Purpose
목적은 기본 사용성입니다.

#누구나
AI 알고리즘을 몰라도 모델을 만들 수 있다.(알고리즘 풀네임)
프로그래밍 언어를 몰라도 학습/테스트를 할 수 있다.(파이썬 동작 설명, gpu/cpu)
IT 에 대한 지식이 없어도 모델을 만들 수 있다.(기존 통계학에 비해 이점을 가진 인코더 부분 더 나아가 데이터탐색만 하고 싶은 사람들)

#용이하게
모델의 학습과정을 추적할 수 있다.
시각화를 통해 데이터 특성을 파악할 수 있다.
데이터의 특성에 맞는 모델을 찾을 수 있다.
해당 데이터의 중요한 feature 들을 찾아 가공 및 선택할 수 있다.

#전체구조
디렉토리
flow 중심
기능중심

#story
홈화면
회원가입 및 로그인
데이터 업로드
- redis
- 직접 csv 올리기
데이터 전처리
모델 선택
모델 학습 후 결과 확인
5-6 모델선택-학습 반복을 통해 최적 모델 탐색 및 피쳐 선택
데이터,모델,예측파일 다운

#전체구조
api : 각종 동작 모듈들이 존재
customer : user space
fastprogress : customize package
homes : 로그인 삭제, 디비 초기화 등
static : image 소스파일
templates : html or css 파일

#0. 서버구동
init_sh 에서 필요한 패키지를 다운
config 에서 각종 변수들 명세
continue : 프로세스 조회 후 app.py 있는 지 확인
app : 각종 url 을 register_blueprint 걸고, 실행

#1. 홈페이지
home.html : id 가 있어야 학습창 생성, 회원삭제 생성
home.__init__ : html 처리 flask 함수들
home.utils : 각종 python support

#2. 회원가입
register.html : 이름 id 를 입력받아 register 로 넘겨줌
home.__init__ : register 가 조건 확인 후 DB 등록 후 로그인 창으로 이동
Home_utils 에서 make directory 함수 사용 --> 실제로 유저스페이스 생성 및 db 에 저장
파이참:완료시 유저 스페이스 생성

#2. 로그인
Login.html : loginform 상속받아서 로그인 입력 받고 flask 로 요청
아이디 입력 후 login
아이디 없으면 no id 출력

#3. 데이터 업로드 - 1
Api.html : file upload -> popupopen event -> load glob.html
glob.html : 업로드 및 삭제 기능을 요청하면
app.py 에서 datafile_upload and datafile delete 를 통해 처리한다.
업로드 csv 데이터들은 모두 user-space 에 존재한다.

#3. 데이터 업로드 - 2
redis-cli 에서 redis 데이터 업로드
app 에서 로컬 디렉토리 검색 후 csv_data_list 를 넘겨준다.
app 에서 데이터베이스 검색 후 view_list 를 넘겨준다.
csv/view 호출 : api 에서 csv/view 리스트로 html 을 자동 생성한다.
redis 호출 : api 에서 redis_load() 함수를 호출하여 redis 에서 데이터 조회 후 html 을 추가한다.

<기본적으로 redis 에서 필요한 파일들을 불러서 그것으로 가공해라라는 철학>
#4. 데이터 전처리 - 1 : csv
Successive_excel.html : load_partial_first() 함수로 excel_post.py 요청
excel_post.py : csv file to html 코드로 변환하고 html 에 로드
Successive_excel.html : Load_left, load_right 로 좌우 움직이면서 자동저장
Successive_excel.html : delete mode - add_del_btn 함수로 delete btn 을 추가한 후 추가 삭제
Successive_excel.html : save_partial_excel 로 저장 이벤트를 만들어
excel_post.py : save_partial_excel 로 로컬에 저장
Successive_excel.html : col_check_save 로 excel_post.py 의 make_view 로 데이터베이스에 저장 -> 뷰 생성

#4. 데이터 전처리 - 2 : view
view_post.py 에서 view_sheet 함수 호출
View.html 을 열어 load_partial_first 를 실행해서 샘플 데이터 초기화를 한다.
view.html 에서 버튼을 체크해서 get_summary_of_colum 함수로 그래프를 불러온다.
statistics.view_summary.py 에서 calculate_stat 함수로 이미지와 요약 그래프를 생성 후
view.html 에 뿌려준다.

#4. 데이터 전처리 - 3 : redis
redis-cli 가서 format 보여주기 (hvals key) -- tablename,row_id,{c1:v1,c2:v2,c3:v3}
Successive_excel.html : load_partial_first() 함수로 excel_post.py 요청
excel_post.py : redis data to html 코드로 변환하고 html 에 로드
Successive_excel.html : Load_left, load_right 로 좌우 움직이면서 자동저장
Successive_excel.html : save_partial_excel 로 저장 이벤트를 만들어
excel_post.py : save_partial_excel 로 로컬에 저장
col_check 는 불가능, redis 데이터는 보존성을 위해 수정 x

#4. 데이터 전처리 - 4 : new
new_excel html 을 불러와서 값을 입력 후
excel_post.py 의 save_new_excel 함수를 사용해서 redis 나 csv 로 저장한다.

#5. 모델 선택
api.html 에서 AiSelect 이벤트로 html 코드 추가 후 엔진들에 필요한 값을 입력 받고
dt_engine/resnet_engin 으로 api 에 html 코드를 추가한다.
이 때 동시에 추가한 엔진을 live_engine_post.py 에 요청 후
enginelive_sql.py 의 insert 를 이용해 데이터 베이스에 넣는다.

추후 api 의 load_engine_histor 를 통해 live_engine_post.py 에서 조회 후 이전 사용했던 엔진들을 불러온다.

6. 모델 학습 및 결과 체크 - 1
--학습
api.html :api 에서 drag & drop 이벤트로 Data 이름, model 타입이름, 하이퍼파라미터, 열 리스트, 데이터 타입을 보내며 ai_post.py 요청
ai_post.py : 모델 별 train/predict/retrain 으로 나눠서 타입에 맞게 분할 후
ai package : decision_tree/torch_resnet 파일을 sub 터미널로 동작시킨다.
progress.txt : 진행할 때마다 현재 상황을 html 파일로 쓰고
app.py : app.py 의 generate_progress url 을 통해 파일 쓴 상태시간을 추적하며
progresbar.html : ai 모델 같은 경우 학습이 진행되는 도중에, progressbar.html 로 진행상황을 보여준다.

학습이 완료되면 result 를 자동으로 results db 에 저장하고
modellive_sql.py 를 통해 modelfile 관련 기록을 저장한다.
완료된 모델 파일은 user space 에 저장한다. 또한 모델 관련 이미지들은 chart 디렉토리에 자동 저장된다.
Html 에 또한 모델을 추가한다.

--예측
api 에서 드래그 & drop 이벤트로
Data 이름, model 타입이름, 모델파일네임, 열 리스트, 데이터 타입을 보낸다.
Ai_post.py 에서 타입에 맞게 파이썬 파일을 호출한다.(predict)
Ai_post.py 에서 예측이 완료되면 result 를 db 에 업데이트하고, live_result_sql.py 를 통해 history 를 저장한다.
중요) 결과는 redis 에 저장하고 api html 에 또한 결과 이미지와 모델 이름을 추가한다.
6. 모델 학습 및 결과 체크 - 2

학습이나 예측을 진행할 때 마다 과거 기록을 추적하도록
api.html 에서 학습/예측 진행 시
log_post.py 에 있는 write_histoy 를 호출해서
백업으로 DB 에 log_history_sql.py 를 이용해서 쓴다.

7. 최적 모델 찾기 5~6 반복 - 1
trace_html : 모델 별 차트 타입과 이름을 이용해 init_chart_kinds() 초기화 한다.
보고 싶은 종류를 골라 trace_html 클릭 한다.
Live_model_chart_post.py 의 load_chart 를 이용해 image 의 소스를 채우고
user space 에서 저장된 차트를 trace.html 에 뿌린다.

7. 최적 모델 찾기 5~6 반복 - 2
same as # 7. 최적 모델 찾기 5~6 반복 - 1

8. output 다운 - 1
결과를 다운로드 받으려면 live_result_post.py 의 live_result_from_redis 를 통해 redis 를 변환해서 내려받는다.

application-1
tabular 데이터 : 성적예측, 문제풀이 시간 예측
자연어 데이터 : 문제 생성 및 추천, 지문 데이터 자동 분석
음성&이미지 : 문제 풀 때 셀럽 목소리로 읽기, 강의 영상 퀄리티 분석 -> 자기 지표 학원 선생님들의 역량을 좀 더 객관화 할 수 있을 것

application-2

1. 온라인적 측면 : 오프라인의 효율적 학습 효과 고취 외에도, 초개인화된 학습을 달성할 수 있다. 디쉐어 직원이 누구나 쉽게 학생들의 수준과 위치를 빠르고 정확하게 진단해낼 수 있다면 웹 서비스나 커리큘럼 기획을 맞춤형으로 할 수 있다. 즉각적인 피드벡으로 학부모님들이 학생들의 현 수준을 지속적으로 정확히 파악해 가정 지도를 고취시킬 수 있다. 이는 인강의 비중이 높은 본 회사의 차별적 경쟁력을 강화 시킬 수 있다.

2. 오프라인적 측면 : 학생들의 기존 문제풀이 관련 데이터, 기존 성적데이터 등을 이용해서 누구나 쉽게 오프라인 분원에서 정확한 문제 풀이 시간 및 성적 점수를 예측하게 할 수 있다.
이는 상담을 할 때 학습 레벨을 정확하게 예측해 학생들을 본원으로 끌어들일 수 있고, 기존의 경우 학생들의 자기진단 지표 및 분원 선생님 학습 보조 도구로써 유용하게 사용 가능할 것이다.

3. 마지막으로 분원 선생님들이 또한 학생 정보를 통해 학습/테스트 문제들을 추천할 수 있으며, 기존 문제들이 부족하다면 R&D 에서 개발한 핵심 문제나 스타일을 기반으로 학생들이 취약한 지문을 넣어 양질의 문제를 만들어 학습을 고취시킬 수 있을 것이다. 이는 급변하는 교육과정에 대해 효과적으로 대처할 수 있다. 기존 문제들 중에 학습 코드에 맞는 문제 은행을 filtering 가능