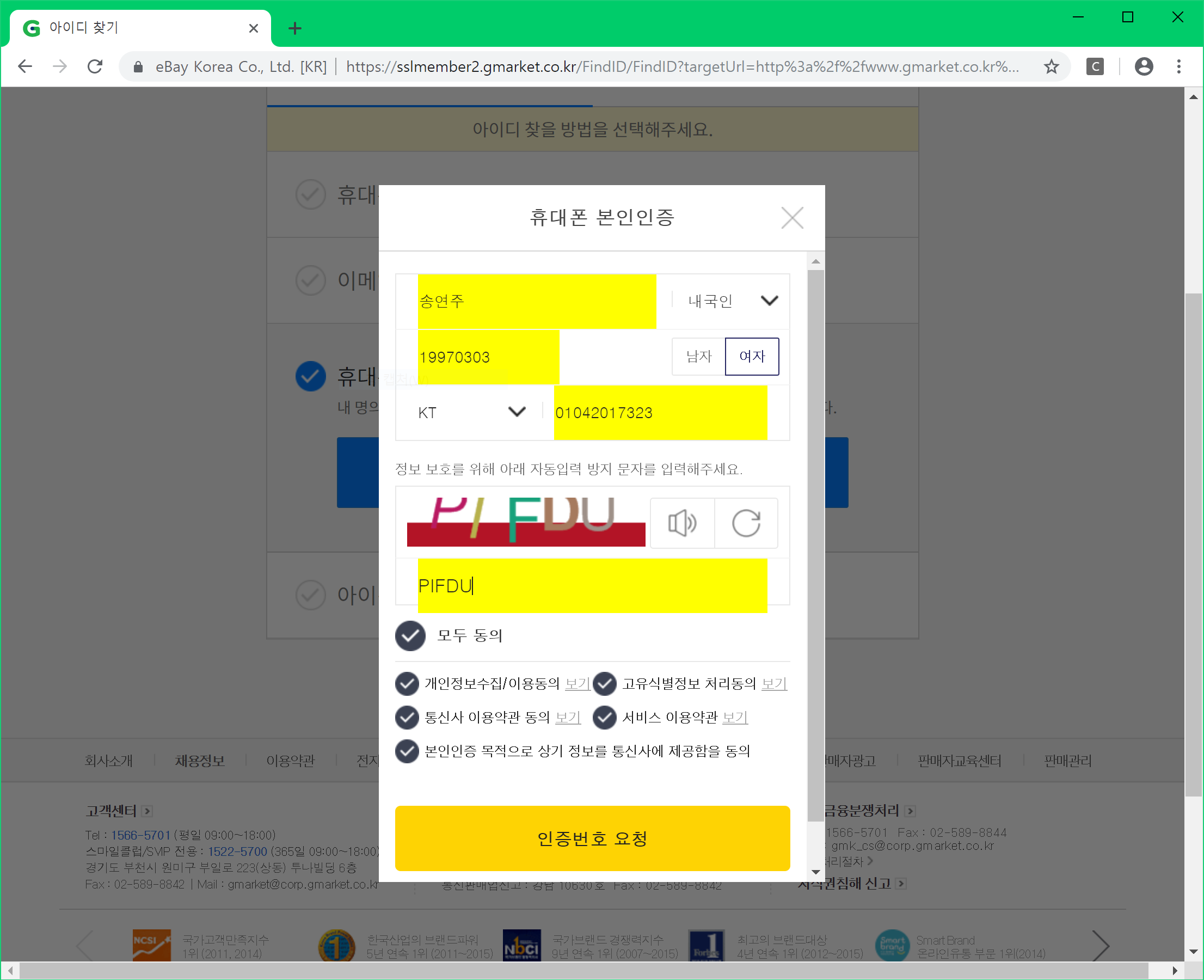
1. 머신 러닝



[그림 16] 캡챠 예측 결과

[그림 16]은 전처리과정을 거친 음성, 이미지 데이터를 기존 학습시켜놓은 머신러닝 모델에 적용하여 예측결과를 출력하는 것을 보여준다. 자동입력시 [그림 16]의 예측결과 리스트를 문자열로 변환시켜 입력된다.

1. 사용자 정보 및 캡챠 입력



[그림 17] 사용자 정보, 캡챠 예측 자동입력

[그림 17]은 예측한 결과와 GUI상에서 입력받은 사용자 정보를 자동으로 웹페이지 상에 입력하는 과정을 보이고 있다.

### **Total test scenario**

전체적인 테스트 시나리오는 4.1.1에서 보인 기능 테스트 시나리오를 마치 사용자가 본 프로그램을 사용하듯이 한 경우를 작성하였다.

먼저, 사용자가 본인인증 사이트에 접근하였을 때 HTML elements를 탐색하여 캡챠 이미지 데이터와 오디오 데이터를 추출한다. 추출한 데이터는 데이터 전처리로 넘겨지게 되는데 데이터 전처리 단계에서는 제공받은 데이터들의 노이즈를 제거하고 머신러닝이 접근할 수 있는 객체 형태로 가공한다. 이후 전처리된 데이터들은 머신러닝으로 넘겨지게 되고 머신러닝은 기존에 가지고 있던 학습된 데이터와 넘겨받은 데이터들의 비교를 통해 값을 예측한다. 이렇게 계산된 결과들은 다시 웹 크롤러에게 넘겨지게 되고 웹 크롤러는 넘겨받은 캡챠 값과 프로그램 동작 전에 사용자로부터 입력받은 사용자 정보를 웹페이지 양식에 맞게 자동 입력한다.

## **Results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STEP | 테스트 항목 | 기대결과 | 실제결과 | 비고 |
| 01 | 프로그램 실행 | 메인 창 띄우기 | 정상동작 |  |
| 02 | 사용자 정보 입력 | 사용자 정보 입력창 띄우기  사용자 정보 저장 | 정상동작 |  |
| 03 | 프로그램 통한 크롬 실행 | 크롬드라이버를 이용한 실행 | 정상동작 |  |
| 04 | 프로그램 통한 웹 서핑 | HTML, 웹 정보 추출 | 정상동작 |  |
| 05 | 본인인증 서비스 접근 | 캡챠 데이터 추출 | 정상동작 |  |
| 06 | 캡챠 이미지 전처리 | 이미지 객체로 변환  블러 제거  라인 제거  알파벳 분리  흑백화 | 정상동작 | 실제 본 프로그램 작동시 사용하지 않음 |
| 07 | 캡챠 오디오 전처리 | Numpy array로 변환  알파벳 분리  단일배열 생성 | 정상동작 |  |
| 08 | 머신 러닝 | 기존 학습데이터 불러와 적용 | 정상동작 |  |
| 09 | 사용자 정보 및 캡챠 입력 | 캡챠 예측 값 입력  사용자 정보 입력  약관 모두 동의하기 입력  해당 연락처로 본인인증 문자 발송 | 정상동작 |  |

# **Conclusion**

* 1. **역할 분담**

각 구성원 별 역할 분담은 하기와 같다.

|  |  |
| --- | --- |
| 김형호 | 프로젝트 전체 진행 관리  문자 인식 라이브러리, Tesseract 테스트  딥러닝 알고리즘 학습 및 테스트 |
| 이상민 | Selenium을 통한 사용자 정보 자동입력  음성 STT, CSR, API 테스트  이미지 전 처리 과정 구현  머신러닝 알고리즘 학습 및 테스트 |
| 송연주 | GUI 설계 및 생성  음성 STT, 구글 speech API 테스트  음성 전 처리 과정 구현  Abstract 작성 |
| 조영준 | GUI 상세 기능 구현  사용자 정보 저장 기능 구현  Selenium을 통한 캡챠 데이터 다운로드  이미지 및 음성 데이터 Train, Test set 분류 |

[표 ] 역할 분담

* 1. **개발 일정**

본 프로젝트의 개발 일정은 하기 표와 같다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 3 | | | | 4 | | | | 5 | | | | 6 | | |
|  |  | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 |
| 기획 | 아이디어 회의 | ● | ● | ● |  |  |  |  |  |  |  |  |  |  |  |
| GUI | UI 제작 |  |  |  | ● | ● | ● | ● |  |  |  |  |  |  |  |  |
| 웹  크롤링 | 캡챠 데이터 추출 |  |  |  |  | ● | ● | ● | ` |  |  |  |  |  |  |  |
| 사용자 정보 입력 |  |  |  |  |  |  |  | ● | ● |  |  |  |  |  |  |
| 음성  처리 | API 활용 음성인식 |  |  |  | ● |  |  |  |  |  |  |  |  |  |  |  |
| 오디오 전처리 |  |  |  |  | ● | ● | ● |  |  |  |  |  |  |  |  |
| 타 사이트 확장 |  |  |  |  |  |  |  |  | ● | ● | ● | ● |  |  |  |
| 이미지 처리 | Tesseract 사용 |  |  | ● |  |  |  |  |  |  |  |  |  |  |  |  |
| 이미지 전처리 |  |  |  | ● | ● | ● |  |  |  |  |  |  |  |  |  |
| 타 사이트 확장 |  |  |  |  |  |  |  |  | ● | ● | ● | ● |  |  |  |
| 머신  러닝 | 머신러닝모델 개발 |  |  |  |  | ● | ● | ● | ● |  |  |  |  |  |  |  |
| 학습 및 테스트 |  |  |  |  | ● | ● | ● | ● | ● |  |  |  |  |  |  |
| 디버깅 | 코드 디버깅 |  |  |  |  |  |  |  |  |  | ● | ● | ● |  |  |  |
| 문서  작성 | 아이디어 제안서 |  |  |  |  | ● | ● | ● |  |  |  |  |  |  |  |  |
| 디자인 설계서 |  |  |  |  |  |  |  | ● | ● | ● |  |  |  |  |  |
| 최종 보고서 |  |  |  |  |  |  |  |  |  |  | ● | ● | ● | ● |  |
| 판넬 제작 | ` |  |  |  |  |  |  |  |  |  |  | ● |  |  |  |
| 대회  참가 | 융합캡스톤디자인 |  |  |  |  |  |  |  | ● | ● | ● | ● | ● | ● | ● | ● |
| LINC+캡스톤디자인 |  |  |  |  |  |  |  |  |  | ● | ● | ● | ● | ● | ● |

[표 ] 개발 일정

3월에는 아이디어 회의를 중점으로 작업하였고 3월 말에는 프로젝트를 어떠한 방법으로 접근해야 효과적일지 검토하는 시기로서 검증할 사이트를 탐색하고 여러 API와 라이브러리를 시험해보았다. 4월은 본격적인 개발 단계로, 웹 크롤링, 이미지 전처리, 음성 전처리 등을 완성하였다. 5월과 6월에는 디버깅과 문서 작업이 주를 이루었다. 또한, 타 사이트로 확장을 위해 조사와 코드 확장이 이루어졌다.

* 1. **개발 비용**

3~6월 간 사용된 개발 비용 지출내역은 하기와 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 연번 | 사용일자 | 집행 세부내역 및 품목 | 사용처 | 사용금액 | 사용방법 |
| 1 | 2019.05.02 | 회의비 | 아대돈부리식당 | 40,000 원 | 현금영수증 |
| 2 | 2019.05.09 | 회의비 | 멘야코이치아주대 | 39,500 원 | 현금영수증 |
| 사용금액 합계 | | **79,500 원** | | | |

[표 ] 개발 비용 내역

개발은 모두 소프트웨어 상에서 구현되었기 때문에 기기 구입 비용이 따로 들지않았다. 다만 두차례에 걸친 팀 미팅에서 점식식사 비용으로 사용하였다.

* 1. **위험요소 및 해결방법**

본 프로젝트 수행 중 맞닥뜨린 위험요소는 다음과 같다. 첫 번째로, 지마켓의 경우 본인인증 페이지 접근 시에 로딩 시간이 7초에서 길게는 10초로 많이 지연되었다. 하지만 이는 확인해본 결과

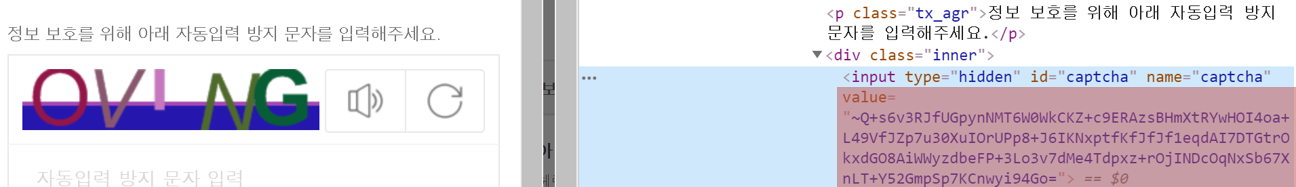
지마켓 웹 페이지 자체의 문제점으로 확인되었고, 따라서 근본적인 해결이 불가능하였다. 그러므로 본 프로젝트에서는 사용자 정보 입력과 머신러닝을 스레드로 구현해 프로그램 동작 속도를 향상시킴으로서 문제를 해결하였다. 두번째, 사용자가 초기에 사용자 정보를 입력할 때 포멧에 맞지 않는 데이터 입력시 프로그램에 오류가 발생하는 문제가 있었다. 하지만 사용자 접보 입력시 Regular expression을 활용햐 유효성 검사를 함으로써 문제를 해결할 수 있었다.

* 1. **취약점 분석 결과 및 해결방안**

국내 사이트의 본인인증 서비스의 취약점를 분석해본 결과 크게 웹페이지 상의 취약점, 캡챠 이미지 상의 취약점, 캡챠 오디오 상의 취약점 세가지로 나눌 수 있었다. 각각의 취약점과 해결방안에 대한 자세한 설명은 아래와 같다.

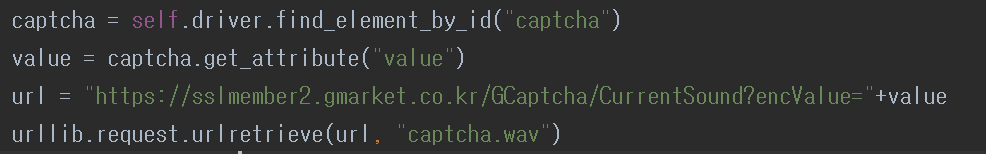
### **웹페이지 상의 취약점**

웹 페이지 상의 취약점은 본인인증 페이지에 존재하는 취약점으로 캡챠 이미지와 오디오 정보를 추출하는 과정에서 발생할 수 있는 취약점을 말한다.



[그림 ] values 값이 노출되어 있는 HTML 코드(지마켓)

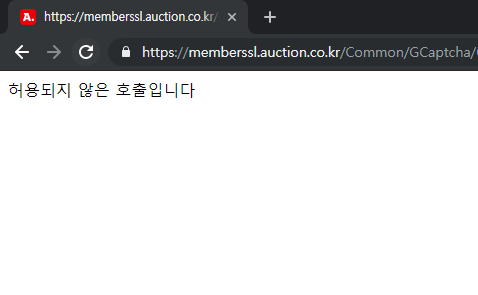
지마켓의 경우 캡챠 이미지가 등장하는 웹 페이지 상의 HTML 코드 내에서 손쉽게 캡챠 이미지 및 음성에 사용되는 value 값을 찾아낼 수 있었다. 원하는 value 값이 노출되어 있다면 기초적인 지식만 있다면 몇 줄의 코드만으로 원하는 정보를 추출할 수 있다.



[그림 ] 캡챠 정보를 다운로드하는 코드

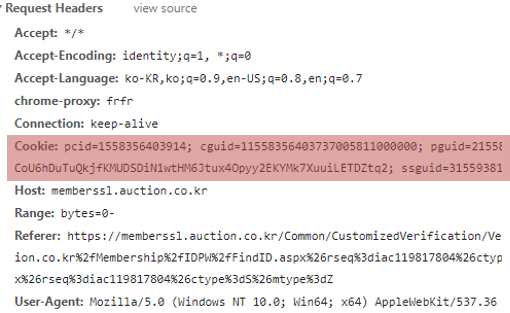
이렇게 보안 매체의 데이터를 손쉽게 가져올 수 있을수록 해킹에 용이하다. 따라서 해당 서비스를 제공해줄 때부터 핵심 데이터의 노출을 최소화할 수 있는 방법을 마련해야 할 것이다.

옥션의 경우 지마켓과 달리 단순히 링크 접속을 통해 음성 파일을 다운로드 할 수 없었고, request 시에 헤더 값을 알맞게 설정해야 다운로드가 가능했다.



[그림 ] 링크 접속을 통한 다운로드 실패 (옥션)

이 때 페이지 상에선 쿠키 값이 매번 바뀌는 것에 반해 실제 request 시도 시 이전 쿠키 값을 사용해도 문제없이 다운로드가 가능했다.



[그림 ] 쿠키 값(옥션)

이는 웹 페이지가 단순 헤더만 검사하기 때문으로 판단되는데, 단순 헤더뿐만 아니라 세부 값까지 검사하도록 한다면 좀 더 높은 보안성을 갖출 수 있을 것이다.

* + 1. **캡챠 이미지 상의 취약점**

다음으로는 캡챠 이미지 데이터 자체의 취약점이다. 먼저 지마켓과 옥션의 이미지 데이터를 보면 서로 다른 7가지 색으로 노이즈와 글자가 구성되어 있다.

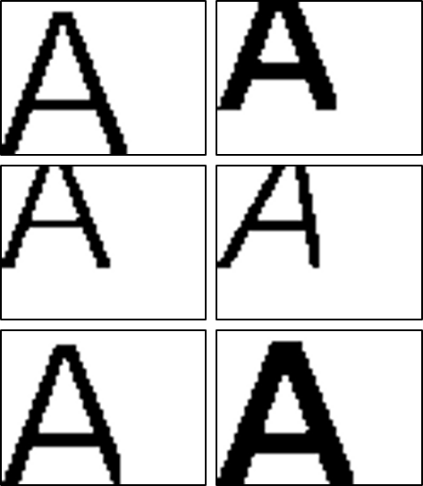


[그림] 캡챠 이미지 원본 (지마켓, 옥션)

이때 노이즈가 단일색으로 구성되어 있으며, 글자와 구별되는 특징들을 가지고 있기 때문에 글자에서 노이즈를 분리해내는 작업이 어렵지 않았다. 또한 각 글자들 또한 단일색으로 구성되어 있기 때문에 각 글자들을 분리하는 작업 역시 어렵지 않게 이루어졌다.



[그림 ] 전처리된 캡챠 이미지 (지마켓, 옥션)

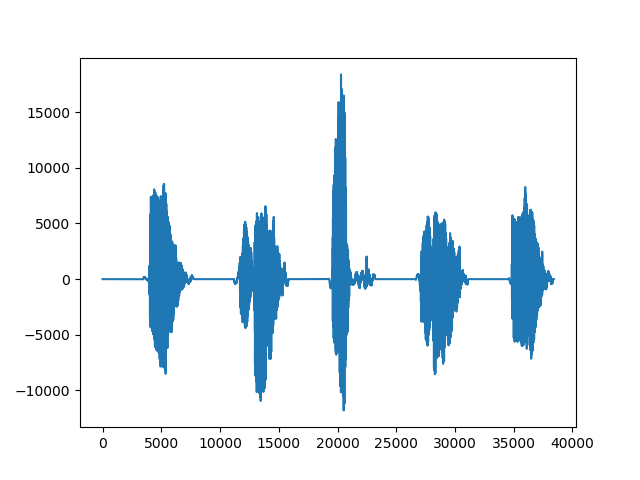


[그림 ] 동일 알파벳

다만, 이미지의 경우 오디오와 달리 같은 알파벳의 경우라도 매번 생성되는 형태가 조금씩 다르기 때문에 정확도가 좀 더 낮게 측정되었다.

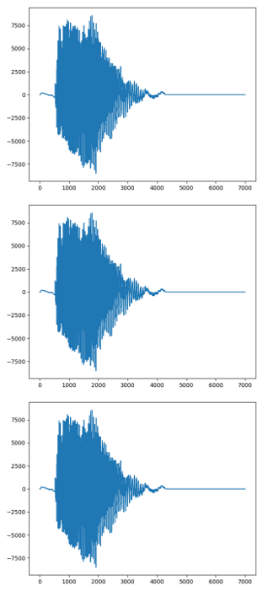
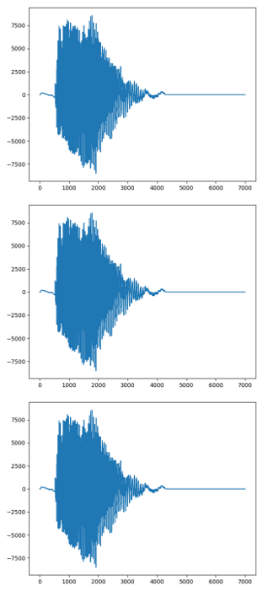
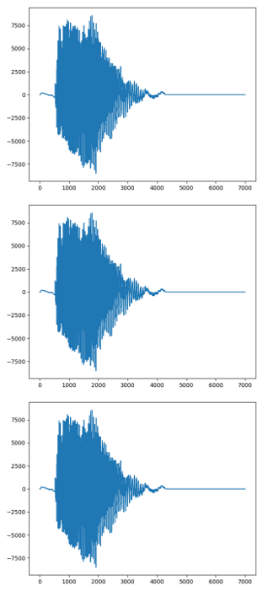
* + 1. **캡챠 오디오 상의 취약점**

다음은 캡챠 음성 데이터 상의 취약점이다. 음성 데이터는 지마켓과 옥션 본인인증 서비스 제공 업체중 하나인 Danal의 데이터를 분석해 보았다.



[그림 ] 캡챠 오디오 원본 (Danal)

이 음성 데이터들을 보면 각 알파벳마다 고정적인 패턴으로 구성되어 있다는 것을 알 수 있으며 또한, 각 알파벳과 알파벳 사이에는 무음(‘0’ 값)이 존재하였다. 이를 통해 손쉽게 각 알파벳 별로 데이터를 분리할 수 있었다. 이처럼 데이터 혹은 노이즈의 패턴이 고정적일수록 더 쉽게 분리하고 노이즈를 제거할 수 있다.



[그림 ] 동일 알파벳

또한, 오디오의 경우 같은 알파벳끼리 오차없이 모두 동일한 형태를 가지고 있었고, 이와 같은 특징들이 인공지능으로 하여금 캡챠 오디오를 쉽게 인식할 수 있게하는 결정적인 요소가 되었다. 따라서 캡챠 오디오를 인공지능으로부터 보다 안전하게 만들기 위해서는 음성 데이터에 배경음이나 잡음과 같은 노이즈를 추가하고 1명 이상의 성우 또는 말투나 목소리가 다른 데이터들을 생성하여 사용하여야 한다.

* 1. **사회적 영향**

본 프로젝트는 국내 사이트의 본인인증 서비스의 신뢰성 검증을 목적으로 하고 있다. 따라서 본 프로젝트를 통해 본인인증 서비스의 발전에 기여하고 국내 본인인증 서비스 연구의 시발점이 되기를 기대하고 있다. 또한, 사회로부터 인공지능의 위험성을 환기시키는 역할을 하기를 기대한다. 마지막으로, 본 프로젝트는 본인인증을 자동우회하는 기능이 있기에 악용될 소지가 있고, 사회에 혼란을 가져올 가능성이 있다.

* 1. **발전 방향**

프로그램을 발전시키기 위해 제안되는 요소들은 다음과 같다. 첫 번째, 현재는 사용자가 프로그램을 사용하기 이전에 각 사이트에 대한 캡챠 정보를 미리 학습하여 학습 데이터를 가지고 있고, 사용자가 프로그램을 실행할 때 불러와 결과를 예측하는 방식이 쓰이고있다. 하지만 캡챠 정보를 예측한 결과를 확인하여 그 결과를 다시 학습 데이터로 피드백하게 된다면 더 높은 정확도를 가질수 있을것으로 예측된다. 두 번째, 현재 본 프로그램은 지마켓과 옥션, 위메프 세 가지 웹 사이트에 대한 검증이 가능하다. 다만, 국내에는 더욱 다양한 웹 사이트들이 있기 때문에 보다 많은 웹사이트들에 대한 검증 또한 가능하도록 프로그램을 확장한다면 프로그램의 실용성이 보다 극대화 될 것이다. 세 번째, 본 프로그램은 현재 크롬 드라이버를 활용하여 크롤링을 구동하기 때문에 크롬을 통해 웹사이트에 접근하였을때만 동작이 가능하다. 하지만 셀레니움 라이브러리는 다양한 웹 브라우저 플랫폼에서 작동이 가능하고, 각 브라우저는 각각의 드라이버를 지원하고 있다. 따라서 프로그램이 크로스브라우징을 지원한다면 사용자는 브라우저에 구애받지 않고 프로그램을 사용할수 있어 사용자 편의성이 증대될 것이다. 마지막으로, 사용자 정보 암호화이다. 현재는 웹페이지 검증용으로 프로그램을 설계하였기 때문에 임의의 사용자 정보를 사용하고, 사용자 정보또한 평문형태로 저장하고 있다. 하지만 프로그램을 상용화하게 된다면 보안성을 위해 사용자 정보를 암호화하여 저장해야 할 것이다. 정리하면 아래와 같다.

* 예측결과의 학습 모델 사용을 통한 정확도 증가
* 프로그램 사용가능한 웹페이지 확대
* 다른 브라우저 사용가능하도록 크로스브라우징 지원
* 사용자 정보 암호화

# **References**

[01] 보안뉴스, “[구글의 웹사이트 보호 장치인 리캡챠, 다시 한 번 깨지다](https://www.boannews.com/media/view.asp?idx=75945)”, 2019.01

[02] github, <https://github.com/ecthros/uncaptcha2>, 2019.01

[03] 위키피디아, <https://ko.wikipedia.org/wiki/CAPTCHA>

[04] 한국정보화진흥원, 인공지능 악용에 따른 위협과 대응 방안

[05] SCD, <https://study.com/academy/lesson/system-context-diagram-description-examples.html>

[06] 웹크롤러, <https://beomi.github.io/2017/02/27/HowToMakeWebCrawler-With-Selenium/>

[07] <https://www.nia.or.kr/common/board/Download.do?bcIdx=20146&cbIdx=82618&fileNo=1>

[08] <http://nownews.seoul.co.kr/news/newsView.php?id=20180222601022>

[09] <http://news.kbs.co.kr/news/view.do?ncd=3601037>