ECE 157A - Final Project

ECE 157 TAs

Fall 2020

Phase 1 Due: Wednesday, 25th of November
Phase 2 Due: Friday, 11th of December
Submission: schedule live demo with TAs
Group: allow working as a group of 2-3 people

1 Introduction

This is the final project where we will build a simple ML analytic Web API using Django [2] and the Django REST framework [3]. This is the opportunity to integrate the ML techniques you have established in the first three homework towards an end-to-end solution that can be easily re-used by others. At the end of this quarter, we will have the most basic skeleton of a full-stack web, based on which we can continue to add more components in ECE 157B next quarter.

2 Motivation

You may wonder why we teach web development in a ML course. Consider the following scenario:

> Recall you work as the only data expert in a medical company. You have done excellently whenever your dear boss assigned you a new data analytic task. However, as machine learning becomes so popular, all colleagues in your company even those from other departments come to you with their own data and ask for a ML point of view. You realize that you don't have time to solve their individual problems, and many problems actually fall into three main categories - classification, regression and outlier detection. One day an idea came across your mind: why not build an AI agent to present my knowledge and analyze data for these people? You remembered there was a talk about the Intelligent Engineering Assistant (IEA) when you visited UCSB. You knew the AI agent has to be a software system which can be easily deployed to various user cases. Aha, you said to yourself, a web app sounds like a good design choice!

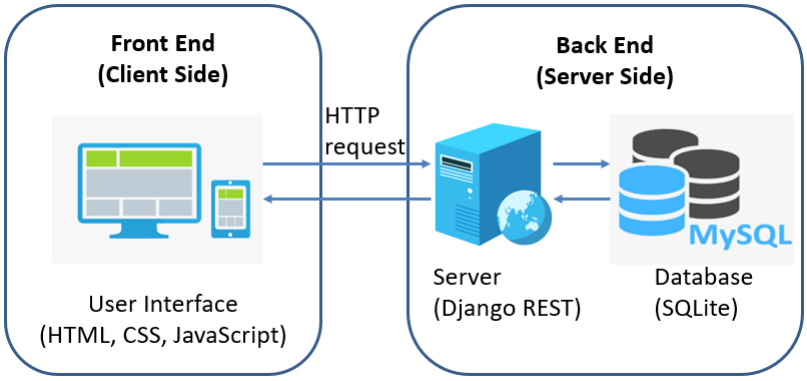

Figure 1: Full stack web development

3 Full Stack Web

A full stack web development project includes developing both the client and server software. Fig 1 shows three major components: the back-end (server), the front-end (client), and the database. The client and server communicate with each other via the standard Http request and response, a protocol for distributed information systems [5]. For example, the GET method retrieve data from the server resources purely based on the request specified on the client side.

There are many paradigms to implement the system. In this project, we will use the Django REST framework [3] to implement our back-end. It is an add-on to the Django framework [2] with more powerful features. The Django framework provides us a lightweight Web server written purely in Python. Note that this server is used only in development and not for real production. Django also configures a default database engine called SQLite [7]. Again, this is for development only. Django REST framework has its native Request class which extends the standard Http request. The Django back-end can be connected to various front-end frameworks such as React [6], Angular [1], etc. But for this project, our front-end can be a simple HTML form [4] rendered also by Django REST framework.

In this project, you will host the server on your local machine. You also need to mock the client side in your local browser. In reality, however, the client and server can be separated and multiple clients can be distributed all around the world.

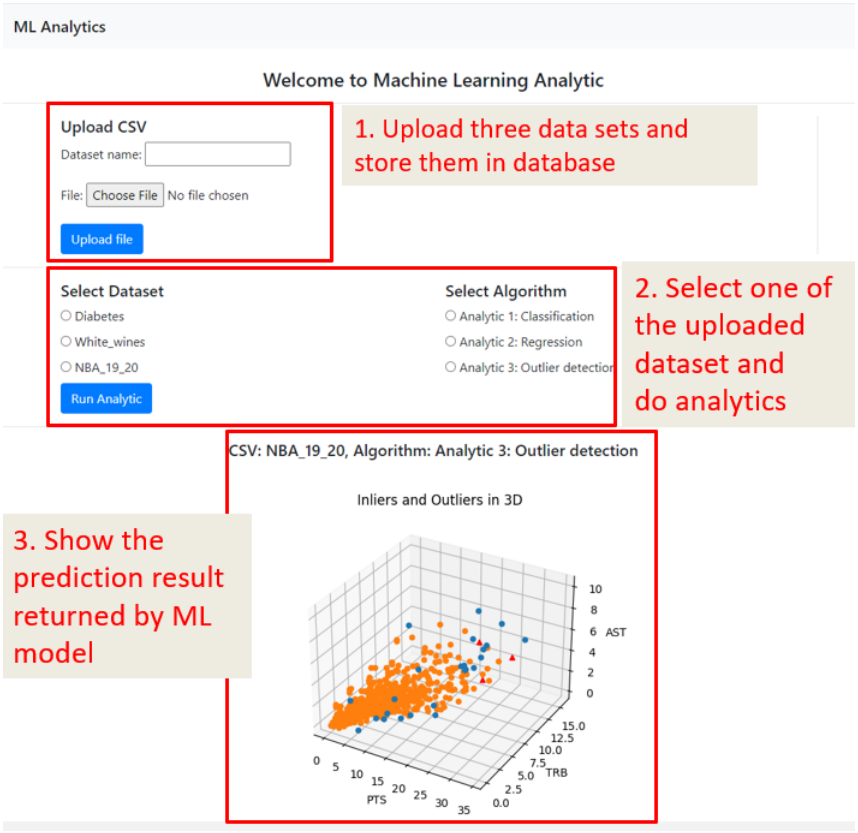

Figure 2: Basic requirements for ECE 157A and ECE 272A

4 Basic functionality requirements

In this project, we require implementing three basic functionalities:

- Upload three data sets (*diabetes.csv, white_wine.csv, NBA_player_stats.csv*) and store them in database.
- Select one of the uploaded data set and perform inference with pretrained ML model correspondingly.
- Show the prediction result returned by ML model (*classification, regression, outlier prediction*)

You should host a web server on your local machine and render a HTML web page as shown in Fig 2. Note that the HTML form template is provided by the TAs. So your job is mainly on implementing the server side with Django framework.

5 Extra requirements for ECE 272A (optional for ECE 157A)

We require grad students enrolled in ECE 272A to implement the following extra functionalities (see Fig 3 and Fig 4). Students in ECE 157A can get extra credits by implementing them:

- Make the app more robust by implementing more hands-on features:
  1. Validate file extension (.csv) before uploading.
  2. Select the latest file for analytic based on upload time, when files have duplicated names.
  3. Delete files based on file name.
- Administrator feature (see Fig 4)
  1. Log in as admin and manage the pretrained model and ML scripts
- Retrieve analytic history
  1. Implement a table listing the existing analytic results.
  2. Make the 3D outlier plot more interactive

6 Recommended Flow

- Read the above sections in detail and understand the overall spec.
- Go through the commands in Django REST tutorial. `https://www.django-rest-framework.org/tutorial/1-serialization/`
- Read the API guide and understand how to use Django REST's API. For example, what is a serializer? `https://www.django-rest-framework.org/api-guide/serializers/`
- Follow the *Final project phase 1 step-by-step guide* provided by TAs to implement phase 1.
- Take enough time to review the key design concepts and think about how you will implement the other requirements.

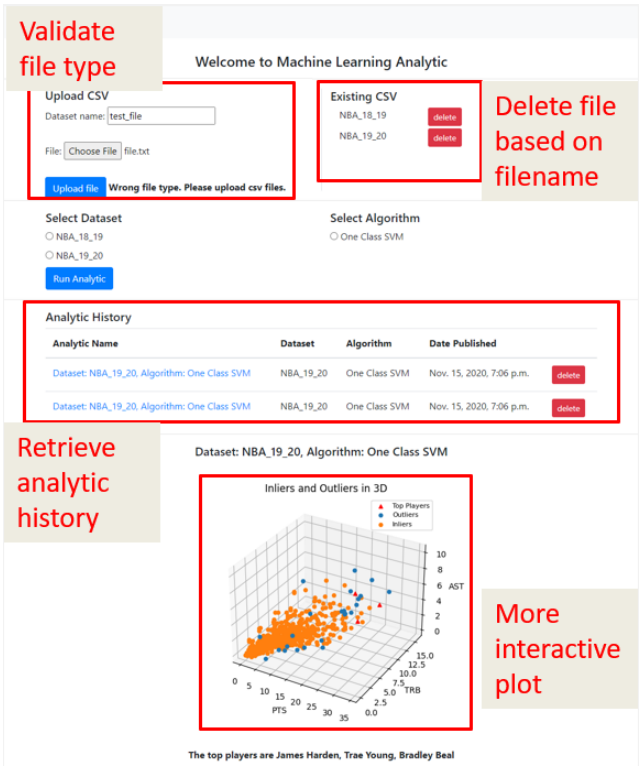

Figure 3: Extra requirements for ECE 272A: more robust features

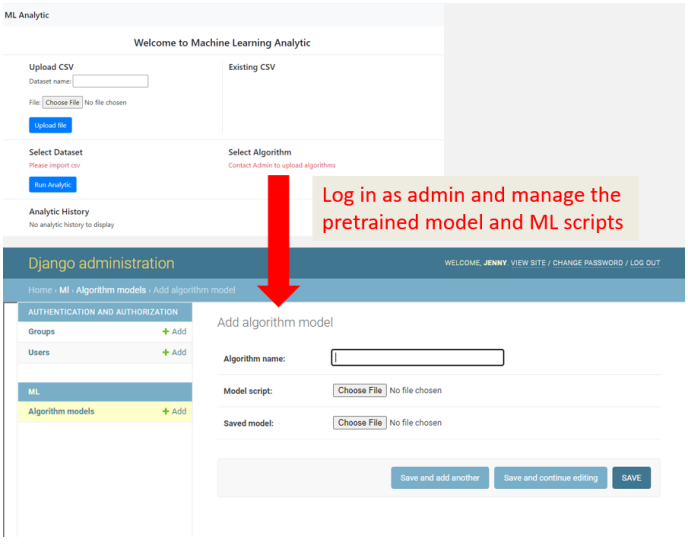

Figure 4: Extra requirements for ECE 272A: admin feature

7 Phase 1 Grading

You should at least implement the first basic requirement (CSV file uploading) by the end of phase 1. But we encourage you to implement more functionalities ahead of time.
In order to get full credits on phase 1, you need to demonstrate the following parts:

- Setup and run Django server on local host. (5 points)
- Configure the database model for file uploading. (5 points)
- Establish the client-server communication with GET/POST methods. (5 points)
- Successfully store all three csv files to the server database after client request. (5 points)
- Render a user-friendly front-end interface as shown in the first red box in Fig 2 (5 points)

8 Phase 1 Demo

TAs will schedule a 10-minute demo time slot with each group on Wednesday, Nov 25th. You will start the server and show your code and web page via screen sharing. Everyone in the group should be prepared to answer some questions regarding your implementation.

References

[1] Angular. `https://angular.io/`.

[2] Django framework. `https://www.djangoproject.com/`.

[3] Django rest framework. `https://www.django-rest-framework.org/`.

[4] Htmlforms. `https://www.w3schools.com/html/html_forms.asp`.

[5] Http. `https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol`.

[6] React js. `https://reactjs.org/`.

[7] Sqlite. `https://www.sqlite.org/index.html`.