Name Xue, Bing Date: Dec 17th, 2022

(last name, first name)

NYU ID: bx2109

Course Section Number: CSCI-GA.2433-001

**Project Part 4**

Total in points (100 points total): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor’s Comments:

Affirmation of my Independent Effort: Bing Xue

(Sign here)

Report on Project Part 4

Team Member:

Bing Xue (bx2109)

Introduction:

We implemented the management system in Django 2.1. We use MySQL as the database. We choose Django and MySQL because they are very popular and free, also they are quite compatible. A Django application is MVC-structured. The main implementation is in view.py, models.py, and temples/. You can deploy it and interact with it via the webpage.

Repo Address: https://github.com/XueBingo/Bookmanagement

Workflow & Data-driven machine learning training:

Diagram

Description automatically generated

1. Collect the new data when we need to order new books. This involves querying the.
2. Preprocess the new data in the same way as the original training data.
3. Combine the new data with the original training data, by concatenating the two datasets or by randomly sampling from the combined dataset to create a new training set.
4. Train the model on the combined dataset with the same hyperparameters and training method that was used for the original training.
5. Evaluate the performance of the retrained model on a held-out test set.

Django & ORM

Django is a popular Python web framework that provides a convenient way to build web applications quickly. It includes an object-relational mapper (ORM) that allows developers to interact with a database using Python code instead of SQL.

The ORM is an important part of Django because it abstracts the underlying database structure and provides a way for developers to create, read, update, and delete database records using Python code. This can make it much easier and more efficient to work with databases in Django, especially for complex queries or operations.

One of the key benefits of using the Django ORM is that it provides a convenient way to interact with the database that stores your book information. With the ORM, I am able to easily create, read, update, and delete records in the database using Python code, rather than having to write raw SQL queries.

This can be especially useful when working with complex queries or operations, as the ORM allows me to focus on the logic of your application rather than the details of the underlying database structure.

My end-to-end solution:

An end-to-end solution refers to a system that covers all aspects of a process, from start to finish. In the context of a bookstore management system, this means that the system should be able to handle all of the tasks and processes involved in managing your bookstore's stock and book information.

To build an end-to-end solution with Django, I take all of the different components and features that are required to manage your bookstore effectively into consideration. This includes things:

* A user interface for bookstore staff to interact with the system
* A database to store book information and stock data
* A way to add, edit, and delete book records in the database

By building a comprehensive solution that covers all of these aspects, I create a system that can handle all of the tasks and processes involved in managing your bookstore. This will make it easier for bookstore staff to work efficiently and ensure that you have the information you need to make informed decisions about your business.

Reference architecture (RA) documentation:

* Overview

The bookstore management system is a web-based application that allows bookstore staff to manage the stock and book information for their bookstore. The system includes a user interface for interacting with the database, as well as a database to store book and stock data.

* Components

The bookstore management system consists of the following components:

User interface: The user interface is a web-based application that allows bookstore staff to interact with the system. It includes features such as the ability to add, edit, and delete book records, as well as view and update stock levels and handle orders.

Database: The database stores all of the book and stock data for the bookstore. It includes tables for storing information about books, authors, publishers, and stock levels.

Backend: The backend of the bookstore management system is responsible for handling requests from the user interface and interacting with the database. It includes functions for creating, reading, updating, and deleting records in the database, as well as generating reports and analytics.

* Architecture

The bookstore management system is built using the Django web framework, with a MySQL database. The user interface is built using HTML and communicates with the backend using the Django REST framework.

The backend handles requests from the user interface and interacts with the database using the Django ORM. It also includes functions for generating reports and analytics, which are used to provide insights into the performance of the bookstore.

* Deployment

The bookstore management system is deployed on a web server, with the user interface accessible through a web browser. The database is hosted on a separate server, and the backend communicates with it using a network connection.

* Conclusion

The bookstore management system is a comprehensive end-to-end solution for managing the stock and book information of a bookstore. It includes a user interface, a database, and a backend, and is built using the Django web framework and a MySQL database. The system can be deployed on a web server.