Dataset Files:

- **user.csv**: contains `user_id`, `products`, `reviews`.
  Example: `7360263,359,0`
- **games.csv**: contains game metadata like `app_id`, `title`, `date_release`, `win`, `mac`, `linux`, `rating`, `positive_ratio`, `user_reviews`, `price_final`, `price_original`, `discount`, `steam_deck`.
  Example:
  `13500,Prince of Persia: Warrior Within™,2008-11-21,true,false,false,Very Positive,84,2199,9.99,9.99,0.0,true`
- **recommendations.csv**: contains recommendation info with `app_id`, `helpful`, `funny`, `date`, `is_recommended`, `hours`, `user_id`, `review_id`.
  Example: `975370,0,0,2022-12-12,true,36.3,51580,0`
- **games_metadata.json**: JSON lines file (not a valid JSON array). Each line is a JSON object like:

```
{
  "app_id": 226560,
  "description": "Escape Dead Island is a Survival-Mystery adventure...",
  "tags": [
      "Zombies",
      "Adventure",...
  ]
}
```

  Note: This file is not parsable using the standard `json.load()` because it's structured as `{}\n{}\n{}` (one object per line). You must parse it **line by line**.

Functional Requirements:

1. Use 80% of the data for training and 20% for testing.
2. Implement and compare the following algorithms for recommendation/classification:
   - K-Nearest Neighbors (KNN)
   - K-Means Clustering
   - Naïve Bayes
   - One algorithm of your choice (e.g., Decision Tree, Random Forest, etc.)
3. Compare algorithm performance using: **F1 Score, Precision, Recall, Accuracy**.
4. Provide data **visualizations** (matplotlib, seaborn, or Streamlit charts).
5. Include a section comparing all algorithms and explain the reason behind the selected algorithm.
6. Try different strategies to **improve results** if performance is low (e.g., feature engineering, hyperparameter tuning, etc.).
7. Build an **interactive Streamlit interface**:
   - Allow the user to input a user_id and show top game recommendations.
   - Provide summary statistics and visual feedback (ratings distribution, recommendation trends, etc.).

Academic Rubrics to Fulfill:

1. **Dataset Selection** – 5 marks
2. **Classification Implementation** – 8 marks
3. **Clustering Implementation** – 8 marks
4. **Performance Evaluation (F1 Score, Precision, Recall, Accuracy)** – 5 marks
5. **Visualization** – 5 marks
6. **Algorithm Comparison and Justification** – 5 marks
7. **Efforts to Improve Results** – 4 marks

Make sure to structure the project clearly with separate modules or functions for:

- Data preprocessing (including parsing `games_metadata.json` line by line)
- Feature engineering
- Model training & evaluation
- Streamlit frontend