Server (Django Rest-API)

System Requirements

- Python 3.6
- virtualenv python module (https://virtualenv.pypa.io/en/latest/installation/)
- Linux or Unix based development system (Linux Recommended, MacOS support is limited)
- Docker 18.00.0+
- Docker Compose

Important Note!

In /recommender/services/train.py AutoML frameworks TPOT and AUTO-SKLEARN might not run properly in non-linux environments. So they can be commented out in the source code for running locally in development outside of a docker container. AUTO-SKLEARN is commented out by default as it will not install on my development environment of MacOS. You might also need to comment out TPOT and not use these models locally. The production docker image allows you to work around these limitations.

Create a Python Virtual Environment

From the project root directory

```
# virtualenv venv
```

Activate Python Virtual Environment

```
# source venv/bin/activate
```

Install dependencies

```
# pip install -r rest-api/requirements.txt
```

Make Migrations

From the rest-api directory

```
# cd rest-api
# python manage.py makemigrations recommender
# python manage.py migrate
```

Create Rest-API Superuser

From the rest-api directory

```
# cd rest-api
# python manage.py createsuperuser
# user: admin, pass: admin123$
```

Run the Rest-API Server

```
# python manage.py runserver
```

- From a web browser navigate to http://localhost:8000/admin/ and verify the Rest-API Server is running and that you can log in.
- From a web browser navigate to http://localhost:8000/api/v1/rest-auth/login/ and verify that the Rest-API Server login page endpoint shows.

If you get an error about the hypervolume module, reinstall deep with the following command

```
# pip install deap==1.0.2.post2
```

Client (VueJS Web Application)

Requirements

- NodeJS
- NPM

Install dependencies with npm

From the client-app directory install dependencies

```
# cd client-app
# npm install
```

Run Client Application

```
# npm run serve
```

From a web browser navigate to http://localhost:8080 and login with superuser credentials. Alternatively navigate to http://localhost:8000/admin/ and setup new credentials.

Docker Builds For Server

Requirements

- Linux (Debian Recommended)
- Docker v18.00.0+
- Docker Compose v3+
- Python v3.6

Build the base docker image

From the root directory:

```
--Build the image
# docker build -t grayrobert/price-recommender-base -f
docker/base/Dockerfile .

--Push to docker hub
# docker push grayrobert/price-recommender-base
```

After running docker images you should have a new base ubuntu image with python installed

Build the production docker image from the base docker image (REST-API Server)

Note: during docker build the training service python file train.production.py replaces the default train.py and enables TPOT and AUTO-SKLEARN models mentioned earlier.

From the root directory:

```
# docker build -t grayrobert/price-recommender-server -f
docker/server/production/Dockerfile .

--You can run the image and jump to a bash prompt using the following
command
# docker run -it --entrypoint /bin/bash grayrobert/price-recommender-
server

--Push to docker hub (please don't do this without permission)
# docker push grayrobert/price-recommender-server
```

Save the server docker image for manual deployment to production

From the root directory:

```
# docker save -o target/price-recommender-server.tar grayrobert/price-
recommender-server:latest
```

Docker Builds For Client

Requirements

- Linux/Unix
- Docker
- Node/NPM

Build the client docker image

From the root directory:

```
--Build the image
# docker build -t grayrobert/price-recommender-client -f
docker/client/Dockerfile .

--Push to docker hub (please don't do this without permission)
# docker push grayrobert/price-recommender-client
```

Run the client docker image

```
# docker run -it -p 8080:8080 --rm grayrobert/price-recommender-client
```

Save the client docker image for manual deployment to production

From the root directory:

```
# docker save -o target/price-recommender-client.tar grayrobert/price-
recommender-client:latest
```

To Run the application in production

Docker compose will orchestrate the build of images and starting of both the server and client, mapping ports and whatever other requirements are needed. Curently the most streamlined way to put the application into production is to install docker, docker compose and git. Clone the source code fom git repository and then use docker-compose to do all the heavy lifting.

On a server with Debian Linux, GIT, Docker and Docker Compose installed and the source code cloned from git repository simply do the following two step deployment. From the root directory:

```
1. Pull the latest changes from master branch
# git pull origin master
2. Run the following command
# docker-compose --file docker/server/production/docker-compose.yml up --
build
```

Then open a browser and navigate to http://ip of server or localhost:80