#engine.py

**Purpose:**

To provide a detailed understanding of the engine.py file in KeeperAI’s recommendation engine. This Python script is responsible for recommending profiles using cosine similarity and displaying the results on a web page in frontend. Below is an explanation of what the code does.

1. **Import and initial setup:**

* Flask, render\_template, request, jsonify: Modules from Flask to create the web server, render HTML templates, handle incoming requests, and return JSON responses.
* pandas: Used for data manipulation and reading CSV files.
* flask\_cors.CORS: Enables Cross-Origin Resource Sharing (CORS), allowing the app to handle requests from different domains.
* cosine\_similarity: A function from scikit-learn that computes similarity scores between vectors.

**Configuration:**

* REC\_LIST\_LEN = 5: Limits the number of recommended profiles to 5.
* app = Flask(\_\_name\_\_, template\_folder='templates'): Initializes a Flask app and specifies the location of HTML templates.
* CORS(app): Enables CORS to allow cross-domain requests to the server.

1. **Loading User Data and Affinity Groups**

* users = pd.read\_csv('./firestore\_users.csv'): Reads user data from a CSV file generated by another script (firestore.py). This CSV contains user profiles and scores.
* Loads a list of affinity group names from groups.txt and stores them in the groups list. Each line in the file represents a different affinity group.

1. **Flask Routes**

* **Root Route (/)**: Renders the main HTML page (recommendation.html).
* **Profile Recommendation Route (/ProfRec)**: Handles profile recommendations based on a provided doc\_id from a POST request.

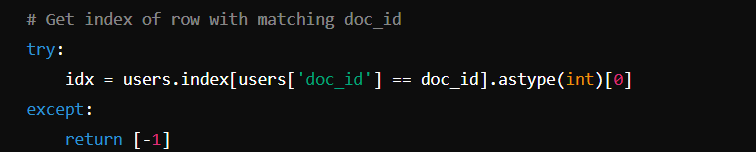
1. **Recommendation logic**

**Recommendation Function: get\_rec(doc\_id: str) -> list**

* **Function Purpose**: Recommends similar profiles based on a provided doc\_id.

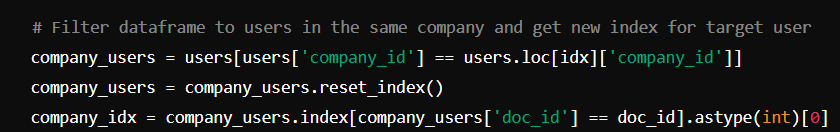
**Finding the User**:

* Finds the row index of the user in the users DataFrame using the doc\_id.
* If no match is found, it returns [-1] to indicate an error.



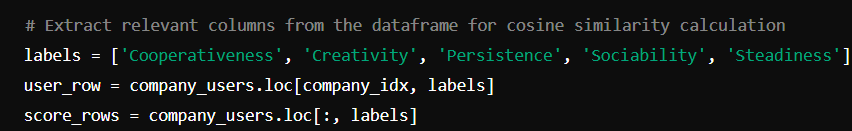
**Filtering by Company**:

* Filters the DataFrame to include only users within the same company as the input user.
* Resets the index to keep track of the new positions and identifies the current user's new index within this filtered subset.



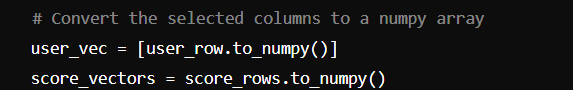
**Extracting Feature Vectors**:

* Selects personality scores and affinity group columns (labels) from the filtered DataFrame.
* user\_row: The specific row (vector) for the target user.
* score\_rows: All relevant rows (vectors) for users in the same company.



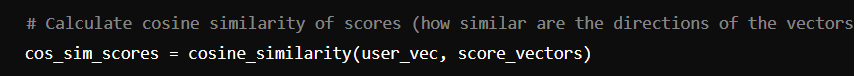
**Converting to NumPy Arrays**:

* Converts the selected rows to NumPy arrays for similarity calculations.



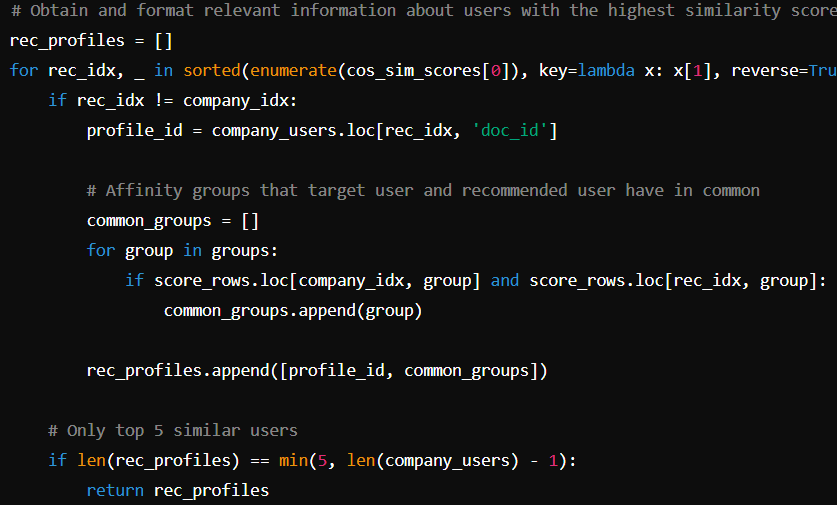
**Calculating Cosine Similarity**:

* Computes the cosine similarity between the target user’s vector and all other vectors in the score\_vectors. Cosine similarity measures how similar the vectors are in direction.



**Finding Top Recommendations**:

* Iterates through the similarity scores, sorted in descending order.
* Skips the target user and collects information about the most similar users, including their doc\_id and shared affinity groups.
* Stops once it has collected up to REC\_LIST\_LEN (5) recommendations or reaches the maximum possible number of other users in the company.



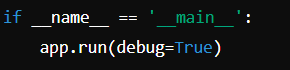
**Returning Recommendations**: Returns the list of recommended profiles, each containing the doc\_id and shared affinity groups.



1. **Running the recommendation engine application**

**Running the Server:**

This block starts the recommendation app in debug mode when the script is executed directly. The server runs locally and listens for incoming requests.



**Summary:**

* The engine.py script recommends user profiles based on personality scores and affinity group membership using cosine similarity.
* The recommendations are tailored to users within the same company and consider both personality alignment and shared group memberships.
* It reads data from CSV files, processes requests, calculates similarities, and returns formatted results to the frontend for display.