# AGENTIC AI LABORATORY


**Presented by:**

**Samir Kumar (2023386036)**

**Under the supervision of:**

**Mr. Ayush Kumar Singh**


**BACHELOR OF TECHNOLOGY**

In

**COMPUTER SCIENCE AND ENGINEERING**


**SHARDA SCHOOL OF ENGINEERING AND TECHNOLOGY**

**GREATER NOIDA**

# Working of the code in the python notebook

The notebook is a tutorial demonstrating five progressively complex strategies for splitting text (chunking) to prepare data for Large Language Models (LLMs). It uses the `langchain` library to implement these methods.

### Level 1: Character Splitting

This section demonstrates the simplest method: splitting text into fixed-size chunks based on character count, ignoring content structure.

- **Manual Implementation:** The code first shows a Python loop that slices a string into 35-character chunks.
- **LangChain Implementation:** It then introduces `CharacterTextSplitter` from LangChain.
    - **Parameters:** It configures `chunk_size` (characters per chunk) and `chunk_overlap` (number of characters shared between adjacent chunks to preserve context).
    - **Workflow:** The `create_documents` method is used to convert raw text into a list of "Document" objects.

### Level 2: Recursive Character Text Splitting

This section addresses the rigidity of Level 1 by using `RecursiveCharacterTextSplitter`. This method attempts to split text hierarchically using a specific list of separators.

- **Separators:** The splitter prioritizes double newlines (paragraphs), then single newlines, then spaces, and finally individual characters.
- **Workflow:**
    - The code initializes the splitter with a `chunk_size` of 65 and `chunk_overlap` of 0.
    - It processes a sample text about "superlinear returns."
    - The output shows that the splitter keeps sentences together where possible. When the `chunk_size` is increased to 450, it successfully keeps entire paragraphs intact because the chunk size is large enough to accommodate the text between double newline separators.

### Level 3: Document Specific Splitting

This level moves beyond generic text to structured formats like Markdown and Code.

- **Markdown:** Uses `MarkdownTextSplitter`. The code demonstrates splitting a document with Headers (#, ##, ###). The splitter recognizes these headers and tries to keep sections together.
- **Python:** Uses `PythonCodeTextSplitter`. It splits code based on class and function definitions (`class`, `def`) to ensure executable blocks remain intact.
- **JavaScript:** Uses `RecursiveCharacterTextSplitter.from_language` with `Language.JS`. It utilizes JS-specific separators like `function`, `const`, `let`, and brackets to split a JavaScript snippet logically.

**Level 4: Semantic Splitting**

This method splits text based on meaning rather than syntax. It uses embeddings to determine where topics change.

- **Setup:** The code installs `langchain_experimental` and sets up `OpenAIEmbeddings`.
- **Workflow:**
    1. **Embeddings:** It generates vector embeddings for sentences in a sample text about Tesla's financial results.
    2. **Similarity:** It calculates the cosine similarity between adjacent sentences. High similarity implies the sentences belong to the same topic; low similarity indicates a topic shift.
    3. **Chunking:** The `SemanticChunker` uses a percentile threshold (e.g., 70th percentile) to identify "breakpoints" where similarity drops significantly, creating chunks at those points.

**Level 5: Agentic Splitting**

This is an experimental "agent-like" approach where an LLM is directly asked to determine the split points.

- **Prompting:** The code initializes a `ChatOpenAI` model (referenced as "gpt-5-nano" in the code, likely a placeholder or custom alias) with a temperature of 0.
- **Instructions:** It sends the raw text to the model with a specific prompt: "You are a text chunking expert. Split this text into logical chunks... Put '<<<SPLIT>>>' between chunks".
- **Processing:** The code takes the LLM's text response, splits the string by the `<<<SPLIT>>>` marker, cleans up whitespace, and prints the resulting logical chunks.