2019

산학 캡스톤 디자인
중간보고서

1 조 FASHION PEOPLE

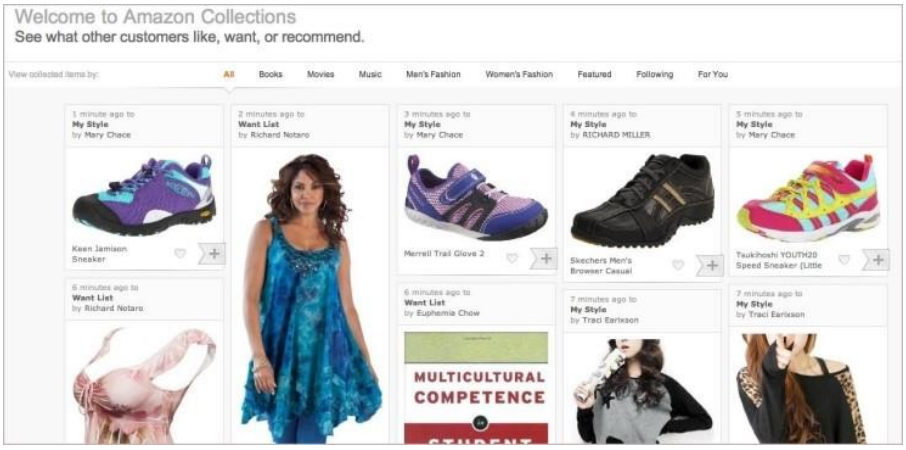
천승현, 이유겸, 박지혜, 최예솜

목차

1.개발동기

1.1 동기

추천 시스템(recommend system)은 현대 상업시장에서 가장 강력하고 필수적인 전략 중 하나이다. 사용자(User)는 모든 아이템들을 다 둘러볼 시간이 없고, 자신에게 가장 필요로 하는 상품을 추천 해 주길 원한다. 추천 시스템을 적용한 웹 쇼핑사이트들은 자신들의 알고리즘을 이용하여 사용자들에게 적합한 아이템을 추천해준다. 사용자들이 기존에 구매했던 상품들의 속성이나, 평점을 분석하여 관련 있는 상품들을 추천 리스트로 작성해 보여준다.



추천시스템을 가진 회사중 가장 유명한 회사는 단연 아마존이다. 아마존은 회원들의 소비 패턴을 분석해 구매 가능한 상품을 추천하는데, 아마존 성장의 일등 공신으로 아마존 매출의 약35%가 추천 상품에서 발생한다. 아마존은 고유의 아이템기반 협업필터링 알고리즘을 'A9'이라 부르고 특허를 등록하였다.

이러한 추천 시스템을 적용한 웹시스템을 개발하기 위해 빅데이터 분야의 요소를 접목시킨다. 빅데이터 수집단계를 통해 데이터를 수집하고, 분석단계를 통해 데이터를 파싱, 정규화 하며 활용단계에서 수집한 데이터들을 이용하기로 한다. 이렇게 빅데이터의 요소를 웹 사이트와 알고리즘을 구현하며 직접 학습해보기로 한다.

1.2 개발제목

20대 남녀를 위한 패션 코디 추천 웹 사이트 개발 (FCMW)

Fashion coordy
for 20's man & woman
in Website

- 데이터를 수집하기 용이하게 남녀 20대라는 키워드를 특정하기로 하고 프로젝트 제목을 요약한 축약어로 FCMW를 사용하기로 한다.

2. FCMW개발내용

2.1 패션 데이터 선정

빅데이터 수집

- 수작업

빅데이터 수집을 위한 방법으로 하나인 수작업으로 데이터를 수집하는 법은 FCMW에서 필요한 상의, 하의, 신발 등의 사진 등을 직접 검색 엔진, 여러 사이트에 접속하여 직접 사진을 다운로드 받아 데이터로써 변형 작업을 거치는 방법이다. 하지만 FCMW에 맞는 데이터를 수집을 위해서는 수작업은 불가능 할 것으로 판단하여 새로운 방법인 웹 크롤러라는 방식으로 대체한다..

- 웹 크롤러

웹 크롤러(Web Crawler)는 조직적, 자동화된 방법으로 월드 와이드 웹을 탐색하는 컴퓨터 프로 그램이다. 웹 크롤러가 하는 작업을 '웹 크롤링'(Web Crawling) 혹은 '스파이더링'(Spidering)이라 부른다. 검색 엔진과 같은 여러 사이트에서는 데이터의 최신 상태 유지를 위해 웹 크롤링한다. 웹 크롤러는 대체로 방문한 사이트의 모든 페이지의 복사본을 생성하는 데 사용되며, 검색 엔진은 이렇게 생성된 페이지를 보다 빠른 검색을 위해 인덱싱한다. 또한 크롤러는 링크 체크나 HTML 코드 검증과 같은 웹 사이트의 자동 유지 관리 작업을 위해 사용되기도 하며, 자동 이메일 수집과 같은 웹 페이지의 특정 형태의 정보를 수집하는 데도 사용된다. FCMW에서 사용하게 될 데이터 수집을 위해 구글에서 패션 코디와 관련 있는 데이터를 크롤링한다.

빅데이터 저장 및 분석

-ImageAI를 이용하여 분석한 정보를 DB에 저장

-SQLite3

DB에 저장하기 위해 사용할 SQLIite3는 별도의 서버 프로세스가 필요 없고 SQL질의 언어의 비표준 변형을 사용하여 데이터베이스의에 액세스할 수 있는 경량 디스크 기반 데이터베이스를 제공하는 C 라이브러리이다. 일부 응용 프로그램은 내부 데이터 저장을 위해 SQLite를 사용할 수 있습니다. SQLite를 사용하여 응용 프로그램을 프로토타입 한 다음 PostgreSQL 이나 Oracle과 같은 더 큰 데이터베이스로 코드를 이식할 수 있다. 또한 django에서 지원하는 기본 DB는 sqlite3이다.

https://www.sqlite.org/index.html

-ImageAI

DeepQuiest AI 사의 CEO인 Moses Olafenwa가 자신의 형제와 함께 개발한 통합 라이브러리로 자유명한 이미지 인식 network(SqueezzeNet 캘리포니아 버클리 대학, 스탠포드 대학, ResNet50 마이크로소프트, InceptionV3 구글, DenseNet121페이스북)들을 한 곳으로 묶어 약 1400만 이미지를 미리 트레이닝하여 사용할 수 있고 자신만의 이미지로 학습시켜 나만의 모델을 생성 가능 하게 한다.

http://imageai.org/

https://github.com/OlafenwaMoses/ImageAI

시각화

-ANACONDA PYTHON Ver 3.7

오픈 소스인 ANACONDA는 Python/R data science 그리고 머신 러닝을 윈도우나 리눅스 맥OS 등에서 가장 쉽게 사용할 수 있게 해주는 프로그램으로 데이터를 분석하여 시각화 하기 위해 사용한다.

https://www.anaconda.com/distribution/

-주피터 노트북(Jupyter Notebook)

주피터 노트북은 크롬이나 마이크로소프트 엣지, 익스플로어 같은 웹 브라우저환경에서 코드를 작성하고 실행을 가능하게 해주는 워킹 툴이다. 파이썬이나 아나콘다를 설치시 기본적으로 동시에 주피터 노트북이 설치되고 인터프리터 대화형 형식의 프로그램으로 실시간으로 데이터의 변화와 코드의 결과를 확인할 수 있다.

https://jupyter.org/

- 데이터 크롤링 방법

FCMW 웹 구현을 위해 필요한 사진 데이터들을 가져오기 위하여 선택 한 웹 크롤링 방식을 사용하기 위해 ANACONDA Python Ver 3.7에서 코드를 작성해야 한다.

```
from google_images_download import google_images_download   #importing the library

response = google_images_download.googleimagesdownload()   #class instantiation

arguments = {"keywords":"Polar bears","limit":5,"output_directory":'image',"print_urls":True}   #creating list of arguments
paths = response.download(arguments)   #passing the arguments to the function
print(paths)   #printing absolute paths of the downloaded images
```

구글에서 이미지를 받아오는 방법인 웹 크롤러를 사용하기 위해 파이썬 코드를 작성한다.

- 데이터 분석, 활용 방법

데이터 분석

웹 크롤러를 통해 구글에서 받아온 의류의 종류에 따라 크롤링 해온 비정형 데이터(unstructured data, 유동 데이터, 미리 정의된 데이터 모델이 없거나 미리 정의된 방식으로 정리되지 않은 정보를 말한다. Ex. 동영상, 사진, SNS의 텍스트) 이러한 비정형데이터를 사용하게 되면 프로그램을 사용하여 이해하는 것을 불가능하게 만든다. 이러한 이유 때문에 정형 데이터(기존 데이터베이스에 저장이 되는 '규격화'된 데이터 Ex. 엑셀파일, DB테이블)로 변환이 필요하다.

데이터 활용

데이터 분석 과정을 통해 얻은 FCMW 웹 구현을 위한 정현 데이터인 의류나 신발 데이터를 의류 중에서도 원피스, 셔츠, 팬츠, 티셔츠 등의 키워드를 통해 FCMW 웹에서의 사용자의 취향에 맞는 의류나 신발 등을 보여 줄 수 있다.

2.2 api

Api란

Application Programming Interface(응용 프로그래밍 인터페이스)로, 프로그램에서 사용할 수 있도록 운영 체제나 프로그래밍 언어가 제공하는 기능을 제어 할 수 있게 만든 인터페이스이다. 패션 이미지들은 비정형데이터로, 용량이 크기 때문에 그대로 사용하기에는 무리가 있으므로 이미지 분석 API를 이용하여 다루기 쉬운 데이터로 변환한다.

Google cloude vision api는

구글 포토에서 쓰이고 있는 서비스로, 머신 비전(Machine Vision) 즉, 머신 러닝(기계학습)을 이용한 이미지 인식 기술의 한 종류이며 구글에서 개발되었다. 또한, 누구나 API를 사용해 자신의 앱, 서비스, 하드웨어에 구글의 기술을 적용 가능하다. 클라우딩 기반으로 모바일이나 성능이 낮은 디바이스에서 구현 가능하며, 높은 수준의 인식률로 오류 비율이 사람이 분류했을 때와 비슷한 수준인 6.65% 으로 사람의 얼굴 이미지에서 감정 읽어내기, 브랜드,언어, 경치, 인공 구조물, 부적절 컨텐츠 인식 가능하다. 그러나 기본으로 제공되는 정도를 사용하면 유료로 변환이 된다는 점 그리고 기능을 구현하기 위한 함수 코드의 복잡성을 가지고 있으며 필요하지 않은 기능들이 많다는 단점을 가지고있다.

ImageAI 

DeepQuiest AI 사의 CEO인 Moses Olafenwa가 자신의 형제와 함께 개발한 통합 라이브러리 이다. 유명한 이미지 인식 network들을 한곳으로 묶어 사용하기 쉽게 만들어 놓았다. ImageAI의 학습 모델은 캘리포니아 버클리 대학, 스탠포드 대학에서 사용한 SqueezzeNet, 마이크로소프트사에서 사용한 ResNet50, 구글에서 사용한 InceptionV3, 페이스북엣 사용하는 DenseNet121가 있다. 1400만 이미지를 미리 트레이닝 시킨 모델을 사용할 수 있으며, 자신만의 이미지로 학습시켜 나만의 모델을 생성가능하다.

2.3 FCMW 웹 사이트

- django를 이용한 웹사이트 개발

-Django

장고는 파이썬으로 작성된 오픈 소스 웹 애플리케이션 프레임워크로 FCMW 웹 개발에 있어서 번거로운 요소들을 새로 개발할 필요 없이 내장된 기능만을 이용해 빠른 개발을 할 수 있기에 사용한다.

데이터 시각화는 FCMW 코디 분석 결과를 쉽게 이해할 수 있도록 시각적으로 표현하고 전달되는 과정을 말한다. FCMW데이터 시각화의 목적은 최종 코디 결과 사진을 통해 명확하고 효과적으로 전달 하는 것이다.

https://www.djangoproject.com/

-사용하는 데이터

FCMW 웹 구현을 위해서는 20대 남녀를 위한 코디를 추천하기 위해 방대한 양의 종류별 의류의 사진, 신발 등의 사진을 필요로 한다. 의류에 대한 데이터로는 의류의 종류(ex.셔츠, 티셔츠, 후드티셔츠, 맨투맨, 블라우스, 니트, 가디건, 코트, 재킷) 이러한 카테고리 등으로 나누어 데이터를 수집한다. 수집한 데이터를 FCMW 웹 구현을 위해 사용한다.

3. FCMW 세부내용

3.1 일정표

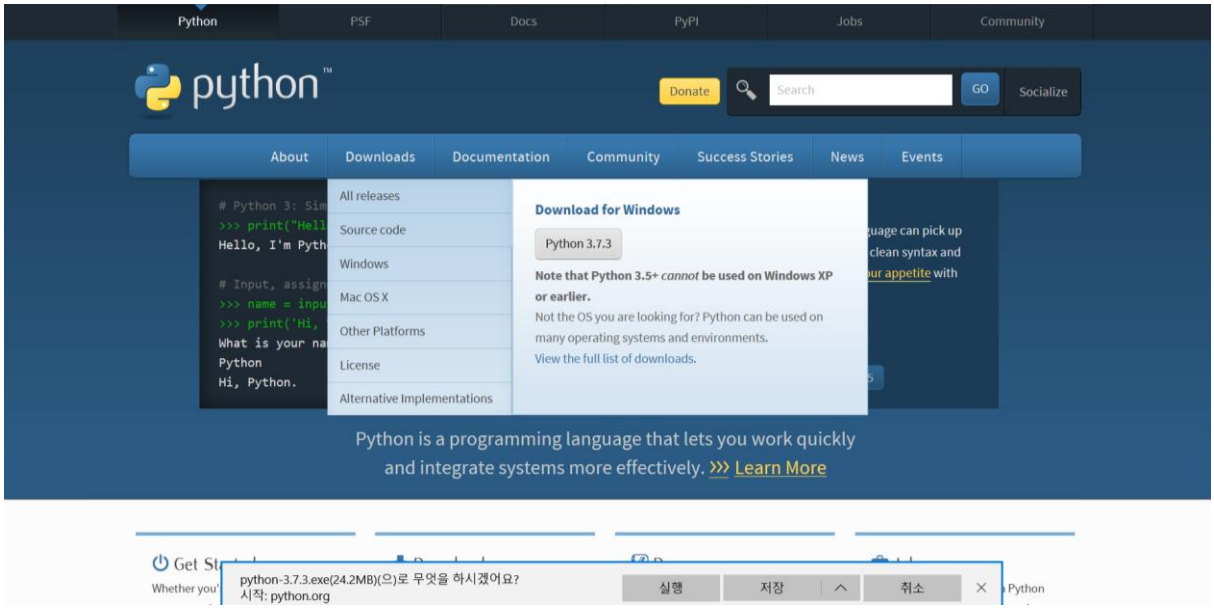
- 일정표

	1주	2주	3주	4주	5주	6주	7주	8주	9주	10주	11주	12주	13주	14주	15주
자료 조사															
정규 토의															
제안서,보고서 작성															
개념 설계															
상세 설계															
1차 구현															
2차 구현															
테스트 및 보완															
리허설 및 유지보수															
비고			주제 발표					중간 발표		1차 구현 발표		2차 구현 발표			최종 발표

3.2 개발환경

- 파이썬 설치 

파이썬 -> 플랫폼이 독립적이고 인터프리터식으로 만들어져 있으며, 객체지향적, 동적 타이핑 대화형 언어로 사용된다.데이터분석에 특화된 언어이며 ,장고를 분석해 웹사이트를 만들 때도 파이썬을 사용하기 때문에 설치를 해준다.
설치 방법은 파이썬 다운로드링크는 https://www.python.org/ 이고,링크에 들어가

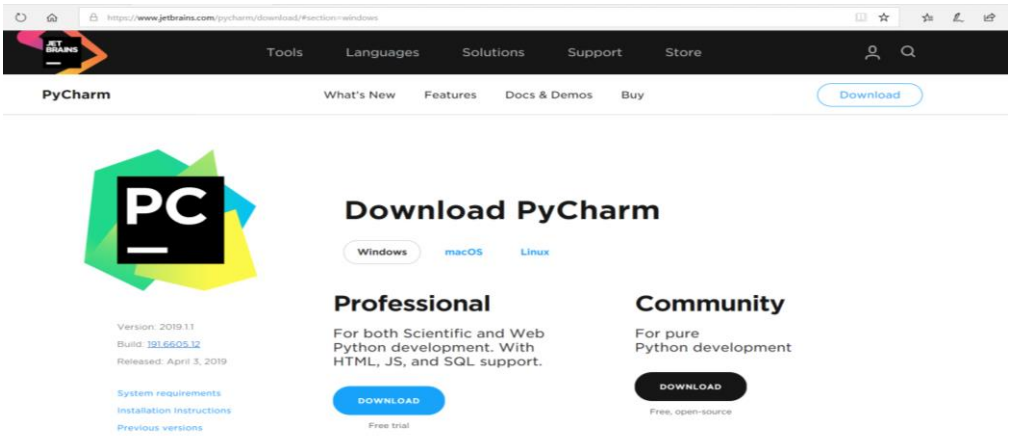


위 그림과 같은 페이지에 접속된다.접속후 downloads 카테고리를 선택한후 자신의 환경을 선택하여 python 3.7.3 을 클릭하여 파이썬을 실행시켜준다.

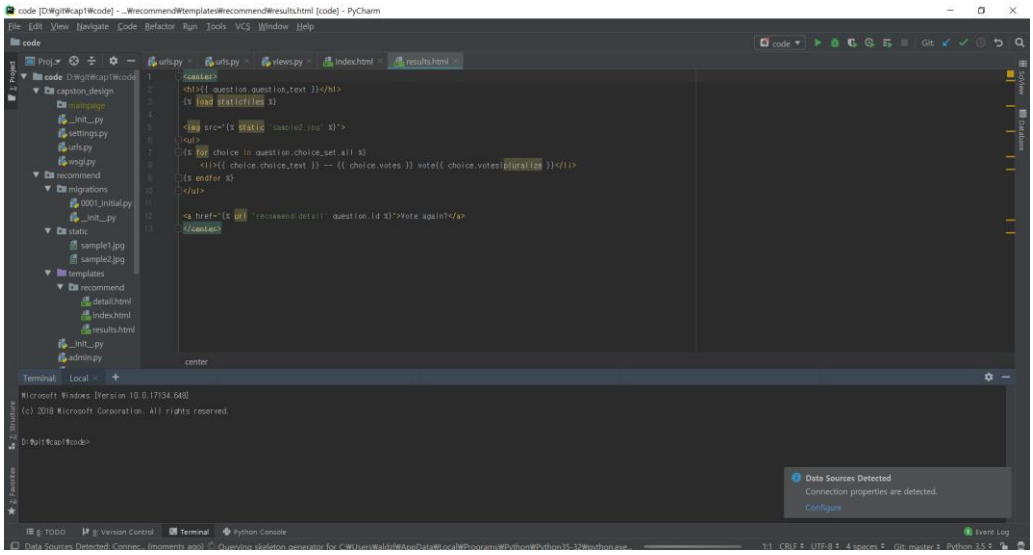
- 파이참 설치

파이참은 Python 개발에 필요한 모든 도구를 제공해 주는 역할을 한다.파이참은 다운로드시 별도의 제약 없으며 다운로드후 ->power shell 에서 장고 설치하기위해 pip install Django 를 입력하며 개발환경을 만들어준다.

다운로드 방법은 https://www.jetbrains.com/pycharm/download/#section=windows 링크에 접속하여 화면에 보이는Download를 클릭하여 Community Download를 눌러준다. Professional Download를 누를시 시간이흐른뒤 유료화 되기 때문에 community를 다운받아준다.

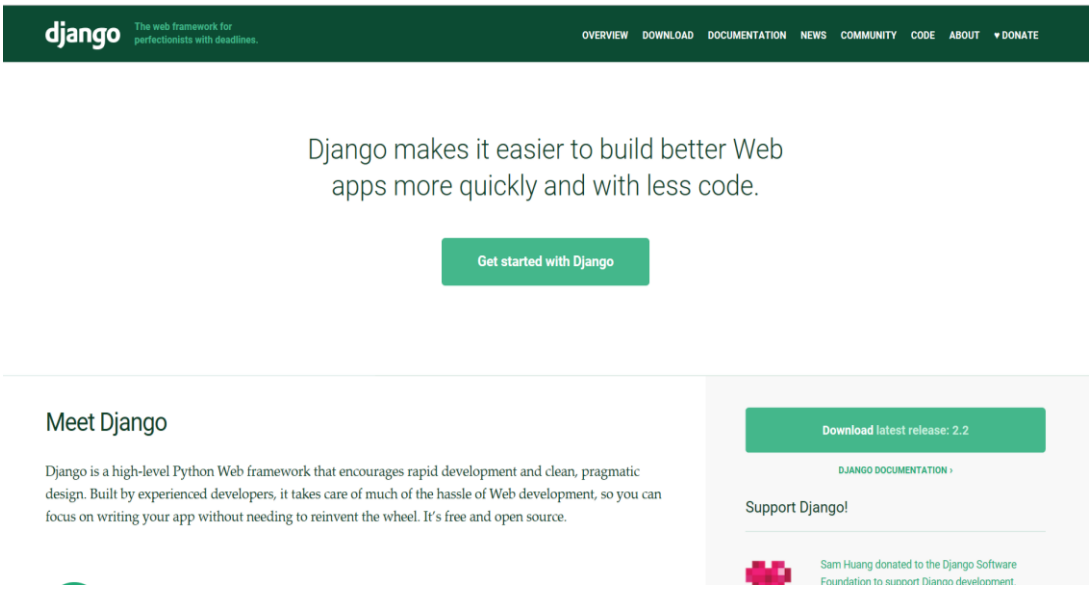


다운로드가 완료되고 실행시키면 pycharm 실행화면이 나온다.



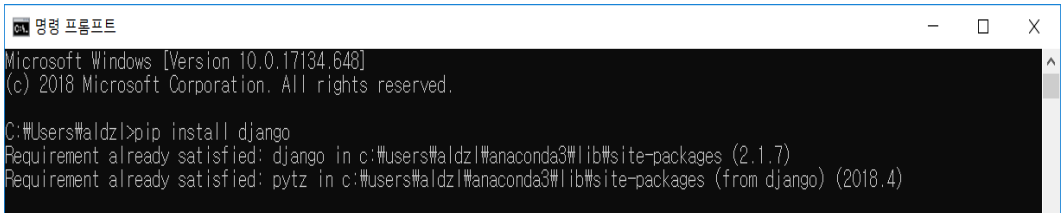
- 장고

장고는 파이썬 웹 프레임워크로 파이썬을 이용하여 웹페이지를 쉽게 구성하기 위해 사용하였다. 장고 다운로드를 위해 https://www.djangoproject.com/ 링크를 타고 들어가면,



위와 같은 화면을 볼수 있다.

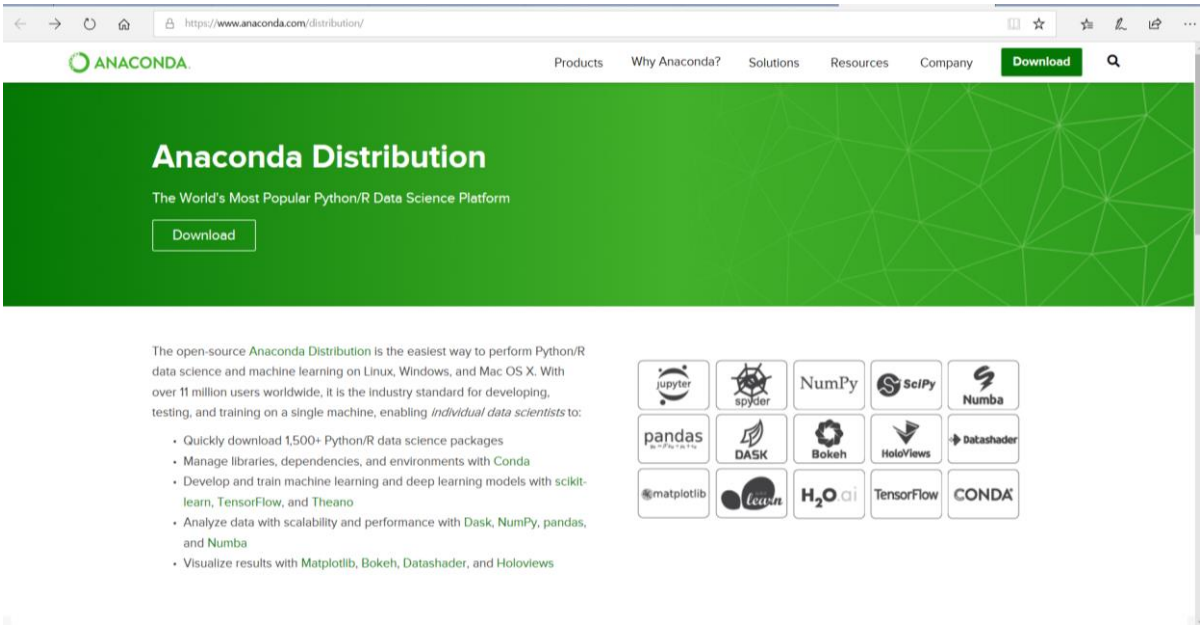
get started with Django를 클릭한 후, Download version 2.2를 클릭하여 다운받아 준다. 또다른 방법으로는 cmd창 에서 사진과 같이 >pip install django명령어를 입력해주면 다운로드 홈페이지에 접속하지 않아도 바로 설치하여 사용 가능하다.



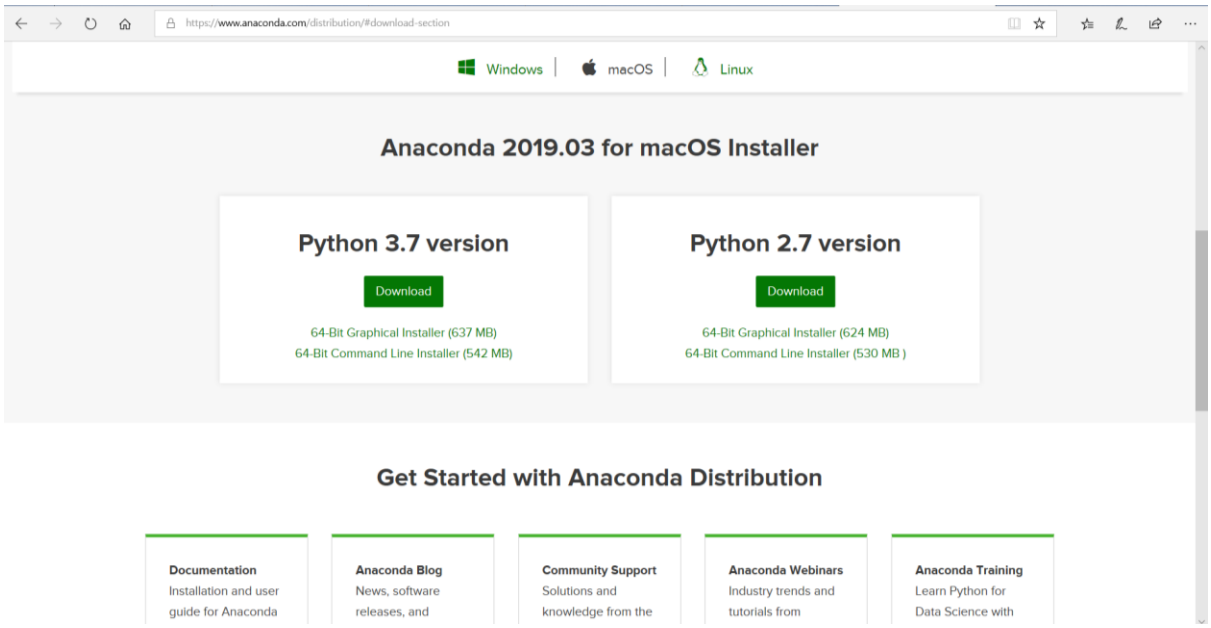
이미 설치되어 있으면 cmd창에 Requirement already satisfied 라는 결과가 나오게 된다.

- 아나콘다 네비게이터 ,

Anaconda 는 데이터 분석용으로 사용되며, 패키지관리를 단순화하고 과학 컴퓨팅을위한 Python 및 R 프로그래밍 언어를 위한 무료 및 오픈소스 배포판이다. **아나콘다 설치를 위해 https://www.anaconda.com/distribution/ 링크에 클릭해주면,** 위와 같은 페이지에 접속이 된다.



여기서 Download버튼을 클릭 하고 각자 필요한 version을 클릭하여 다운로드하여 설치해준다



3.3 개발상황

- 3.3.1 개발 현황

프로젝트를 진행하기 위해 필요한 프로그램을 설치하고 환경을 구성한다. 이후에 데이터를 수집하고 활용하기 위해 파이썬을 선택한다. 파이썬은 데이터를 분석하는데 특화된 프로그래밍 언어로 다양한 패키지를 제공하고 있다. 우선 아나콘다 네비게이터에서 제공하는 주피터 노트북으로 데이터를 관리하고 필요한 패키지를 설치해서 함수를 작성한다. 코디 사진을 수집하기 위한 웹 크롤러 패키지를 설치하고, 코디 이미지를 분석하기 위한 이미지 분석 api를 설치한다.

django는 파이썬으로 웹 사이트를 제작하기 위해 제공하는 프레임워크이다. 파이썬 특화 통합개발환경인 파이참으로 django패키지를 다운받고 웹 사이트 프로토타입을 제작한다.

- 3.3.2 코드

(1)image_train

```
#conda install tensorflow
!pip3 install --upgrade tensorflow
!pip3 install numpy
!pip3 install scipy
!pip3 install opencv-python
!pip3 install pillow
!pip3 install matplotlib
!pip3 install h5py
!pip3 install keras
!pip3install https://github.com/OlafenwaMoses/ImageAI/releases/download/2.0.2/imageai-2.0.2-py3-none-any.whl
```

```
from IPython.display import Image
Image(filename='image/1.jpg')
```

```
from IPython.display import Image
Image(filename='image/1.jpg')
from imageai.Prediction import ImagePrediction
import os
execution_path = os.getcwd()
prediction = ImagePrediction()
prediction.setModelTypeAsResNet()
prediction.setModelPath(os.path.join(execution_path,
"resnet50_weights_tf_dim_ordering_tf_kernels.h5"))
prediction.loadModel()
```

```
predictions, probabilities = prediction.predictImage(os.path.join(execution_path, "1.jpg"), result_count=5 )
for eachPrediction, eachProbability in zip(predictions, probabilities):
    Image(filename='1.jpg')
    print(eachPrediction , " : " , eachProbability)
```

(2)image_crawler

```
!pip install google_images_download

from google_images_download import google_images_download   #importing the library

response = google_images_download.googleimagesdownload()   #class instantiation

arguments = {"keywords":"셔츠","limit":10,"output_directory":'image',"print_urls":True}   #creating list of arguments
paths = response.download(arguments)   #passing the arguments to the function
print(paths)   #printing absolute paths of the downloaded images
```

(3)djagno website

code/cpaston_design/settings.py

```
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '7wet!$zj9t-mbf=)1@#*@nlr-ez*%cd4@kw0^1unzngqr)s#45'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['*']


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'recommend.apps.RecommendConfig',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'capston_design.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
```

```
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'capston_design.wsgi.application'


# Database
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}


# Password validation
# https://docs.djangoproject.com/en/2.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/2.1/topics/i18n/

LANGUAGE_CODE = 'ko-kr'

TIME_ZONE = 'Asia/Seoul'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.1/howto/static-files/

STATIC_URL = '/static/'
```

```
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'recommend','static')
]

STATIC_ROOT = os.path.join(BASE_DIR, 'static')
```

code/cpaston_design/urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('recommend/', include('recommend.urls')),
]
```

code/cpaston_design/wsgi.py

```
import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'capston_design.settings')

application = get_wsgi_application()
```

code/recommend/migrations/0001_initial.py

```
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Choice',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('choice_text', models.CharField(max_length=200)),
                ('votes', models.IntegerField(default=0)),
            ],
        ),
        migrations.CreateModel(
            name='Question',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('question_text', models.CharField(max_length=200)),
                ('pub_date', models.DateTimeField(verbose_name='date published')),
```

```
            ],
        ),
        migrations.AddField(
            model_name='choice',
            name='question',
            field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to='recommend.Question'),
        ),
    ]
```

code/recommend/templates/recommend/detail.html

```
<center>
<h1>{{ question.question_text }}</h1>

{% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif %}

{% load staticfiles %}

<img src="{% static 'sample1.jpg' %}">

<form action="{% url 'recommend:vote' question.id %}" method="post">
{% csrf_token %}
{% for choice in question.choice_set.all %}
    <input type="radio" name="choice" id="choice{{ forloop.counter }}" value="{{ choice.id }}" />
    <label for="choice{{ forloop.counter }}">{{ choice.choice_text }}</label><br />
{% endfor %}
<input type="submit" value="Vote" />
</form>
</center>
```

code/recommend/templates/recommend/index.html

```
<html lang="en">
<center>
<head>
    <meta charset="UTF-8">
    <title>Fashion coordy recommendation for 20's men and girls</title>
</head>
    </center>
<center>
<body>
<h1>Fashion People</h1>
<p> Fashion coordy recommendation for 20's men and girls in Website</p>
<img src=https://i.pinimg.com/736x/7b/e8/8a/7be88acbef00892a7de81d422101a8ed.jpg width="500"
height="600" >


{% if latest_question_list %}
    <ul>
    {% for question in latest_question_list %}
        <p><button type="button"><a href="/recommend/{{ question.id }}/">Start</a></button></p>
    {% endfor %}
    </ul>
{% else %}
    <p>No recommend are available.</p>
{% endif %}
```

```
<font size=0.8 color = #BDBDBD>
Copyright (c) 2019 Cheon Seounghyun, Lee yookyum, Park jihye, Choi yesom

    Permission is hereby granted, free of charge, to any person obtaining a copy of this software
and associated documentation files (the "Software"),
    to deal in the Software without restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, sublicense,
    and/or sell copies of the Software, and to permit persons to whom the Software is furnished to
do so, subject to the following conditions:

    The above copyright notice and this permission notice shall be included in all copies or
substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING
BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
    WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
</font>
</body>
    </center>
</html>
```

code/recommend/templates/recommend/results.html

```
<center>
<h1>{{ question.question_text }}</h1>
{% load staticfiles %}

<img src="{% static 'sample2.jpg' %}">
<ul>
{% for choice in question.choice_set.all %}
    <li>{{ choice.choice_text }} -- {{ choice.votes }} vote{{ choice.votes|pluralize }}</li>
{% endfor %}
</ul>

<a href="{% url 'recommend:detail' question.id %}">Vote again?</a>
</center>
```

code/recommend/admin.py

```
from django.contrib import admin

# Register your models here.
from recommend.models import Question, Choice


admin.site.register(Question)
admin.site.register(Choice)
```

code/recommend/apps.py

```
from django.apps import AppConfig
```

```
class RecommendConfig(AppConfig):
    name = 'recommend'
```

code/recommend/models.py

```
from django.db import models

# Create your models here.
class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

    def __str__(self):
        return self.question_text


class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)

    def __str__(self):
        return self.choice_text
```

code/recommend/urls.py

```
from django.urls import path
from recommend import views


app_name = 'recommend'
urlpatterns = [
    path('', views.index, name='index'),
    path('<int:question_id>/', views.detail, name='detail'),
    path('<int:question_id>/results/', views.results, name='results'),
    path('<int:question_id>/vote/', views.vote, name='vote'),
```

code/recommend/views.py

```
from django.shortcuts import render
from django.shortcuts import get_object_or_404, render
from django.http import HttpResponseRedirect
from django.urls import reverse

from recommend.models import Choice, Question

# Create your views here.
def index(request):
    latest_question_list = Question.objects.all().order_by('-pub_date')[:5]
    context = {'latest_question_list': latest_question_list}
    return render(request, 'recommend/index.html', context)

def detail(request, question_id):
    question = get_object_or_404(Question, pk=question_id)
    return render(request, 'recommend/detail.html', {'question': question})

def vote(request, question_id):
```

```
    question = get_object_or_404(Question, pk=question_id)
    try:
        selected_choice = question.choice_set.get(pk=request.POST['choice'])
    except (KeyError, Choice.DoesNotExist):

        return render(request, 'recommend/detail.html', {
            'question': question,
            'error_message': "You didn't select a choice.",
        })
    else:
        selected_choice.votes += 1
        selected_choice.save()

        return HttpResponseRedirect(reverse('recommend:results', args=(question.id,)))

def results(request, question_id):
    question = get_object_or_404(Question, pk=question_id)
    return render(request, 'recommend/results.html', {'question': question})
```

- 3.3.3 실행 화면

(1)image_crawler

```
In [1]: !pip install google_images_download
```

```
Collecting google_images_download
  Downloading https://files.pythonhosted.org/packages/7c/5a/ecc4cfb3efe5c7e83252f4dd8c6dbbba411a2d1485b3840859e2781a0b6d/google_images_download-2.5.1.tar.gz
Collecting selenium (from google_images_download)
  Downloading https://files.pythonhosted.org/packages/80/d6/4294f0b4bce4de0abf13e17190289f9d0613b0a44e5dd6a7f5ca98459853/selenium-3.141.0-py2.py3-none-any.whl (904kB)
Requirement already satisfied: urllib3 in c:₩users₩user₩anaconda3₩lib₩site-packages (from selenium->google_images_download) (1.24.1)
Building wheels for collected packages: google-images-download
  Running setup.py bdist_wheel for google-images-download: started
  Running setup.py bdist_wheel for google-images-download: finished with status 'done'
  Stored in directory: C:₩Users₩User₩AppData₩Local₩pip₩Cache₩wheels₩db₩91₩54₩94ea83c9384724ef2d2f3f735b4bfcd61eda897f2c02d70838
Successfully built google-images-download
Installing collected packages: selenium, google-images-download
Successfully installed google-images-download-2.5.1 selenium-3.141.0
```

```
In [1]: from google_images_download import google_images_download   #importing the library

        response = google_images_download.googleimagesdownload()   #class instantiation

        arguments = {"keywords":"셔츠","limit":5,"output_directory":'image',"print_urls":True}   #creating list of arguments
        paths = response.download(arguments)   #passing the arguments to the function
        print(paths)   #printing absolute paths of the downloaded images
```

```
Item no.: 1 --> Item name = 셔츠
Evaluating...
Starting Download...
Image URL: http://theresheis.kr/web/product/big/201803/477_shop1_943702.jpg
Completed Image ====> 1. 477_shop1_943702.jpg
Image URL: https://assets.burberry.com/is/image/BurberryLtd/c547d73af6347e61091e2fef5c1ad38a34ee1da8.jpg?$BBY_V2_ML_3X4$&wid=570&hei=760
Completed Image ====> 2. c547d73af6347e61091e2fef5c1ad38a34ee1da8.jpg
Image URL: http://gopeople.co.kr/web/product/big/201707/1379_shop1_949441.jpg
Completed Image ====> 3. 1379_shop1_949441.jpg
Image URL: https://shop.r10s.jp/advanceclothing/cabinet/2017aw09/2173-as3-9_10.jpg
Completed Image ====> 4. 2173-as3-9_10.jpg
Image URL: https://ae01.alicdn.com/kf/HTB1LGauPXXXXXbCapXXq6xXFXXXL/-.jpg_640x640.jpg
Completed Image ====> 5. -.jpg

Errors: 0

{'셔츠': ['C:₩₩Users₩₩User₩₩Desktop₩₩capston₩₩image₩₩셔츠₩₩1. 477_shop1_943702.jpg', 'C:₩₩Users₩₩User₩₩Desktop₩₩capston₩₩image₩₩셔츠₩₩2. c547d73af6347e61091e2fef5c1ad38a34ee1da8.jpg', 'C:₩₩Users₩₩User₩₩Desktop₩₩capston₩₩image₩₩셔츠₩₩3. 1379_shop1_949441.jpg', 'C:₩₩Users₩₩User₩₩Desktop₩₩capston₩₩image₩₩셔츠₩₩4. 2173-as3-9_10.jpg', 'C:₩₩Users₩₩User₩₩Desktop₩₩capston₩₩image₩₩셔츠₩₩5. -.jpg']}
```

(2)imgae_train

In [7]:
```
!pip3 install https://github.com/OlafenwaMoses/ImageAI/releases/download/2.0.2/imageai-2.0.2-py3-none-any.whl
```

```
Collecting imageai==2.0.2 from https://github.com/OlafenwaMoses/ImageAI/releases/download/2.0.2/imageai-2.0.2-py3-none-any.whl
  Downloading https://github.com/OlafenwaMoses/ImageAI/releases/download/2.0.2/imageai-2.0.2-py3-none-any.whl (151kB)
Installing collected packages: imageai
Successfully installed imageai-2.0.2
```

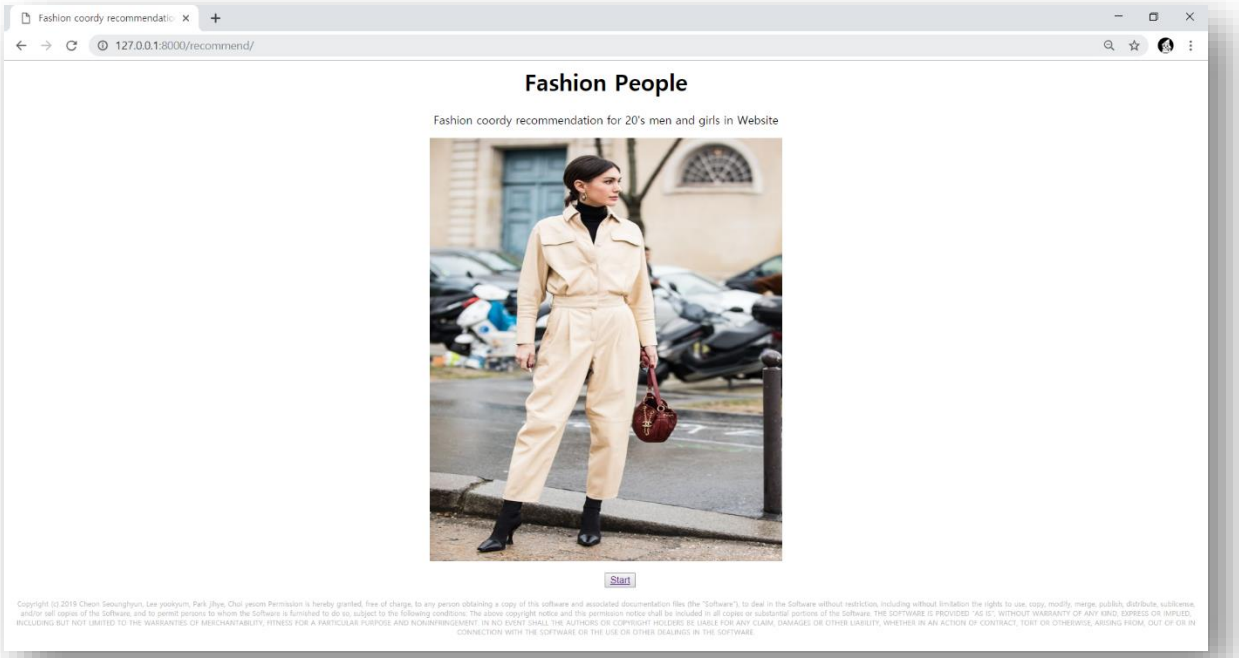
In [1]:
```
from IPython.display import Image
Image(filename='1.jpg')
```

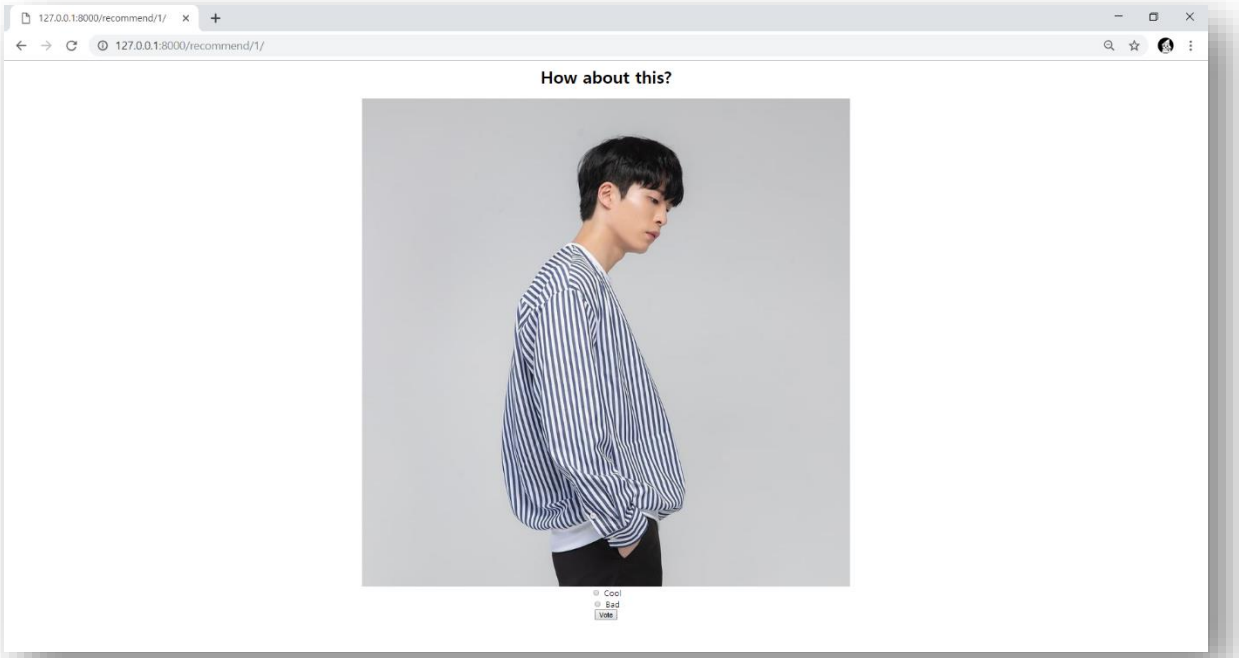
Out[1]:

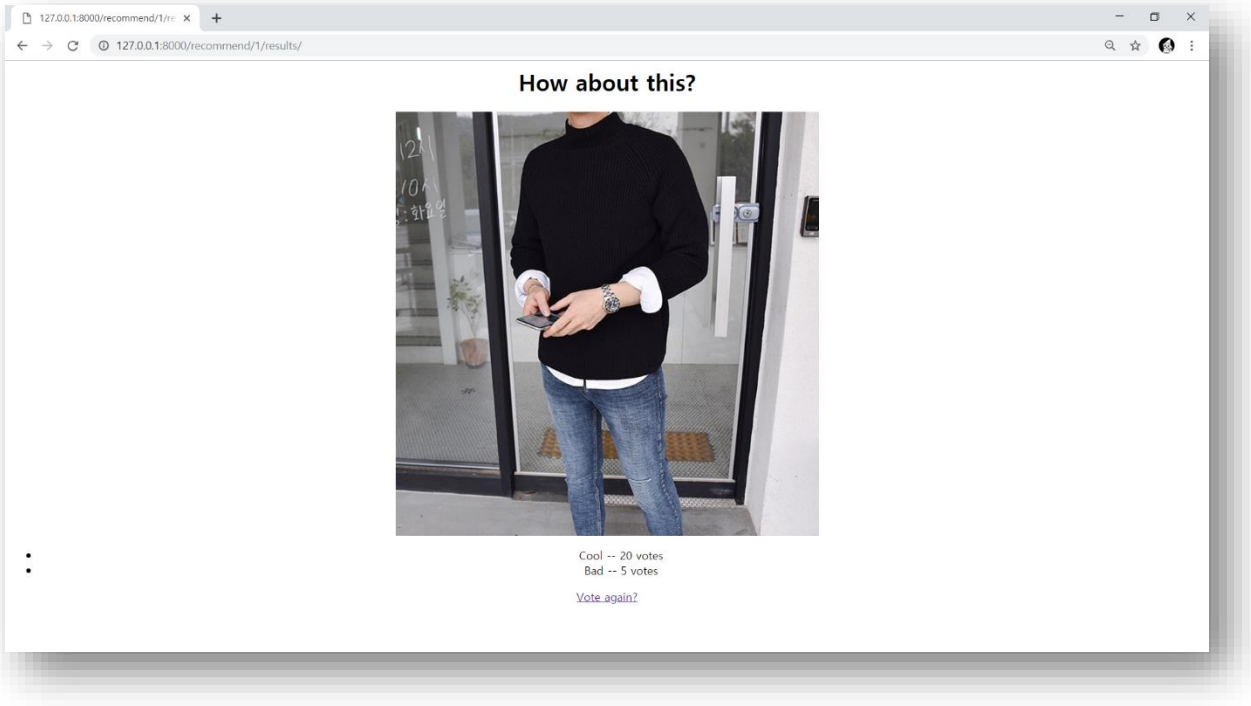
In [8]:
```
from IPython.display import Image
Image(filename='image/1.jpg')

from imageai.Prediction import ImagePrediction
import os

execution_path = os.getcwd()
prediction = ImagePrediction()
prediction.setModelTypeAsResNet()
prediction.setModelPath(os.path.join(execution_path, "resnet50_weights_tf_dim_ordering_tf_kernels.h5"))
prediction.loadModel()


predictions, probabilities = prediction.predictImage(os.path.join(execution_path, "1.jpg"), result_count=5 )


for eachPrediction, eachProbability in zip(predictions, probabilities):
    Image(filename='1.jpg')
    print(eachPrediction , " : " , eachProbability)
```

```
convertible  :  52.459532022476196
sports_car  :  37.6128613948822
pickup  :  3.175130859017372
car_wheel  :  1.8175015226006508
minivan  :  1.7487077042460442
```

(3)Django website







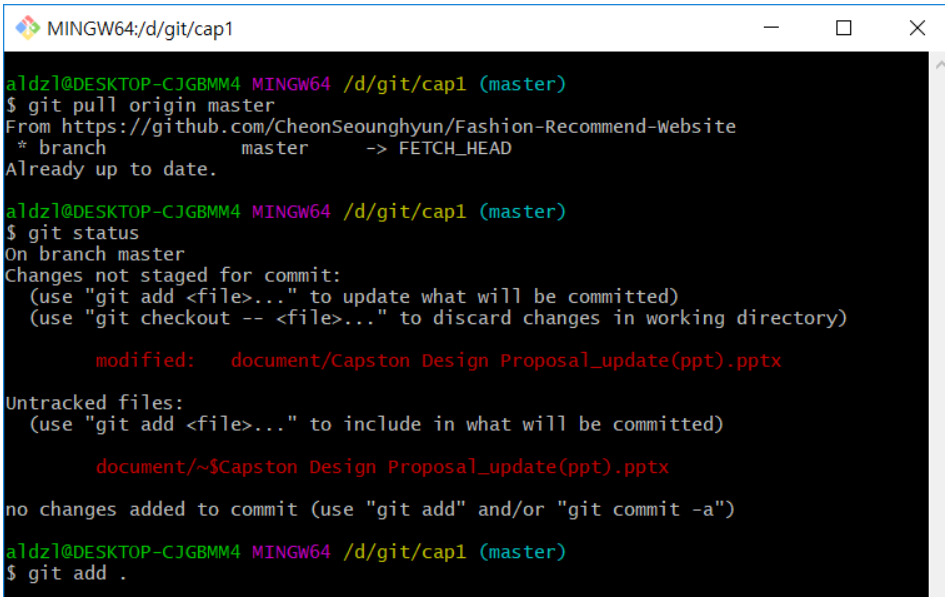
3.4 개발예정

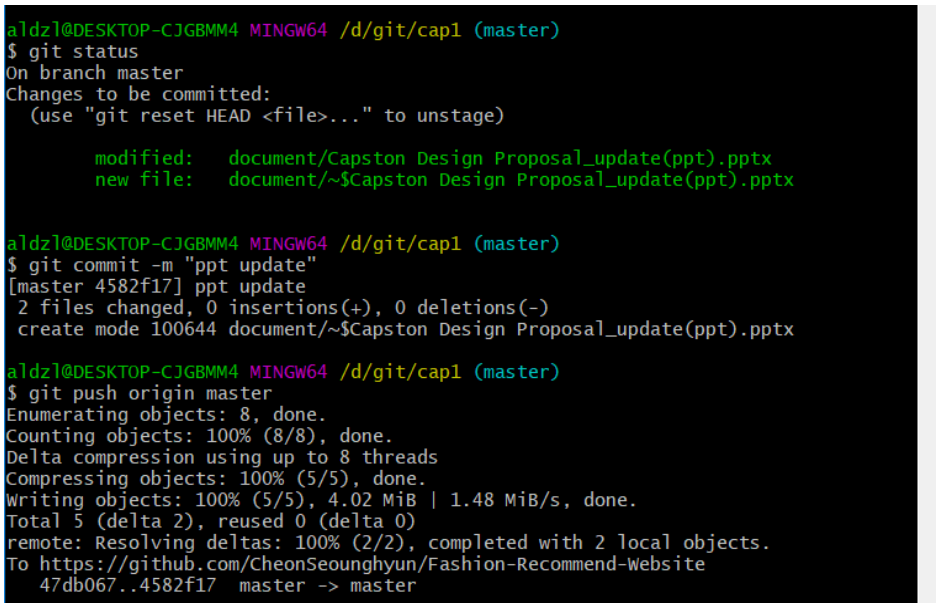
현재 웹사이트는 정적파일을 서버에 직접 올려 사용하고 있는 프로토타입이다. 내부 알고리즘이 동작하는게 아니고 고정된 화면을 반복해서 보여준다. 즉 사용자의 선택이 추천에 반영되지 않는다. 앞으로 파이썬을 이용하여 내부 알고리즘을 개발하고, 웹 사이트 패키지에 등록하여 사용자의 선택에 맞는 아이템을 추천해주는 추천 웹사이트를 궁극적으로 개발하는 것을 목표로 한다.

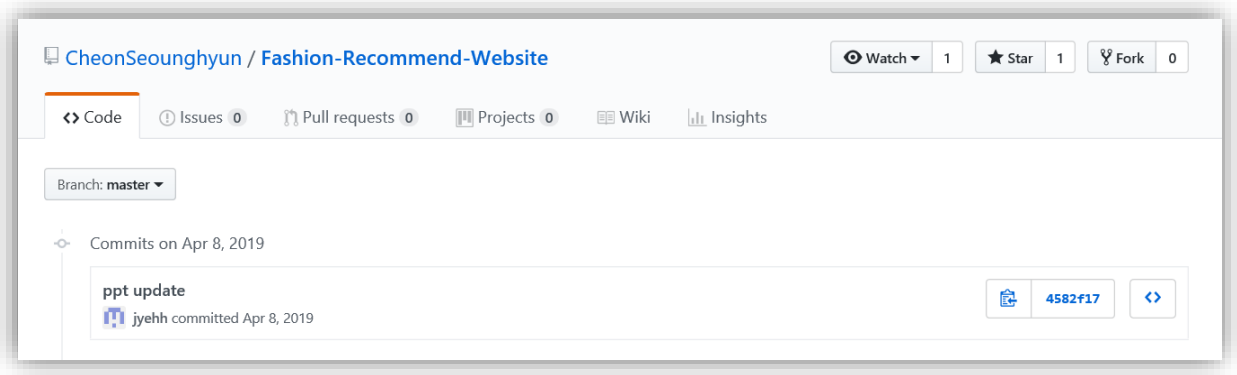
3.5 협업방식

- 깃허브

'깃'(Git)은 전문 호스팅 업체이다. 깃허브는 오픈소스 소프트웨어의 중심지(hub) 역할을 하면서 오픈소스 프로젝트가 널리 퍼지는 데 크게 기여하고 있다. 깃허브는 깃을 보다 편하게 이용할 수 있게 만든 호스팅 서비스다. 깃은 명령어를 입력하면서 이용해야 하지만 깃허브는 웹 그래픽 기반으로 깃을 이용할 수 있는 환경을 제공했다. 깃은 오픈소스 소프트웨어로 2005년에 개발된 분산형 버전관리 시스템(DVCS)이다. 포크는 내 계정으로 외부 프로젝트 코드 저장소를 그대로 복사해주는 것을 말한다. 오픈소스 프로젝트들은 수십명에서 수백명의 사람들이 사람들이 함께 소스코드를 고칠수있다. 위와 같은 이유로 깃허브를 사용하였다.







$git pull origin master는 원격 저장소에서 파일을 가져오는 명령어 로 파일 내용을 최신화 시킨다. 여기서 origin 은 원격 저장소 이름으로 링크를 간략화 시킨것이고, master는 버전을 나타낸다. $git status는 파일들의 변경 상태를 확인해준다. 변경된 파일은 빨간색으로 나타나고, 변경사항이 없을 시 초록색으로 나타난다. $git add .에서 .은 디렉토리 전체를 의미 하며, 깃허브에 변경된 파일을 등록해준다. $git commit –m "----" 는 파일 업로드 명령어로, " " 안에 들어가는 이름으로 파일이 업로드 된다. $git push origin master라는 명령어로 원격 저장소인 깃허브 사이트에 전체적인 내용이 업로드된다.

프로젝트 저장소 링크 https://github.com/CheonSeounghyun/Fashion-Recommend- Website

조원별 깃허브 id, 저장소 링크

https://github.com/LeeYooKyum LeeYooKyum ,

https://github.com/choiyesom choiyesom,

https://github.com/jyehh jyehh

3.6 팀원 소개, 역할 및 소감

- 각자 소개와 역할, 소감 작성(5줄 이상)

천승현(Cheonseounghyun)

역할: 내부 알고리즘 구현

소감: 산학 캡스톤 디자인 과목을 수강 하기전 수업계획서를 읽어보며 팀프로젝트 강의라는 것을 알게 되어 팀원과의 소통과 토의가 중요하다고 생각하였습니다. 지난 약 8주간 프로젝트를 진행하며 주차별로 목표를 정하고 정해진 목표를 달성하기 위해 팀원들과 소통하며 문제를 해결하는 과정이 매우 인상적이었습니다. 앞으로 최종 목표까지 구현하기 위해서 더 많은 시간을 써야하겠지만, 지난 4학년동안 재학하며 배웠던 여러가지 지식과 문제를 해결하는 능력을 이용하여 팀원들과 함께 완성된 결과물을 제출하고 싶습니다.

최예솜(choiyesom)

역할: 웹 사이트 구축
소감: 처음 빅데이터를 가지고 웹페이지를 만들기 시작할 때 막막함도 있었지만 조원들과 함께 모두 서로에게 도움을 줄 수 있도록 하여 순조롭게 빅데이터 요소를 웹사이트와 알고리즘을 구현하여 웹 만들기를 시도해 볼 수 있었습니다. FCMW 페이지 구현 중 가장 기억에 남는 부분은 웹 크롤링을 통해 이미지 API를 더 빠르고 효율적으로 얻게 되는 부분을 자세히 공부하게 되었습니다. 처음 파이 참을 접해 접하기 힘든 지식이 필요하기도 했었지만 책을 빌려 조원들과 공부하고, 부족한 부분도 학습하였습니다.
또한 빅데이터 수집, 저장 및 분석, 시각화를 순서대로 알아가며 결과물까지 다가가려는 노력을 하였고 그에 대응하는 결과물도 만들어 냈습니다. 마지막 최종은 더 풍부한 지식과 배움으로 좋은 결과물이 나올 때까지 노력할 것입니다.

이유겸(LeeYooKyum)

역할: 보고서 작성 및 검토

소감: 산학캡스톤디자인1 과목을 처음 수강할 때 이 과목은 지금까지 배워왔던 어느 과목과는 다른 모든 것을 교수님이 알려주시는 것이 아닌 학생들이 스스로 또는 학생들끼리 협업을 해서 프로젝트를 진행해야 한다는 점이다. 이러한 이유로 많은 부담감을 느끼게 되었지만 조원들과 같이 프로젝트를 진행해보니 혼자였다면 하지 못할 부분들을 협업을 통해 원활히 진행하게 된 것 같다. FCMW 웹 구현을 하던 중 가장 흥미로웠던 부분은 ImageAI를 통해 간단한 차량 예시를 통해 차량의 종류를 분석하는 부분인 것 같다. 지금까지의 과목들은 사실 결과물만 내려고 노력했던 것 같다. 하지만 FCMW 웹 구현 프로젝트를 진행을 해보니 중간 과정 없이는 좋은 결과물은 없다고 생각된다.

박지혜(jyehh)

역할 : 데이터베이스 구현

이번 산학 캡스톤 디자인이라는 과목을 수강하면서 진행하게 된 프로젝트에서 SQL Lite를 사용하여 Data Base를 구현하는 파트를 맡아 활동하였습니다. 지난 인턴십 활동을 하면서 데이터베이스와 자바로 웹 페이지 구현하기를 배웠었는데, 그 덕분에 이번 프로젝트 진행하고 이해하는데 있어 조금이나마 도움이 되어 다행이었습니다. 또, 이번 프로젝트를 하면서 새로운 경험 또한 많이 하게 되었는데 그 중 웹 크롤링을 통해 필요한 이미지들만 가져왔던 것과, 가져온 사진들을 분석하는 것이 신기하고 재미있었습니다. 단계별로 프로젝트가 완성이 되어가는 모습을 보며 모르는 게 많은 만큼 서로서로 도와가며 진행해야 하기 때문에 많은 대화를 통해 조금 더 가까워 질 수 있었고 그래서 다 같이 뿌듯함을 느낄 수 있는 것이 좋았다고 생각합니다.

4. 참조

4.1 참조 링크

- 최종 보고서 추가 예정

4.2 깃허브 히스토리

- 최종 보고서 추가 예정