PRODUCT RECOMMENDATION SYSTEM FOR E-COMMERCE PLATFORM

1. INTRODUCTION

This documentation provides a comprehensive overview of the recommendation system developed for an e-commerce platform. The goal of this project is to implement a machine learning model that recommends alternative products based on user search queries. The system identifies products with similar technical specifications but from different brands and suppliers, offering a range of price points.

2. PROJECT OVERVIEW

Objective

The primary objective is to create a recommendation system that can:

- Suggest alternative products based on user search queries.

- Identify products with similar technical specifications across various brands and suppliers.

- Provide a range of price points for the recommended products.

Datasets

Two datasets are used for this implementation:

1. **Laptops Dataset** (`ecommerce_data.csv`): Contains information about various laptop models.

  - Columns: `Company`, `TypeName`, `Inches`, `ScreenResolution`, `Cpu`, `Ram`, `Memory`, `Gpu`, `OpSys`, `Weight`, `Price`

2. **Electronics Dataset** (`electronics_product.csv`): Contains information about various electronics products.

  - Columns: `name`, `main_category`, `sub_category`, `image`, `link`, `ratings`, `no_of_ratings`, `discount_price`, `actual_price`

3. IMPLEMENTATION DETAILS

3.1. Data Preprocessing

Laptops Dataset

1. Loading Data: Loaded the dataset using `pandas`.
2. Dropping Unnecessary Columns: Removed the `Unnamed: 0` column which was not needed.
3. Handling Missing Values: Dropped rows with missing values.
4. Lowercasing: Converted all text columns to lowercase to ensure uniformity.
5. Feature Combination: Combined various columns into a single `combined_features` column for vectorization.

Electronics Dataset

1. Loading Data: Loaded the dataset using `pandas`.
2. Handling Missing Values: Dropped rows with missing values in the `name` column.
3. Lowercasing: Converted all text columns to lowercase to ensure uniformity.
4. Feature Combination: Combined `main_category` and `sub_category` into a single `combined_features` column for vectorization.

3.2. Model Development

TF-IDF Vectorization

1. Initialization: Initialized `TfidfVectorizer` for both laptops and electronics datasets.
2. Fitting: Fitted the vectorizer on the `combined_features` column of each dataset to create TF-IDF matrices.
3. Cosine Similarity: Computed cosine similarity between products within each dataset to determine similarity scores.

Recommendation Functions

1. Laptop Recommendations:

- Query Preparation: Created a query DataFrame and combined features similar to the training data.
- Vectorization: Transformed the query into a TF-IDF vector.
- Similarity Calculation: Calculated cosine similarity between the query and all products.
- Top Recommendations: Sorted products based on similarity scores and selected the top 5.

2. Electronics Recommendations:

- Query Preparation: Created a query DataFrame and combined features.
- Vectorization: Transformed the query into a TF-IDF vector.
- Similarity Calculation: Calculated cosine similarity between the query and all products.
- Top Recommendations: Sorted products based on similarity scores and selected the top 5.

3.3. User Interface

1. Login System: Implemented a simple login system using Streamlit to manage access to the recommendation features.
2. Sidebar Filters: Provided options for users to select product types, and specific attributes (e.g., Company, CPU) for laptops, and categories for electronics.
3. Display Recommendations: Displayed recommended products with relevant details, including price and specifications for laptops, and price, ratings, and links for electronics.

4. CODE EXPLANATION

4.1. Importing Libraries

```
import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer
```

from sklearn.metrics.pairwise import cosine_similarity

import streamlit as st

4.2. Preprocessing Data

Load datasets

laptops_df = pd.read_csv('ecommerce_data.csv')

electronics_df = pd.read_csv('electronics_product.csv')

Preprocessing for laptops

laptops_df = laptops_df.drop(columns=['Unnamed: 0'])

laptops_df = laptops_df.dropna()

laptops_df = laptops_df.applymap(lambda s: s.lower() if type(s) == str else s)

laptops_df['combined_features'] = laptops_df['Company'] + ' ' + laptops_df['TypeName'] + ' ' + laptops_df['ScreenResolution'] + ' ' + laptops_df['Cpu'] + ' ' + laptops_df['Ram'] + ' ' + laptops_df['Memory'] + ' ' + laptops_df['Gpu'] + ' ' + laptops_df['OpSys']

Preprocessing for electronics

electronics_df = electronics_df.dropna(subset=['name'])

electronics_df = electronics_df.applymap(lambda s: s.lower() if type(s) == str else s)

electronics_df['combined_features'] = electronics_df['name'] + ' ' + electronics_df['main_category'] + ' ' + electronics_df['sub_category']

4.3. Model Initialization and Training

python

TF-IDF Vectorizer and Cosine Similarity for laptops

laptops_tfidf_vectorizer = TfidfVectorizer(stop_words='english') laptops_tfidf_matrix = laptops_tfidf_vectorizer.fit_transform(laptops_df['combined_features'])

laptops_cosine_sim = cosine_similarity(laptops_tfidf_matrix, laptops_tfidf_matrix)

TF-IDF Vectorizer and Cosine Similarity for electronics

electronics_tfidf_vectorizer = TfidfVectorizer(stop_words='english')

electronics_tfidf_matrix = electronics_tfidf_vectorizer.fit_transform(electronics_df['combined_features'])

electronics_cosine_sim = cosine_similarity(electronics_tfidf_matrix, electronics_tfidf_matrix)

4.4. Recommendation Functions

python

Function for laptop recommendations

```
def get_laptop_recommendations(query):
    query_df = pd.DataFrame([query], columns=laptops_df.columns[:-2])
    query_df['combined_features'] = query_df['Company'] + ' ' + query_df['TypeName'] + ' ' + query_df['ScreenResolution'] + ' ' + query_df['Cpu'] + ' ' + query_df['Ram'] + ' ' + query_df['Memory'] + ' ' + query_df['Gpu'] + ' ' + query_df['OpSys']
    query_tfidf = laptops_tfidf_vectorizer.transform(query_df['combined_features'])
    query_sim = cosine_similarity(query_tfidf, laptops_tfidf_matrix)
    sim_scores = list(enumerate(query_sim[0]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:6]
    product_indices = [i[0] for i in sim_scores]
    return laptops_df.iloc[product_indices]
```

Function for electronics recommendations

```
def get_electronics_recommendations(query):
    query_df = pd.DataFrame([query])
    query_df = query_df.dropna(subset=['main_category', 'sub_category'])
```

```
    query_df['combined_features'] = query_df['main_category'] + ' ' + query_df['sub_category']
    query_tfidf = electronics_tfidf_vectorizer.transform(query_df['combined_features'])
    query_sim = cosine_similarity(query_tfidf, electronics_tfidf_matrix)
    sim_scores = list(enumerate(query_sim[0]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:6]
    product_indices = [i[0] for i in sim_scores]
    return electronics_df.iloc[product_indices]
```

4.5. Streamlit Interface

```
python
 Streamlit app setup
st.title('Product Recommendation System')

 User login
if 'login' not in st.session_state:
    st.session_state.login = False

if not st.session_state.login:
    username = st.text_input('Username')
    password = st.text_input('Password', type='password')

    if st.button('Login'):
        if username == 'admin' and password == 'admin':
            st.session_state.login = True
        else:
            st.error('Invalid username or password')
else:
```

```
st.sidebar.title('Search Filters')
product_type = st.sidebar.selectbox('Product Type', ['Laptops', 'Electronics'])

if product_type == 'Laptops':
    Laptop search filters and recommendations
    ...

elif product_type == 'Electronics':
    Electronics search filters and recommendations
    ...

if st.button('Logout'):
    st.session_state.login = False
```

5. CONCLUSION

The implemented recommendation system is capable of providing product suggestions based on user queries. It uses TF-IDF vectorization and cosine similarity to find similar products across different brands and suppliers. The system is built with Streamlit to provide a user-friendly interface for interacting with the recommendation engine.