Web Crawling

머신러닝 기반 데이터 분석, 예측 파트 진행 순서

1) 분석 및 예측

시각화, 머신러닝:
- python
- numpy, Pandas
- Matplotlib,Seaborn
- 기초통계
- Scikit learn

- 탐색적 데이터 분석 방법으로 데이터를 분석함
- 분석 데이터를 시각화 하는 방법을 익힘
- 머신러닝 이해 하고 사이킷런 활용해 다양한 머신러닝 모델을 만들고 평가하는 방법 적용

2) Web pgm 기본

Front end side :
- HTML5
- CSS3
- Javascript
- jQuery

- 웹에 산재되어 있는 데이터를 수집, 분석하기 위한 웹 문서 표현 기술인 웹 표준 활용 능력 익힘.
- 웹 데이터의 구조 이해

3) 데이터 저장

Back end side:
- Django : python 기반 web server 프레임워크
- Mysql - CRUD
- MongoDB(js기반)

- 데이터를 구조화 하여 저장하는 방법 적용
- 정형, 비정형 데이터 유형을 이해하고, 저장하는 방법 적용
- 클라우드 서비스 이해, 프리티어 서비스를 활용해 웹서비스 구현

4) 데이터.수집가공

웹 크롤링&스크래핑:
- Python 기반
  - BeautifulSoup
  - Selenium
- 머신러닝 통합 예제
- Linux shell pg

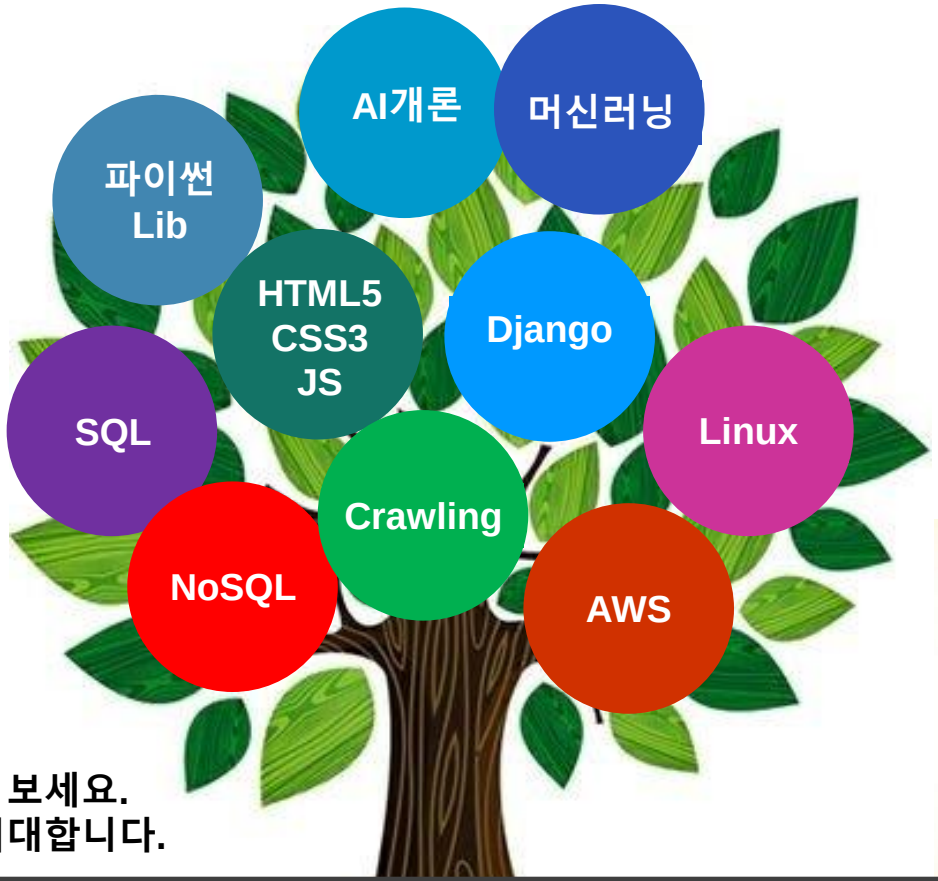
- 웹크롤링 및 스크래핑 기술을 적용
- 웹에 산재되어 있는 데이터를 수집, 가공, 파일로 저장하는 방법 활용
- 데이터 분석을 위해 전처리 방법을 익힘

5) 팀 협업 프로젝트

- 의미 있는 도출을 위한 팀 주제 정하기
- 웹크롤링, 오픈데이터
- 데이터 DB 저장
- 데이터 분석, 시각화
- 머신러닝 예측
- 웹 서비스로 구현하기

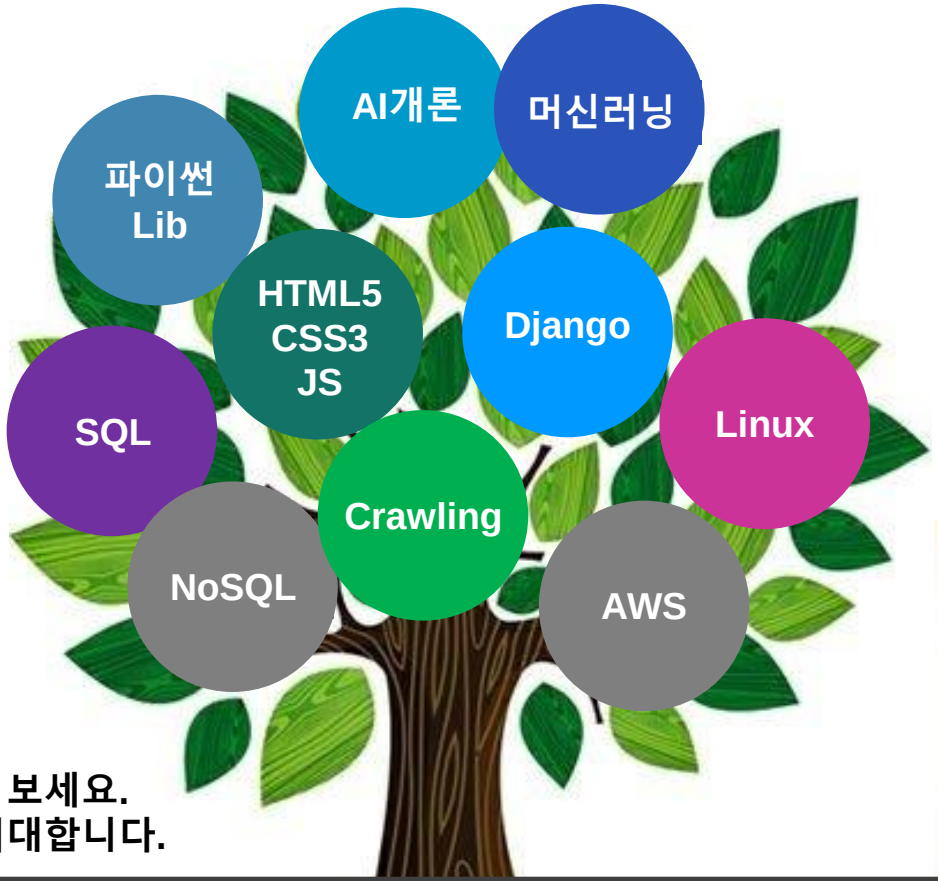
클라우드 서비스
AWS

데이터 분석 강의 내용(10.5~11.16)



어떤 씨앗을 심을지 고민해 보세요.
모두 좋을 결실이 있기를 기대합니다.

데이터 분석 강의 내용(10.5~11.16)



어떤 씨앗을 심을지 고민해 보세요.
모두 좋을 결실이 있기를 기대합니다.

학습 내용

1. 웹크롤링 기초
2. 정적 크롤링
3. 동적 크롤링
   - browser 제어
   - selenium

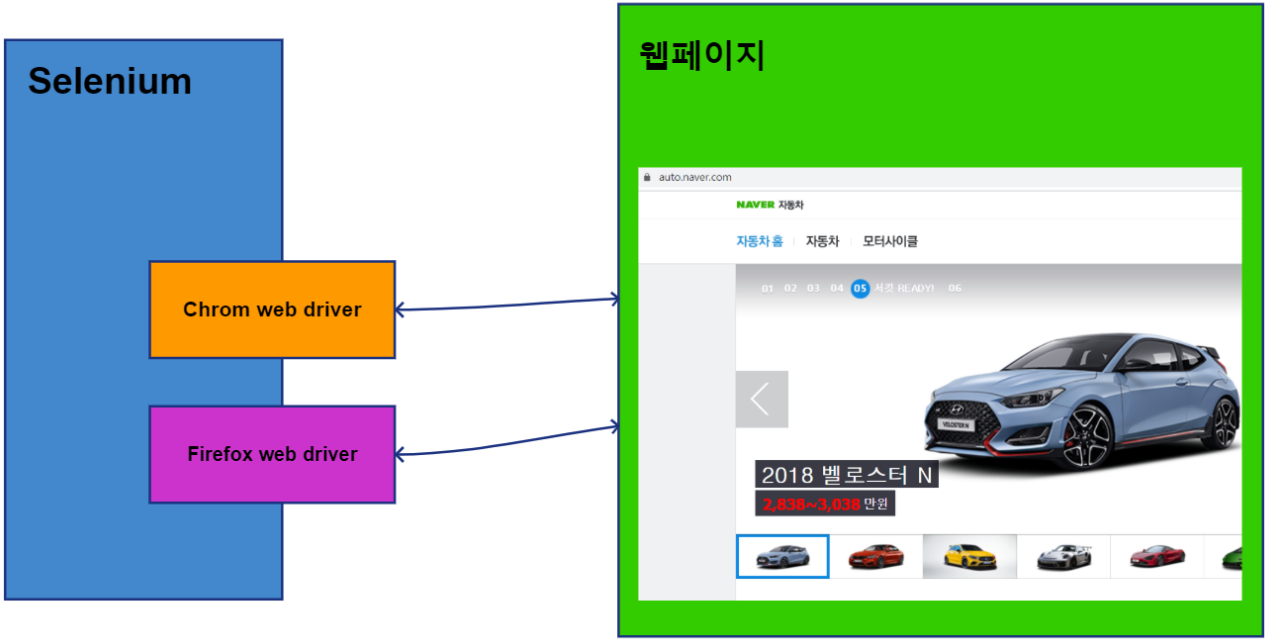
동적 크롤링

정적 크롤링 vs. 동적 크롤링

	정적 크롤링	동적 크롤링
크롤링 속도	빠르고 간단함	느리고 복잡함
개발 편의성	처음엔 쉽지만 고도화 어려움	처음엔 손이 많이 가지만 나중엔 편리함
디버깅 편의성	테스트 쉬움	테스트 어려움
오류 취약점	상대적으로 낮음	상대적으로 높음

Selenium 원리 및 기능

- python 크롤링 시, 동적인 동작을 곁들여서 크롤링 할 수 있도록 도와주는 라이브러리

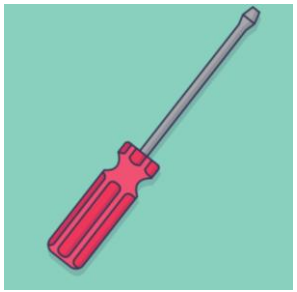


Selenium 원리 및 기능

- html 문서의 특정 html 요소를 마우스 클릭을 발생시킬 수 있음.
  - 게시판 페이지를 크롤링 한뒤, 다음 페이지 버튼을 찾아서 마우스 클릭하여 다음페이지로 이동하여 크롤링 가능
- input 엘리먼트에 텍스트를 채워 넣기 가능
- web driver인 가상 브라우저와 연동하여 기능 구현 함

동적크롤링 – 셀레니움(크롬 기준)

- 크롬 브라우저 설치
- 크롬 driver 다운로드
- 셀레니움 파이썬 라이브러리 설치



동적크롤링 – 셀레니움(크롬 기준)

크롬 Driver 역할

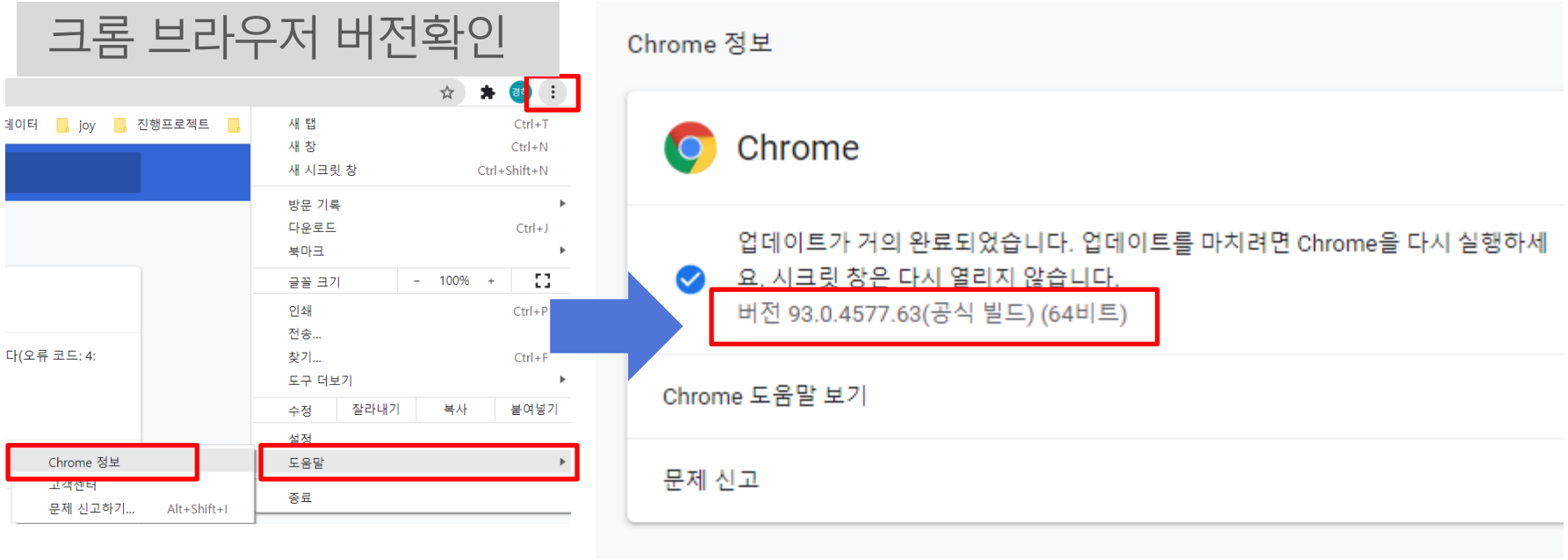
- 브라우저와 셀레니움 간의 통신
- 주의 사항 : **브라우저 버전과 driver 버전을 맞춰야 함**
- 시스템의 크롬 브라우저 버전 확인

$ google-chrome --version

```
(crawling) himedia@himedia:~/bigdata$ google-chrome --version
Google Chrome 91.0.4472.114
(crawling) himedia@himedia:~/bigdata$
```

동적크롤링 – 셀레니움(크롬 기준)

- 크롬 버전 확인 – 브라우저 [도움말-chrome 정보]



동적크롤링 – 셀레니움(크롬 기준)

- chrome driver 설치

  https://chromedriver.chromium.org/downloads

  크롬 브라우저의 버전에 맞는 드라이브 다운로드,
  압축 풀기, 작업 폴더로 복사

- selenium library 설치(python 가상환경)

  $ pip install selenium

동적크롤링 – 셀레니움 테스트

- chrom webdriver manager
  selenium crawling 시 **실시간으로 브라우저 버전을 맞춰 줌**
  $ pip install webdriver-manager (가상환경에 설치)

```
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager

driver = webdriver.Chrome(ChromeDriverManager().install())
time.sleep(3)
chrome.close()
```

동적크롤링 – 셀레니움 테스트

- selenium을 활용한 브라우저 제어 예시

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from webdriver_manager.chrome import ChromeDriverManager
import time
chrome = webdriver.Chrome(ChromeDriverManager().install())
chrome.get(http://daum.net)

elem = chrome.find_element_by_class_name("link_login")
elem.click()
chrome.find_element_by_class_name("btn_g").click()
chrome.back()
time.sleep(2)
chrome.forward()
time.sleep(2)
```

동적크롤링 – 셀레니움 테스트

```
chrome.back()
time.sleep(2)
elem = chrome.find_element_by_name("q")
elem.send_keys("사과")
elem.send_keys("바나나")
chrome.find_element_by_id("q").clear()
elem.send_keys("사과")
elem.send_keys(Keys.ENTER)

items = chrome.find_elements_by_class_name("thumb_img")
for item in items:
    print(item.get_attribute("src"))
chrome.close()
```

동적크롤링 – 셀레니움 테스트

- webdriver.Chrome() options 지정

```
from selenium import webdriver
import time  # 셀레니움 실행 시 기다려야하는 시간들이 있음.

# 크롬 옵션 주기, 크롬을 실행시킬 때 브라우저 함수 실행
options = webdriver.ChromeOptions()     # 옵션 객체 생성
options.add_argument("window-size=1000,1000")  # 실행 윈도우 크기
options.add_argument("no-sandbox") # 텝 간에 분리 함
chrome = webdriver.Chrome("./chromedriver", options=options)
chrome.get("http://naver.com")     # 브라우저로 url 실행
time.sleep(3)
chrome.close()
```

동적크롤링 – 셀레니움

- 로딩 시 기다리는 여러 방법들

```
from selenium import webdriver
import time  # 셀레니움 실행 시 기다려야하는 시간들이 있음.
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC

chrome = webdriver.Chrome("./chromedriver")
chrome.get("http://naver.com")    # 브라우저로 url 실행
time.sleep(3)                     # python time 라이브러리
chrome.implicitly_wait(3)    # 크롬드라이브와 통신하는 지점에서 delay
# 지정 요소가 로딩 될 때까지 기다림(예시 : 최대 10초 기다림)
WebDriverWait(chrome, 10).until(EC.presence_of_element_located((By.CSS_SELECTOR,
"input[name=query]")))
chrome.close()
```

웹페이지 load 타임 라인

- 웹 브라우저 요청 실행
- 웹서버가 HTML 응답
- HTML 그리기
- HTML 그리기 + CSS 적용
- JavaScript 실행, html, css, js 와 동급
- 추가 요소에 적용 되어 있는 js 실행

동적크롤링 – 셀레니움 실습

- element(요소)를 찾기
- 지정 selector의 모든 요소를 리턴함

find_elements_by_css_selector("selector")

- 지정 selector의 요소 1개를 리턴함.

find_element_by_css_selector("")

자동로그인을 구현하기 위해 필요한 라이브러리

- 캡차를 피하기 위해 복사해서 붙여넣기 기능 구현시 필요

```
# 복사 붙여넣기 python 라이브러리
$ pip install pyperclip

# 시스템에서 복사해서 붙여넣기 기능 사용 가능
$ sudo atp-get install xsel
```

동적크롤링 – 셀레니움 실습

- selenium 기본 실습
- 구글 이미지 모으기
  - 키보드 키워드, 개수 입력받기
- 네이버 쇼핑 로그인
- 필요한 쇼핑 목록 가져오기

- [문제 해결] 네이버 로그인 후, 자신의 메일 목록 가져오기

동적크롤링 – 셀레니움 실습

- 메일 목록 가져오기 할 때 필요한 메소드

html 문서내에 iframe이 있을 경우 :
iframe으로 swithching 하여 필요한 작업 후,
처음 문서로 다시 switching 진행

- iframe으로 swithching하는 메소드

  webdriver.switch_to.frame("iframe name") # iframe_name : name 속성 값

- 처음 문서로 되 돌아가는 메소드

  webdriver.switch_to.default_content()