CONTENT RECOMMENDATION SYSTEM

PROBLEM STATEMENT:

To Recommend top three posts for each user based on profile interests, past engagement, and content attributes.

Given Data:

1. Users.csv
2. Posts.csv
3. Engagements.csv

SOLUTION:

The objective is to develop a system (Model) that recommends the top three posts for each user by leveraging user-specific data.

To address this problem, I developed a machine learning model with logic that recommends the top posts for each user.

NOTE:
Before approaching the solution, the code files have been broken into four parts, you can follow the workflow in sequence, numbered from 1 to 4, as indicated in the file names. If any of the code files generate csv or pkl files, those outputs are also numbered accordingly to match the respective code file (EX: when "**1**-Data.ipynb" is executed, its output is saved as "**1**-result.csv")

UNDERSTANDING THE DATASETS:

1.Users.csv

user_id	Users unique ID.
age	Age of the users.
gender	Gender of the users.
top_3_interest	Users top 3 interested posts.

Past_engagement_score	How engaged were users with the posts earlier.

2.Posts.csv

Post_id	The post's unique ID.
creator_id	Creators/Users unique ID.
content_type	Type of content the post contains.
tag	The post's category.

3.Engagements.csv

user_id	Users unique ID.
post_id	The post's unique ID.
engagement	Whether the user is engaged with the post or not.

DATA PREPERATION:

Which features are need to be consider to train the data for model and What is the target variable here?

Let's see!!

My approach is to treat the Engagement feature as the target variable, solely for the purpose of building a classification model. However, the main goal is to recommend the top three posts that each user is most likely to be interested in. This can be achieved by using the predicted probabilities from the model, combined with a joins-based logic.

1. Initial Data Preparation

I started by using the Users.csv and Engagements.csv files. To assign the Engagement label to each user in Users.csv, I calculated the mean engagement score for each user from Engagements.csv. This average was added as a new column called Current_engagement_score.

Then, the Engagement column was derived based on the following logic:

- If Current_engagement_score > past_engagement_score,then the Engagement is set to 0 (i.e., the user is likely already engaged, though we still recommend posts).
- Otherwise, it is set to 1 (I.e., the user is less engaged and should be recommend relevant post.

2. Creating Interest column

I created the 'Interest' column by transforming the top_3_interest column into a single value, retaining only the top interest for each user.

3. Creating Interest Referance File

I Extracted all unique interests and saved them in a separate file called "Interests_list.csv", assigning a unique ID to each interest for future reference.

4. Finally Prepared Dataset

The Final dataset contains the following columns:

["user_id", "age", "gender", "past_engagement_score", "Interest", "Engagement"]

The Prepared dataset was saved as: "1-Prepared_data.csv"

DATA CLEANING & ENCODING:

To prepare the data for modelling, I performed the following steps:

- The dataset contains two categorical features: gender and Interest.
- The gender feature was encoded using Numpy arrays,
  'M' -> 1
  'F' -> 0
  Others -> -1
- The Interest feature was encoded using the "Interests_list.csv" file. By performing a left join between main dataset and this file, a new numerical feature called 'ID' was added to represent 'Interest'.
- After encoding, the original 'Interest' feature was dropped, as it was no longer needed since the 'ID' was there.

- Now, the cleaned dataset contains the following features ["user_id", "age", "gender", "past_engagement_score", "ID", "Engagement"]

The final cleaned dataset was saved as “2-Model_Input.csv”

MACHINE LEARNING MODEL:

1. Model Selection

- I initially experimented with a Decision Tree model, but its performance was relatively low. Although linear models could also be considered, I decided to proceed with a Random Forest due to its ability to handle non-linear relationships and feature interactions more effectively.
- Choosing a Random Forest does come with some risk, especially given the limited number of data samples. However, with careful management of overfitting – primarily through proper hyperparameter tuning, the model was performed well.

2. Model Building

- Now, we had a well-prepared input dataset for the model. The initial set of input features included:

  ["age", "gender", "past_engagement_score", "ID"] with “Engagement” as the target variable.
- The model was first trained using a Random Forest Classifier, and hyperparameter tuning was performed using GridSearchCV to identify the optimal hyperparameters. After fitting the model, feature importance was analysed based on the Gini impurity.

Feature	Gini
past_engagement_score	0.71
age	0.18
ID	0.08
gender	0.01

- From the feature importance results, the gender feature was found to have minimal impact and was therefore removed. The model was then retrained using top three features: ["age", "past_engagement_score", "ID"]

- Hyperparameter tuning was applied again on this reduced feature set and finally fitted the model with these hyperparameters.

Hyperparameters	value	Description
max_depth	5	Limits how deep each tree can grow.
min_samples_leaf	5	Controls the minimum number of samples required in leaf node.
max_features	log2	Controls how many features are randomly considered at each split [i.e., log2(x)]
n_estimators	50	Specifies the number of trees in the forest.

3. Model Evaluation

- Accuracy: 96
- AUC ROC: 99.2
- Confusion Matrix:
  [[22 1]
  [3 24]]
- F1 Score: 92.3

The Final model was saved using Pickle and stored as “3-Model RF 2025-09-30.pkl”

4. Model Testing

Up to this point, the model was successfully built. Now, we move on to the core objective – generating the top 3 post recommendations for each user.

- A new dataset was created with the features: [“user_id”, “age”, “past_engagement_score”]
- Although the gender feature was not used during modelling, it was still encoded to maintain consistency.
- To generate interest combinations for each user, a **cross join** (Cartesian product) was performed between this dataset and the “Interests_list.csv” file. For example, if there are 30 users and 10 unique interests, this operation results in m*n rows (30*10=300) – one for every possible user-interest pair.

- Using the model (3-Model RF 2025-09-30.pkl), predict_proba was applied to this expanded dataset to obtain the engagement probability for each ID (numerical version of Interests).
- The numeric ID values were then mapped back to their corresponding ‘Interest’ names via a left join with “Interests_list.csv”.
- The results were sorted in descending order of probability for each user to identify the most likely interests. Only the top 3 interests per user were reatained.
- The ‘Interest’ column was grouped by User_id and join values with ',' and renamed to top_3_interest to match the required output format.
- The final output consists of:
  [“user_id”, “top_3_interests”]

This Final result was saved as “4-Output.csv”

→ Task has been completed, and the model is now capable of recommending the top 3 interests for each user based on historical data. However, new user data will continue to arrive over time. Continuously retraining the model manually every time is not efficient. So, it should self-learn the data and need to adjust their hyperparameters. To address this, the next phase is updating the system to SELF-LEARNING ML MODEL.