**Hyper-Personalized Financial Recommendation System**

**Overview:** The hyper-personalized recommendation system aims to deliver highly tailored financial product recommendations to individual users by leveraging their transaction history, financial profiles, market trends, and sentiment analysis.

**Key Features**

1. **User Profiling**:
   * Captures user details such as income, risk profile, goals, and transaction history.
   * Generates unique embeddings for personalized financial insights.
2. **Embedding Generation**:
   * Leverages **DistilBERT** to convert transaction history into meaningful feature vectors for similarity matching.
   * Embeddings are used to match user preferences with financial products.
3. **Real-Time Market Trends**:
   * Fetches dynamic market data (e.g., S&P 500 index) using the **Alpha Vantage API**.
   * Incorporates market trends like Bullish, Bearish, or Neutral into recommendations.
4. **Sentiment Analysis**:
   * Analyzes financial news headlines using **VADER** and **TextBlob** to assess product-related sentiments.
   * Enhances the recommendation engine with sentiment-driven scores.
5. **Contextual Recommendations**:
   * Combines user preferences, sentiment insights, and market trends to rank and recommend top financial products.
6. **Educational Resources**:
   * Provides users with financial education materials tailored to their goals, such as retirement planning and investment strategies.
7. **API Integration**:
   * Offers a /recommend endpoint via Flask for generating recommendations dynamically.

**Setup Instructions**

**Prerequisites**

1. Python (3.7 or higher).
2. Required Python libraries:
   * pandas
   * numpy
   * scikit-learn
   * Flask
   * vaderSentiment
   * textblob
   * transformers
   * requests
3. An Alpha Vantage API key for fetching market data. (access key is required)

**Installation**

1. Clone the repository or download the code files.
2. Install the necessary dependencies: pip install pandas numpy scikit-learn flask vaderSentiment textblob transformers requests
3. Place the data.xlsx file containing user data in the specified directory.

**Running the Application**

1. Start the Flask server:
2. Access the /recommend endpoint by sending GET requests with user-specific parameters:
3. <http://localhost:5000/recommend?userID=U001&city=Mumbai&goal=Retirement&risk_profile=Conservative&budget=100000>



**How It Works**

1. **Data Loading**:
   * User information and transaction history are loaded from data.xlsx.
   * Transaction history is embedded using NLP models for semantic similarity computation.
2. **Embedding Generation**:
   * The generate\_embeddings function transforms transaction history using **DistilBERT**.
3. **Market Trend Analysis**:
   * Retrieves real-time data via **Alpha Vantage** to assess market trends impacting recommendations.
4. **Sentiment Analysis**:
   * Uses VADER and TextBlob to analyze financial news and determine product sentiment scores.
5. **Recommendation Engine**:
   * Computes similarity scores between user embeddings and product embeddings using **cosine similarity**.
   * Incorporates dynamic contextual scoring (goal matching, risk profile, sentiment, etc.).
   * Returns the top 3 recommended products based on weighted scores.

**Dependencies**

| **Library** | **Purpose** |
| --- | --- |
| Flask | Hosts the API for generating recommendations |
| pandas | Manages user data and transaction history |
| numpy | Computes similarity scores and embeddings |
| scikit-learn | Implements cosine similarity |
| vaderSentiment | Performs sentiment analysis on financial headlines |
| TextBlob | Conducts additional sentiment analysis |
| transformers | Generates embeddings using DistilBERT |
| requests | Fetches external data like market trends |