**TaskManager Class Documentation**

**Overview**

The TaskManager class is designed to provide a task management system with functionalities such as adding tasks, removing tasks, listing tasks, prioritizing tasks, and recommending tasks based on task descriptions. It leverages machine learning techniques, specifically Multinomial Naive Bayes classification, for task prioritization.

**Class Structure**

The TaskManager class includes the following methods:

**# \\_\\_init\\_\\_()**

- Description: Initializes a new TaskManager object.

- Parameters: None

- Functionality: Creates an empty task list DataFrame and loads tasks from a CSV file if it exists. Trains the machine learning model for task prioritization.

**# load\_tasks()**

- Description: Loads tasks from a CSV file if available.

- Parameters: None

- Functionality: Reads task data from a CSV file named 'tasks.csv' into the task list DataFrame.

**# save\_tasks()**

- Description: Saves the current task list to a CSV file.

- Parameters: None

- Functionality: Writes the task list DataFrame to a CSV file named 'tasks.csv'.

**# train\_model()**

- Description: Trains the machine learning model for task prioritization.

- Parameters: None

- Functionality: Utilizes CountVectorizer for text preprocessing and Multinomial Naive Bayes classifier for classification.

**# add\_task(description, priority)**

- Description: Adds a new task to the task list.

- Parameters:

- description (str): Description of the task.

- priority (str): Priority of the task (Low/Medium/High).

- Functionality: Creates a new DataFrame for the task and appends it to the task list. Saves the updated task list and retrains the model.

**# remove\_task(description)**

- Description: Removes a task from the task list.

- Parameters:

- description (str): Description of the task to be removed.

- Functionality: Filters out the task with the specified description from the task list. Saves the updated task list.

**# list\_tasks()**

- Description: Lists all tasks in the task list.

- Parameters: None

- Functionality: Prints the current task list DataFrame. If the task list is empty, it prints a message indicating no tasks are available.

**# prioritize\_tasks()**

- Description: Sorts tasks in the task list based on their priority level.

- Parameters: None

- Functionality: Sorts the task list DataFrame based on the 'priority' column in ascending order (Low < Medium < High). Saves the updated task list.

**# recommend\_task()**

- Description: Recommends a task based on machine learning predictions.

- Parameters: None

- Functionality: Uses the trained model to predict the priority of tasks and randomly selects a task with the predicted priority for recommendation.

**# main\_menu()**

- Description: Displays the main menu and handles user input.

- Parameters: None

- Functionality: Presents a menu with options to add, remove, list, prioritize, recommend tasks, and exit the application. Handles user input to execute corresponding operations.

**Usage**

1. Initialization:

- Create an instance of the TaskManager class.

2. Loading Tasks:

- Load tasks from a CSV file using the load\_tasks() method.

3. Task Operations:

- Add tasks using add\_task(description, priority) method.

- Remove tasks using remove\_task(description) method.

- List all tasks using list\_tasks() method.

- Prioritize tasks using prioritize\_tasks() method.

4. Machine Learning:

- Train the model for task prioritization using train\_model() method.

5. Recommendation:

- Get task recommendations based on predictions using recommend\_task() method.

6. User Interaction:

- Use main\_menu() method to interact with the task management system via a menu-driven interface.