

Capstone Project Report: Recommender System for Extracurricular Activities

1. Introduction

1.1. Project Overview

This project develops a recommender system to suggest extracurricular activities to students based on their interests, skills, and preferences. Using advanced data engineering and machine learning techniques, the system aims to provide personalized recommendations to enhance students' extracurricular engagement.

1.2. Objectives

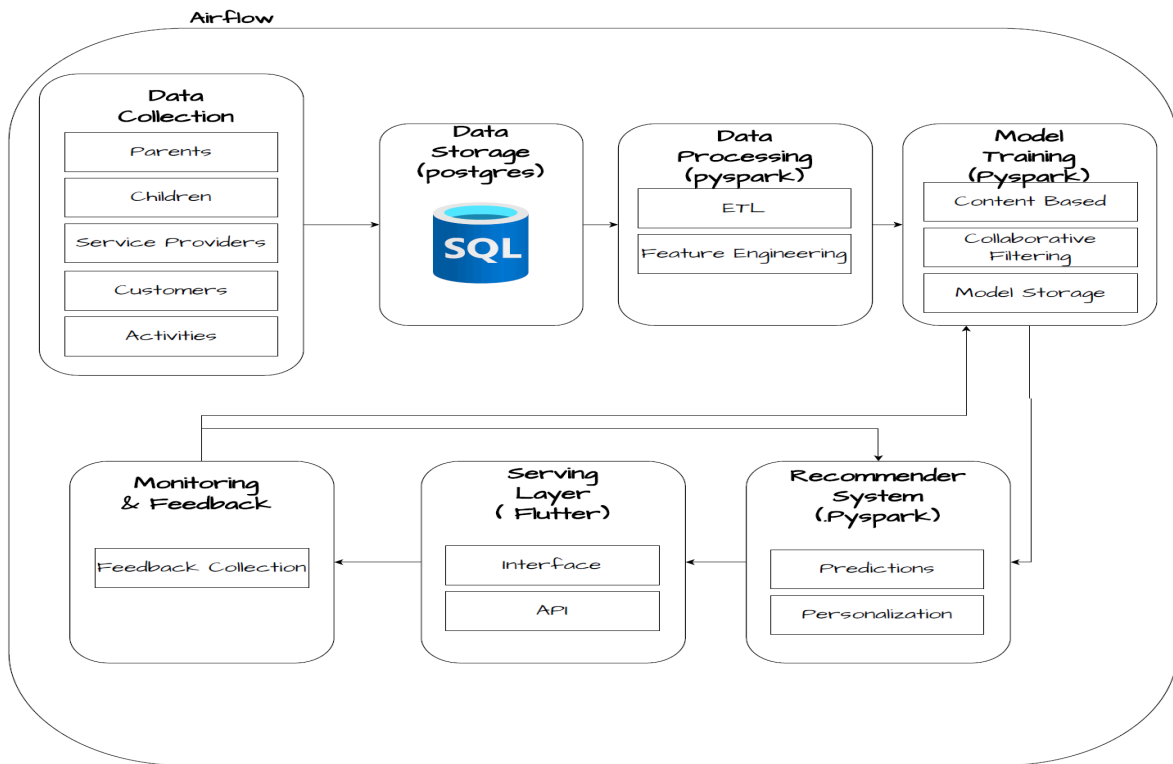
- **Data Collection:** Aggregate data from various sources, including students and activities.
- **Data Storage:** Store the data in a PostgreSQL database.
- **Data Processing:** Perform ETL and feature engineering with PySpark and Airflow.
- **Model Training:** Develop content-based and collaborative filtering models.
- **Serving Layer:** Deploy the system to deliver recommendations.
- **Monitoring & Feedback:** Collect and utilize user feedback for continuous improvement.

2. Architecture

2.1. System Architecture

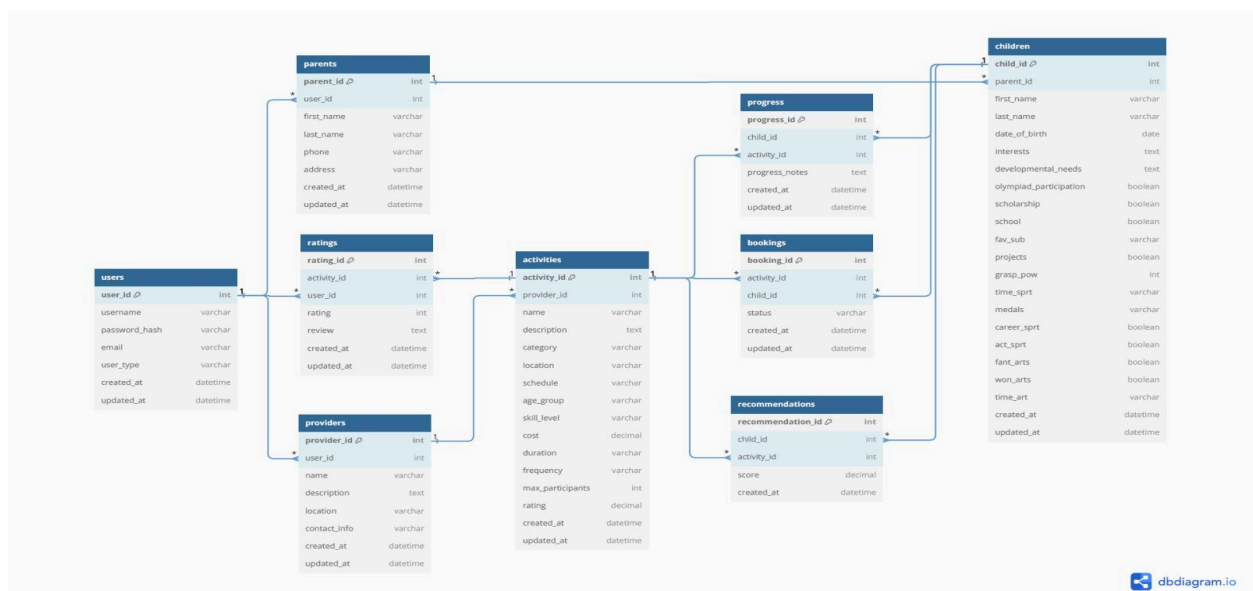
The system architecture integrates several components including data extraction, feature engineering, model training, recommendation generation, and visualization. The architecture uses Apache Airflow for orchestrating the ETL and model training pipelines, PostgreSQL for data storage, and Python for machine learning tasks.

Recommender System For Extracurricular Activities



2.2. Database Schema

The database schema includes tables for students, activities, and recommendations. It supports efficient querying and data manipulation necessary for generating personalized recommendations.



3. Data Exploration

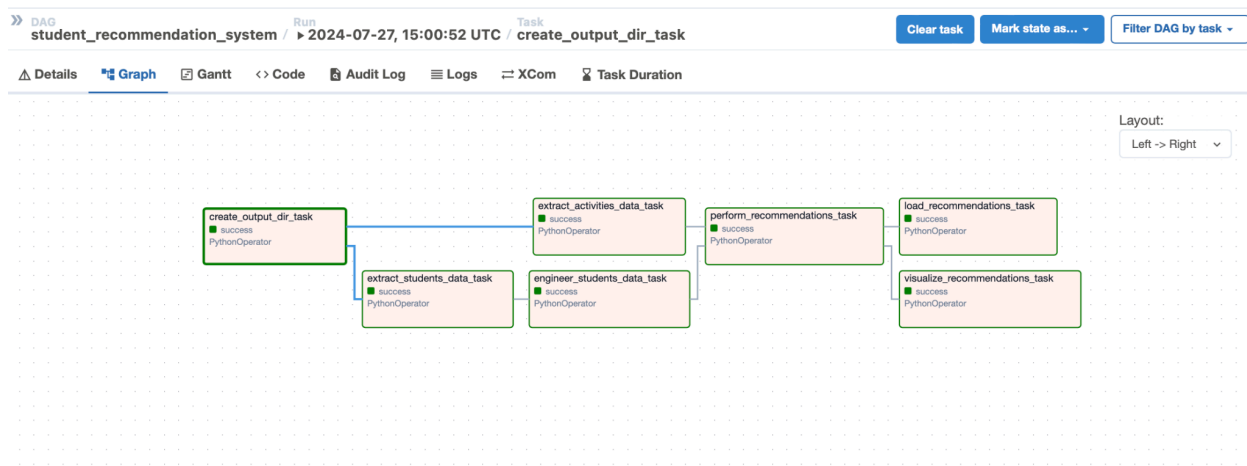
3.1. Dataset Description

The dataset includes attributes such as StudentID, Name, AcademicInterest, ExtracurricularActivities, Skills, Location, YearOfStudy, Major, GPA, Languages, ClubMemberships, and ResearchInterests. This comprehensive dataset provides a detailed view of students' academic and extracurricular profiles.

3.2. Exploratory Data Analysis

- **Descriptive Statistics:** Summary statistics of numerical attributes, including mean, median, and mode.
- **Visualizations:** Graphs and plots to visualize distributions, trends, and relationships within the data.

4. Data Processing



4.1. ETL Pipeline

- **Extraction:** Data is extracted from PostgreSQL using Apache Airflow.
- **Transformation:** Data is cleaned, preprocessed, and features are engineered using Python.
- **Loading:** Processed data is loaded back into PostgreSQL for further use.

4.2. Feature Engineering

- **Categorical Encoding:** Convert categorical variables to numerical form.
- **Normalization:** Scale numerical features to a standard range.

5. Model Development

5.1. Content-Based Filtering

- **Approach:** Used to provide personalized recommendations based on the features of activities and students' profiles. It's useful for tailoring recommendations to individual preferences.
- **Implementation:** Similarity scores are calculated based on attributes such as academic interest, skills, and extracurricular activities.

How It Works:

- **Feature Extraction:** This model relies on the features of the items themselves. For each student, it considers their interests and skills and compares these to the features of the activities.
- **Vectorization:** It uses TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer to transform text data into numerical vectors. This helps in capturing the importance of words in the context of the entire dataset.
- **Similarity Calculation:** Cosine similarity is used to measure the similarity between the student's feature vector and the activity's feature vectors. This determines how similar an activity is to a student's interests.
- **Recommendations:** Activities are recommended based on their similarity scores with the student's profile.

Why It Is Used:

- **Personalization:** Content-based filtering is highly personalized because it recommends items based on the specific attributes and interests of each student. If a student is interested in a particular skill or activity, the model will recommend activities that match those attributes.
- **Scalability:** It works well even with a relatively small amount of data about activities since it leverages the characteristics of the items themselves.
- **Independence:** It does not rely on the preferences or activities of other students, so it can be effective even if there's little historical data about other students' activities.

5.2. Collaborative Filtering

- **Approach:** Used to identify and suggest activities that similar students have found valuable. It leverages patterns in collective behavior and can uncover less obvious recommendations based on peer activity.
- **Implementation:** Utilizes a collaborative filtering approach to identify activities enjoyed by similar students.

5.3. Model Evaluation

- **Metrics:** Models are evaluated using precision, recall, and accuracy. These metrics provide insights into the effectiveness of recommendations.

- **Validation:** Data is split into training and testing sets to validate model performance.

How It Works:

- **User-User Similarity:** This method identifies similarities between students based on their behavior (e.g., extracurricular activities) and recommends activities liked by similar students.
- **Matrix Factorization:** Computes a similarity matrix based on students' activities.
- **Recommendation Generation:** It finds students who are similar to a target student and suggests activities that these similar students have engaged in but the target student has not.

Why It Is Used:

- **Discovery of Hidden Interests:** Collaborative filtering can identify patterns that might not be immediately apparent from the content alone. For example, students with similar interests may be found to enjoy activities that are not directly related to their stated interests but are popular among their peers.
- **Social Influence:** It leverages the collective behavior of the student population to make recommendations, which can be useful in suggesting activities that might be trending or popular among similar students.
- **Effective with Rich Data:** When there is ample data on student activities, collaborative filtering can provide highly relevant recommendations by analyzing similarities between users.

By combining both models, we aim to achieve a more comprehensive recommendation system that benefits from both individual personalization and collective insights.

6. Deployment

6.1. Serving Layer

- **Interface:** Develop a user-friendly interface with Flutter.
- **API:** Build RESTful APIs to serve recommendations.

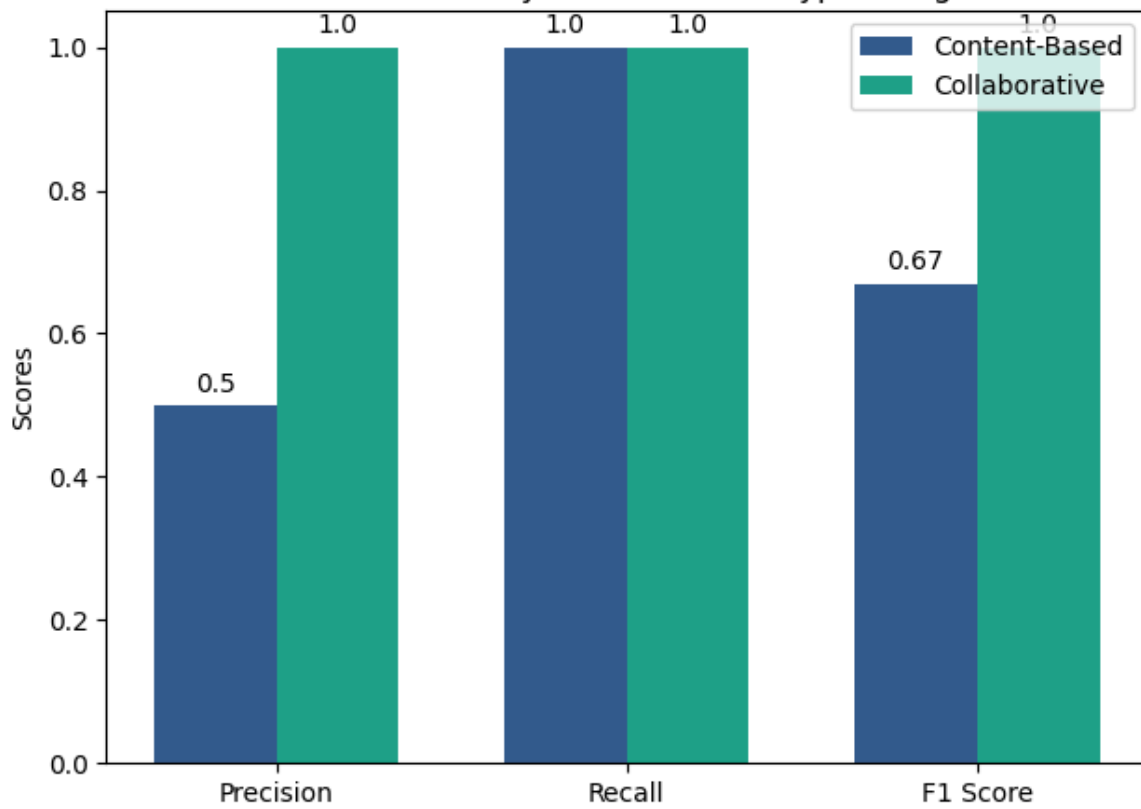
6.2. Monitoring & Feedback

- **Feedback Loop:** Mechanisms are implemented to collect user feedback on recommendations.
- **Continuous Improvement:** Feedback is used to retrain and improve models to enhance recommendation accuracy.

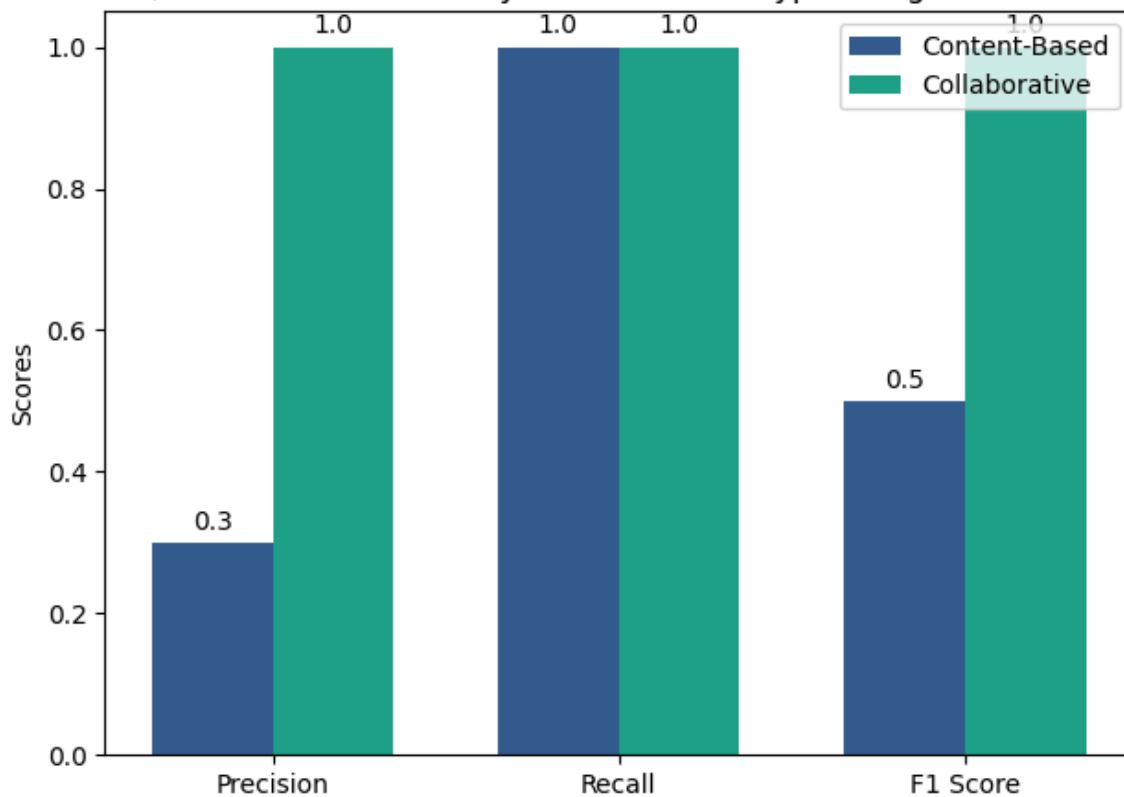
7. Results

7.1. Evaluation Outcomes

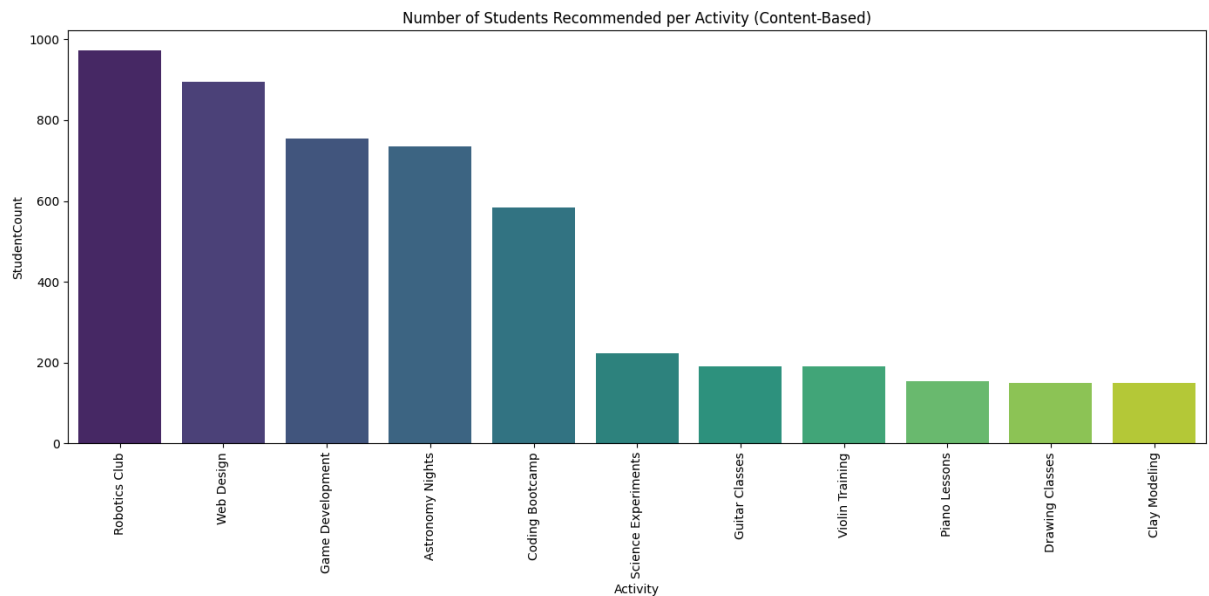
Precision, Recall and F1 Score by Recommender Type Using Cosine Similarity



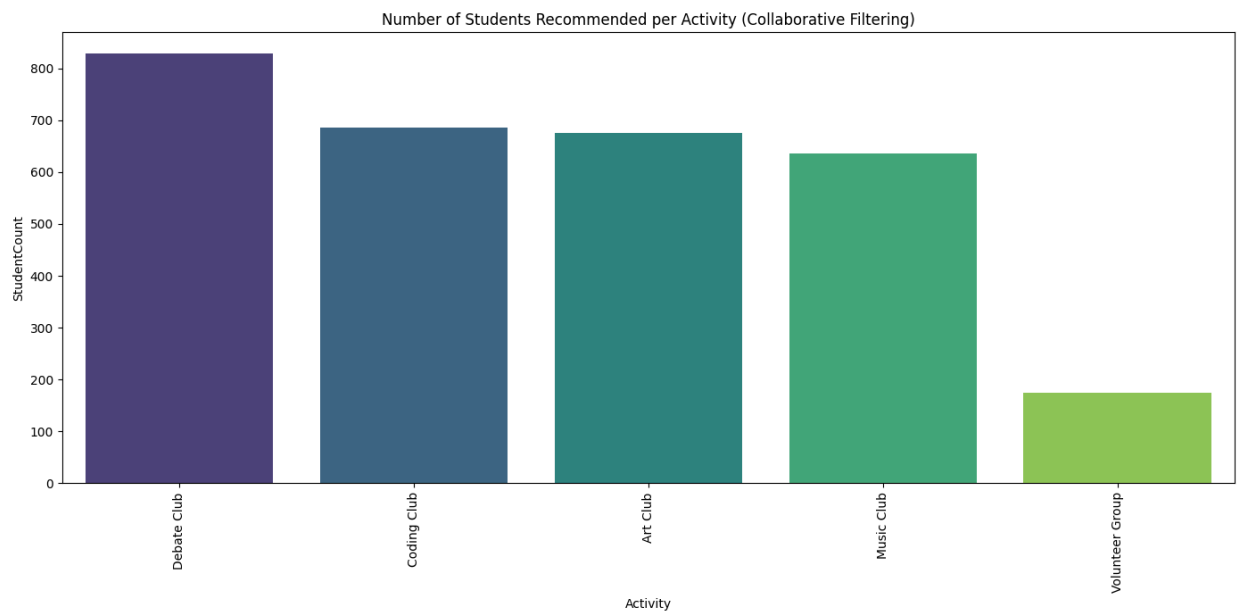
Precision, Recall and F1 Score by Recommender Type Using Euclidean distance



- **Content-Based Filtering:** Achieved Precision: 0.5, Recall: 1.0, F1 Score: 0.67



- **Collaborative Filtering:** Achieved Precision: 1.0, Recall: 1.0, F1 Score: 1.0



8. Conclusion

8.1. Summary

The project successfully developed a personalized recommender system for extracurricular activities. Leveraging advanced data processing and machine learning techniques, the system

provides tailored recommendations that help students engage more meaningfully in extracurricular activities.

8.2. Apache Airflow DAG Report: Student Recommendation System

1. Overview

This Apache Airflow Directed Acyclic Graph (DAG) is designed to generate student recommendations using two methods: content-based filtering and collaborative filtering. The recommendations are derived from students' data and activities data stored in PostgreSQL.

2. Default Arguments

The `default_args` dictionary sets the default parameters for the tasks in the DAG:

- `owner`: 'airflow'
- `depends_on_past`: False
- `email_on_failure`: False
- `email_on_retry`: False

3. Directories

Two directories are set up for output and visualizations:

- `output_dir`: /opt/airflow/dags/output
- `visualization_dir`: /opt/airflow/dags/visualizations

4. Functions

4.1 `create_output_dir()`

- Creates necessary directories for storing output and visualizations if they don't exist.

4.2 `extract_students_data_from_postgres(**kwargs)`

- Uses `PostgresHook` to extract student data from PostgreSQL and save it as a CSV file.
- Stores the data in XCom as JSON for further processing.

4.3 `extract_activities_data_from_postgres(**kwargs)`

- Extracts activities data from PostgreSQL and saves it as a CSV file.
- Stores the data in XCom as JSON.

4.4 `engineer_students_data(**kwargs)`

- Performs data engineering tasks such as stripping whitespace and sorting elements in several columns.
- Saves the engineered data as a CSV file and stores it in XCom as JSON.

4.5 `perform_recommendations(**kwargs)`

- **Content-Based Filtering:**
 - Uses TF-IDF Vectorizer to transform features into vectors and compute cosine similarity between student features (`'academicinterest', 'extracurricularactivities', 'skills', 'researchinterests'`) and activity features (`'name', 'description', 'category'`).
 - Selects top recommendations based on similarity scores and stores them in a DataFrame.
- **Collaborative Filtering:**
 - Computes similarity between students based on their extracurricular activities using cosine similarity.
 - Recommends activities based on the interests of similar students and stores these recommendations.

4.6 `load_recommendations(**kwargs)`

- Loads recommendations into the PostgreSQL `recommendations` table.
- Clears previous recommendations before inserting new ones.

4.7 `visualize_recommendations(**kwargs)`

- Aggregates recommendation counts for content-based and collaborative filtering methods.
- Generates bar plots to visualize the number of recommendations per activity and saves these plots as PNG files.

5. DAG Definition

The DAG `student_recommendation_system` is configured as follows:

- **Description:** Generates student recommendations based on content-based and collaborative filtering methods.
- **Schedule Interval:** `None` (manual trigger)
- **Start Date:** 1 day ago
- **Catchup:** False

6. Tasks

6.1 `create_output_task`

- Executes `create_output_dir()` to set up necessary directories.

6.2 `extract_students_data_task`

- Executes `extract_students_data_from_postgres()` to extract student data.

6.3 `extract_activities_data_task`

- Executes `extract_activities_data_from_postgres()` to extract activities data.

6.4 `engineer_students_data_task`

- Executes `engineer_students_data()` to engineer student data.

6.5 `perform_recommendations_task`

- Executes `perform_recommendations()` to generate recommendations.

6.6 `load_recommendations_task`

- Executes `load_recommendations()` to load recommendations into PostgreSQL.

6.7 `visualize_recommendations_task`

- Executes `visualize_recommendations()` to generate and save visualizations of recommendations.

7. Task Dependencies



















The tasks are executed in the following order:

1. `create_output_task` → `extract_students_data_task`,
`extract_activities_data_task`
2. `extract_students_data_task` → `engineer_students_data_task`
3. `extract_activities_data_task` → `perform_recommendations_task`
4. `engineer_students_data_task` → `perform_recommendations_task`
5. `perform_recommendations_task` → `load_recommendations_task`,
`visualize_recommendations_task`

8.3. Docker Compose Configuration Report

1. Overview

This Docker Compose configuration sets up a development environment for a recommendation system using various services. It includes PostgreSQL, PgAdmin, PySpark, Jupyter, and Apache Airflow. Each service is configured with specific settings and is connected through a common Docker network.

Name	Image	Status	CPU (%)	Port(s)
 dsproject		Running (6/7)	30.88%	
 airflow-webserver 643c3bcae7f9 	apache/airflow:2.9.2	Running	0.19%	8081:8080 
 airflow-scheduler 86b6562842f2 	apache/airflow:2.9.2	Running	6.09%	
 pgadmin 9ea99045a012 	dpage/pgadmin4:latest	Running	0.03%	5050:80 
 pyspark 47a054c1698b 	jupyter/pyspark-notebook:latest	Running	22.13%	8888:8888 
 jupyter 3eb46b2d9c14 	jupyter/base-notebook:latest	Running	0.01%	8889:8888 
 postgres 78eb8f9ac083 	postgres:latest	Running	2.43%	5432:5432 

2. Services

2.1 PostgreSQL

- **Image:** `postgres:latest`
- **Container Name:** `postgres`
- **Environment Variables:**
 - `POSTGRES_USER`: admin
 - `POSTGRES_PASSWORD`: admin
 - `POSTGRES_DB`: recommender_db
- **Ports:**
 - Exposes port `5432` for PostgreSQL
- **Networks:**
 - Connects to `recommender_network`

2.2 PgAdmin

- **Image:** `dpage/pgadmin4:latest`
- **Container Name:** `pgadmin`
- **Environment Variables:**

- PGADMIN_DEFAULT_EMAIL: admin@admin.com
 - PGADMIN_DEFAULT_PASSWORD: admin
- **Ports:**
 - Exposes port 5050 for PgAdmin web interface
- **Depends On:**
 - Depends on postgres
- **Networks:**
 - Connects to recommender_network

2.3 PySpark

- **Image:** jupyter/pyspark-notebook:latest
- **Container Name:** pyspark
- **Environment Variables:**
 - JUPYTER_ENABLE_LAB: "yes"
 - NB_USER: admin
 - NB_PASS: admin
- **Ports:**
 - Exposes port 8888 for Jupyter Notebook
- **Networks:**
 - Connects to recommender_network

2.4 Jupyter

- **Image:** jupyter/base-notebook:latest
- **Container Name:** jupyter
- **Environment Variables:**
 - JUPYTER_ENABLE_LAB: "yes"
 - NB_USER: admin
 - NB_PASS: admin
- **Ports:**
 - Exposes port 8889 for Jupyter Notebook
- **Networks:**
 - Connects to recommender_network

2.5 Airflow Initialization

- **Image:** apache/airflow:2.9.2
- **Container Name:** airflow-init
- **Environment Variables:**
 - AIRFLOW__CORE__EXECUTOR: LocalExecutor
 - AIRFLOW__DATABASE__SQL_ALCHEMY_CONN: postgresql+psycopg2://admin:admin@postgres/recommender_db
 - AIRFLOW__CORE__LOAD_EXAMPLES: 'False'
 - AIRFLOW__WEBSERVER__SECRET_KEY: "my_secret_key"

- `AIRFLOW__CORE__FERNET_KEY:`
"jC1902-VCgyLfIRLwICmwDRuW9fkdRw_p4hljhartj0="
 - `_AIRFLOW_WWW_USER_USERNAME:` admin
 - `_AIRFLOW_WWW_USER_PASSWORD:` admin
- **Entrypoint:**
 - Initializes the Airflow database and creates an admin user
- **Depends On:**
 - Depends on `postgres`
- **Networks:**
 - Connects to `recommender_network`

2.6 Airflow Webserver

- **Image:** `apache/airflow:2.9.2`
- **Container Name:** `airflow-webserver`
- **Environment Variables:**
 - `AIRFLOW__CORE__EXECUTOR:` LocalExecutor
 - `AIRFLOW__DATABASE__SQL_ALCHEMY_CONN:`
`postgresql+psycpg2://admin:admin@postgres/recommender_db`
 - `AIRFLOW__CORE__LOAD_EXAMPLES:` 'False'
 - `AIRFLOW__WEBSERVER__AUTHENTICATE:` 'True'
 - `AIRFLOW__WEBSERVER__AUTH_BACKEND:`
'airflow.www.security.password_auth'
 - `AIRFLOW__WEBSERVER__SECRET_KEY:` "my_secret_key"
 - `AIRFLOW__CORE__FERNET_KEY:`
"jC1902-VCgyLfIRLwICmwDRuW9fkdRw_p4hljhartj0="
 - `_AIRFLOW_WWW_USER_USERNAME:` admin
 - `_AIRFLOW_WWW_USER_PASSWORD:` admin
- **Command:**
 - Starts the Airflow webserver
- **Ports:**
 - Exposes port `8081` for Airflow web interface
- **Depends On:**
 - Depends on `airflow-init`, `postgres`, `pgadmin`, `pyspark`, `jupyter`
- **Volumes:**
 - Mounts the `./dags` directory to `/opt/airflow/dags`
- **Networks:**
 - Connects to `recommender_network`

2.7 Airflow Scheduler

- **Image:** `apache/airflow:2.9.2`
- **Container Name:** `airflow-scheduler`
- **Environment Variables:**
 - `AIRFLOW__CORE__EXECUTOR:` LocalExecutor

- `AIRFLOW__DATABASE__SQL_ALCHEMY_CONN:`
`postgresql+psycopg2://admin:admin@postgres/recommender_db`
- `AIRFLOW__CORE__LOAD_EXAMPLES:` 'False'
- `AIRFLOW__WEBSERVER__SECRET_KEY:` "my_secret_key"
- `AIRFLOW__CORE__FERNET_KEY:`
`"jC1902-VCgyLfIRLwICmwDRuW9fkdRw_p4hljhrtj0="`
- `_AIRFLOW_WWW_USER_USERNAME:` admin
- `_AIRFLOW_WWW_USER_PASSWORD:` admin
- **Command:**
 - Starts the Airflow scheduler
- **Depends On:**
 - Depends on `airflow-init`, `postgres`, `pgadmin`, `pyspark`, `jupyter`
- **Volumes:**
 - Mounts the `./dags` directory to `/opt/airflow/dags`
- **Networks:**
 - Connects to `recommender_network`

3. Network

- **Network Name:** `recommender_network`
- **Driver:** `bridge`

4. Summary

This Docker Compose setup creates a comprehensive environment for managing a recommendation system:

- **PostgreSQL** for database management.
- **PgAdmin** for database administration.
- **PySpark** and **Jupyter** for data processing and notebooks.
- **Airflow** for orchestrating data pipelines, including a webserver and scheduler.

8.4. Future Work

- **Model Enhancement:** Explore additional algorithms for better accuracy.
- **Scalability:** Optimize the system for larger datasets.
- **User Experience:** Improve the user interface based on feedback.

Appendices

A. Data Dictionary

This dataset contains information about students, their academic interests, extracurricular activities, skills, location, year of study, major, GPA, languages spoken, club memberships, and research interests. Here's a description of each field in the dataset:

StudentID: *A unique identifier for each student.*

Name: *The name of the student.*

AcademicInterest: *The field of study or academic interest of the student.*

ExtracurricularActivities: *Extracurricular activities the student is involved in.*

Skills: *Skills possessed by the student.*

Location: *The city or location where the student is currently based.*

YearOfStudy: *The year of study for the student (e.g., Freshman, Junior, Senior, Graduate).*

Major: *The major or field of study that the student is pursuing.*

GPA: *The Grade Point Average of the student.*

Languages: *Languages spoken or known by the student.*

ClubMemberships: *Memberships in various clubs or organizations.*

ResearchInterests: *Research interests or specialization of the student.*

This dataset provides a comprehensive overview of each student's academic and extracurricular background, including their skills and interests. It can be used for various analyses and insights into students' profiles and can help in making informed decisions related to academic and extracurricular activities.

B. References

- <https://ieeexplore.ieee.org/abstract/document/10179757>
- <https://www.kaggle.com/datasets/abtabm/hobby-prediction-basic>
- <https://www.kaggle.com/datasets/kamakshilahoti/student-extracurriculars-info/data>
- [Recommender Systems | ML-005 Lecture 16 | Stanford University | Andrew Ng \(youtube.com\)](#)
- Jadhav, A. (2023). *Applied Recommender Systems with Python: Build Recommender Systems with Deep Learning, NLP, and Graph-Based Techniques*. Packt Publishing.