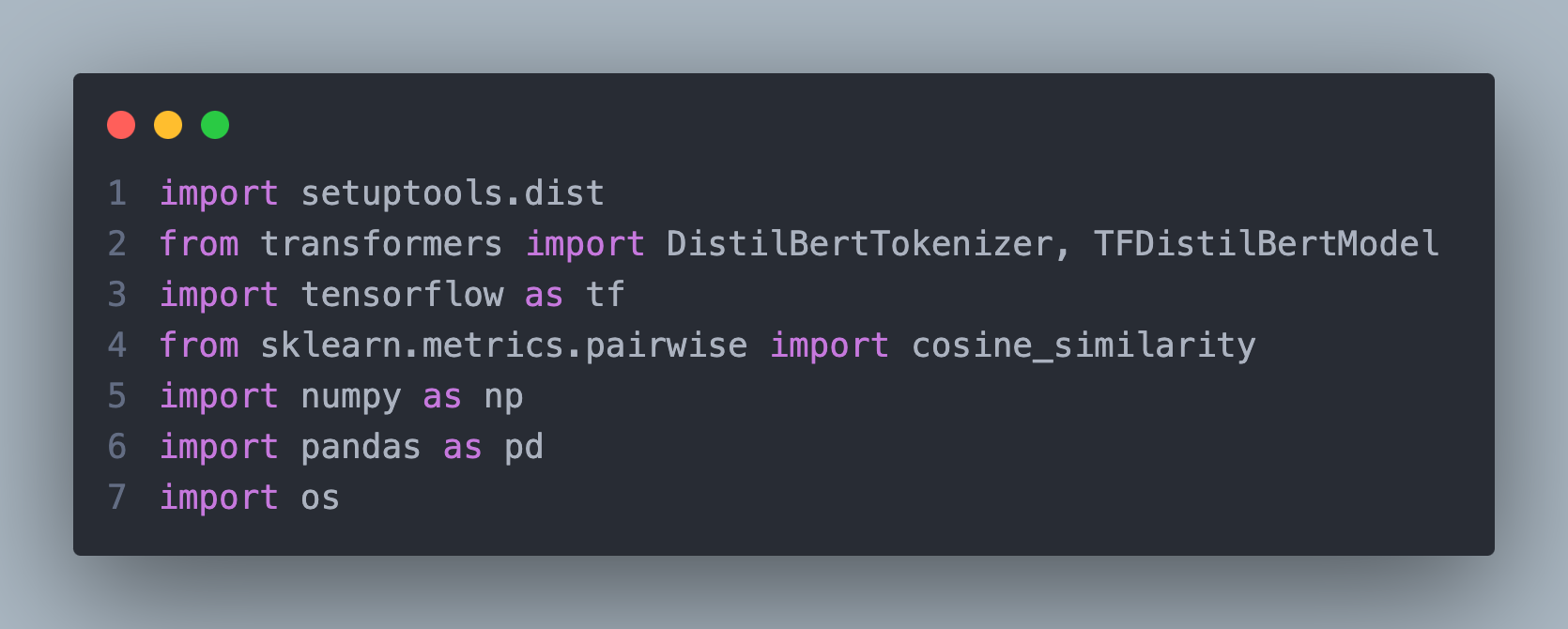
**Explanation of the Context based item recommendation model**

**Importing Necessary Libraries:**

****

This kernel imports essential Python libraries for natural language processing, machine learning, and file handling:

* **setuptools.dist:** Manages dependencies.
* **transformers:** Provides pre-trained models like DistilBERT for generating embeddings.
* **tensorflow:** Used for tensor operations and running the DistilBERT model.
* **sklearn.metrics.pairwise:** Calculates cosine similarity between embeddings.
* **numpy:** Handles numerical operations.
* **pandas:** Manages structured data.
* **os:** Handles directory and file paths.

**Loading Data:**

A black rectangle with white text

Description automatically generated

Loads the dataset containing cleaned product information from the Coles supermarket into a Pandas DataFrame.

**Initializing the DistilBERT Model:**

A screen shot of a computer program

Description automatically generated

* **DistilBertTokenizer:** Prepares text data for the DistilBERT model.
* **TFDistilBertModel:** Loads the pre-trained DistilBERT model for generating text embeddings.
* Both models use the weights **distilbert-base-uncased**.

**Embedding Generation for Product Names:**

A computer screen shot of a code

Description automatically generated

* Extracts product names and generates embeddings using the DistilBERT model.
* Each product name is tokenized, padded to a fixed length (20), converted to tensors, and passed through DistilBERT.
* Mean pooling is applied to generate a single embedding for each product name.

**Batch Processing for Efficiency:**

A screenshot of a computer program

Description automatically generated

* Processes product names in batches of size 32 for faster computation.
* Tokenizes and generates embeddings for each batch.
* Saves embeddings for each batch as **.npy** files in the specified directory.

**Saving and Loading all Embeddings:**

**A screen shot of a computer

Description automatically generated**

* Combines and saves all embeddings into a single **.npy** file for future use.
* Loads the saved embeddings from the file for further processing.

**Compute Similarity Matrix:**

**A black rectangle with white text

Description automatically generated**

* Computes pairwise cosine similarity between product embeddings.
* The result is a similarity matrix where each entry represents the similarity between two products.

**Product Recommendation:**

**A computer screen shot of a program code

Description automatically generated**

* Defines a recommendation function that finds similar products for a given product based on the similarity matrix.
* Returns the top **n** most similar products (excluding the queried product itself).
* Example recommendations are generated for the first product in the dataset.

**Output:**

**A screen shot of a computer code

Description automatically generated**

* Prints the shape of the loaded embeddings, confirming successful computation.
* Displays the folder where embeddings are saved.
* Outputs recommendations for the example product.

This modular and efficient approach ensures scalability, accuracy, and easy integration of product similarity and recommendation functionality.