Smart Search Tool for Analytics Vidhya Courses:

# Approach and Methodology

# 1. Problem Understanding

The objective of this project is to build a smart search tool that recommends the most relevant free courses on Analytics Vidhya based on a user's query. The challenge involves handling natural language input, understanding course descriptions, and ranking them based on semantic similarity to the query.

# 2. Embedding Model Selection

For understanding the meaning of user queries and course descriptions, an embedding model is essential. Embedding models convert textual data into numerical representations, enabling comparison based on the similarity of the text.

Chosen Model: I used the distilbert-base-nli-stsb-mean-tokens model from the Sentence Transformers library.

- Why DistilBERT?  
 - It is a lightweight and fast version of BERT (Bidirectional Encoder Representations from Transformers), making it suitable for real-time search applications.  
 - It has been fine-tuned on natural language inference (NLI) tasks, which makes it good for understanding semantic similarities.  
 - The mean-tokens pooling method aggregates information across the sentence for better sentence-level embeddings.

- Why Sentence Transformers?  
 - Sentence transformers offer easy-to-use methods for generating sentence and text embeddings.  
 - They provide pre-trained models optimized for similarity and clustering tasks.

# 3. Similarity Calculation

After embedding the query and course descriptions, the similarity between the query and all available course descriptions is calculated using cosine similarity. This is done to rank the courses.

- Cosine Similarity: Cosine similarity measures the cosine of the angle between two vectors, providing a good measure for comparing text embeddings.  
 - It ranges from -1 (opposite direction) to 1 (same direction), where 1 indicates high similarity.

# 4. Ranking Courses

Once similarities are calculated between the query embedding and each course embedding, the courses are ranked based on the highest cosine similarity score.  
Top 3 Recommendations: We return the top 3 most similar courses to provide the user with multiple options, increasing the chances of finding a suitable course.

# 5. Data Input Source

The course data (title and description) is imported from a CSV file, which contains information on various programming and data science courses. The course descriptions are used to generate embeddings, and course titles are returned as recommendations.

# 6. Deployment Consideration

The smart search tool is deployed using a Flask web application:  
- The model and embeddings are loaded at the server startup to ensure fast response times.  
- The user submits their query via a search box, and the top 3 relevant courses are returned in real-time.

# 7.Conclusion

This smart search tool leverages advanced natural language processing techniques to provide personalized course recommendations. By employing the distilbert-base-nli-stsb-mean-tokens model for embeddings and cosine similarity for ranking, the application effectively meets user needs in an intuitive and efficient manner.