To build a "Product Recommendation System" Python project using pandas, scikit-learn, and Streamlit, here is a detailed guide aligned with your objectives and mini-guide, along with example code snippets:

**Steps for Building Product Recommendation System**

**1. Load Product + User Rating Data**

* Load datasets: product info (name, thumbnails, description) and user ratings (user\_ids, product\_ids, ratings).
* Use pandas for easy data manipulation.

**2. Preprocess and Vectorize Data**

* Prepare rating matrix for collaborative filtering (users × products).
* For content filtering, extract product features (categories, tags) and convert them to vectors (e.g., TF-IDF).

**3. Build Collaborative + Content Filters**

* Collaborative filtering: Use user-item matrix and techniques like cosine similarity or matrix factorization.
* Content filtering: Use product feature vectors and similarity metrics.
* Combine results or create hybrid recommender.

**4. Input Product Name to Get Suggestions**

* Provide input widget in Streamlit for product name.
* Find similar products using collaborative and/or content filtering.

**5. Show Thumbnails/Details in UI**

* Display recommended products with images and details using Streamlit components.

**6. Export Recommendations to CSV**

* Allow export of recommendation results to CSV file.

**Example Implementation Outline**

python

**import** pandas **as** pd

**from** sklearn.metrics.pairwise **import** cosine\_similarity

**from** sklearn.feature\_extraction.text **import** TfidfVectorizer

**import** streamlit **as** st

*# Sample load data*

products = pd.DataFrame({

'product\_id': [1, 2, 3, 4],

'name': ['Phone A', 'Phone B', 'Laptop A', 'Laptop B'],

'description': ['smartphone with camera', 'smartphone with battery', 'lightweight laptop', 'gaming laptop'],

'thumbnail': ['url1', 'url2', 'url3', 'url4']

})

ratings = pd.DataFrame({

'user\_id': [1, 1, 2, 3, 3],

'product\_id': [1, 2, 2, 3, 4],

'rating': [5, 3, 4, 5, 2]

})

*# Create user-item matrix for collaborative filtering*

user\_item\_matrix = ratings.pivot(index='user\_id', columns='product\_id', values='rating').fillna(0)

*# Collaborative filtering similarity between products*

collab\_sim = cosine\_similarity(user\_item\_matrix.T)

*# Content filtering based on descriptions*

tfidf = TfidfVectorizer()

tfidf\_matrix = tfidf.fit\_transform(products['description'])

content\_sim = cosine\_similarity(tfidf\_matrix)

**def** get\_product\_index(name):

**return** products[products['name'] == name].index[0]

**def** recommend\_products(product\_name, top\_n=3):

idx = get\_product\_index(product\_name)

*# Combine collaborative and content similarity (simple average)*

sim\_scores = (collab\_sim[idx] + content\_sim[idx]) / 2

top\_indices = sim\_scores.argsort()[::-1][1:top\_n+1] *# exclude self*

**return** products.iloc[top\_indices]

*# Streamlit UI*

st.title('Product Recommendation System')

selected\_product = st.selectbox('Choose a product:', products['name'])

**if** st.button('Get Recommendations'):

recommendations = recommend\_products(selected\_product)

st.write("Recommended Products:")

**for** \_, row **in** recommendations.iterrows():

st.image(row['thumbnail'], width=100)

st.write(f"\*\*{row['name']}\*\*")

st.write(row['description'])

*# Export to CSV option*

csv = recommendations.to\_csv(index=False)

st.download\_button(label="Download Recommendations CSV", data=csv, file\_name='recommendations.csv', mime='text/csv')

**Explanation**

* Loads product and user ratings data.
* Constructs collaborative filtering by user-item ratings matrix and cosine similarity.
* Builds content filtering using TF-IDF on product descriptions.
* Combines similarity scores for hybrid recommendations.
* Streamlit UI accepts product name input, shows thumbnails/details, and allows exporting recommendations as CSV.

**Deliverables**

* Recommender Streamlit app (above).
* Test data embedded in code or external files.
* Exported recommendation report CSV.

If more details on data sources, model enhancements, or UI improvements are needed, further help can be provided. Would a file upload feature for data or integration with real product APIs be useful?To build a "Product Recommendation System" Python project using pandas, scikit-learn, and Streamlit, the workflow involves loading product and user ratings data, preprocessing, building collaborative and content filtering models, recommending products based on user input, showing product images/details in the UI, and allowing export of recommendations to CSV.

**Key Steps**

1. Load the product and rating datasets with pandas.
2. Create a user-item rating matrix for collaborative filtering.
3. Vectorize product descriptions using TF-IDF for content-based filtering.
4. Calculate similarity matrices for both collaborative and content filters.
5. Take user input product name in Streamlit and combine similarity scores to generate recommendations.
6. Display product thumbnails and details in the Streamlit app.
7. Provide an option to export recommended products' data to a CSV file.

**Example Code Outline**

python

**import** pandas **as** pd

**from** sklearn.metrics.pairwise **import** cosine\_similarity

**from** sklearn.feature\_extraction.text **import** TfidfVectorizer

**import** streamlit **as** st

*# Sample data*

products = pd.DataFrame({

'product\_id': [1,2,3,4],

'name': ['Phone A','Phone B','Laptop A','Laptop B'],

'description': ['smartphone with camera','smartphone with battery','lightweight laptop','gaming laptop'],

'thumbnail': ['url1','url2','url3','url4'] *# Replace with actual image URLs or paths*

})

ratings = pd.DataFrame({

'user\_id': [1,1,2,3,3],

'product\_id': [1,2,2,3,4],

'rating': [5,3,4,5,2]

})

*# Collaborative Filtering: user-item matrix & cosine similarity*

user\_item\_matrix = ratings.pivot(index='user\_id', columns='product\_id', values='rating').fillna(0)

collab\_sim = cosine\_similarity(user\_item\_matrix.T)

*# Content Filtering: TF-IDF on descriptions*

tfidf = TfidfVectorizer()

tfidf\_matrix = tfidf.fit\_transform(products['description'])

content\_sim = cosine\_similarity(tfidf\_matrix)

**def** get\_product\_index(name):

**return** products[products['name'] == name].index[0]

**def** recommend\_products(product\_name, top\_n=3):

idx = get\_product\_index(product\_name)

combined\_sim = (collab\_sim[idx] + content\_sim[idx]) / 2

top\_idx = combined\_sim.argsort()[::-1][1:top\_n+1]

**return** products.iloc[top\_idx]

*# Streamlit UI*

st.title("Product Recommendation System")

selected\_product = st.selectbox("Choose a product:", products['name'])

**if** st.button("Get Recommendations"):

recs = recommend\_products(selected\_product)

st.write("Recommended Products:")

**for** \_, row **in** recs.iterrows():

st.image(row['thumbnail'], width=100)

st.write(f"\*\*{row['name']}\*\*")

st.write(row['description'])

csv = recs.to\_csv(index=False)

st.download\_button("Download Recommendations CSV", csv, file\_name="recommendations.csv")

**Deliverables**

* Streamlit app UI that recommends products with thumbnails/details.
* Test data embedded or external files for products and ratings.
* CSV export of recommendations.