APPLIED MACHINE LEARNING FOR TEXT ANALYSIS PROJECT REPORT

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLGY

SUBMITTED BY

B.JAYANTH KUMAR	**(2300039077)**
M.MANDEEP CHAKRAVARTHY	**(2300039108)**
V. CHAITANYA	**(2300039116)**
SK. VALIYA BABJI	**(2300039117)**

Department of Computer Science and Engineering

KONERU LAKSHMAIAH EDUCATION FOUNDATION
Green Fields, Vaddeswaram.

ACKNOWLEDGEMENTS

We would like to express my sincere gratitude to **Prof. Malladi Ravishankar**, Department of Computer Science and Engineering (CSE-2), **KL Deemed to be University**, for his valuable guidance, encouragement, and continuous support throughout the completion of my **Applied Machine Learning for Text Analysis Project Report** titled *"Plagiarism Detection Mini-Project."*

His insightful suggestions and constant motivation greatly contributed to the successful completion of this project.

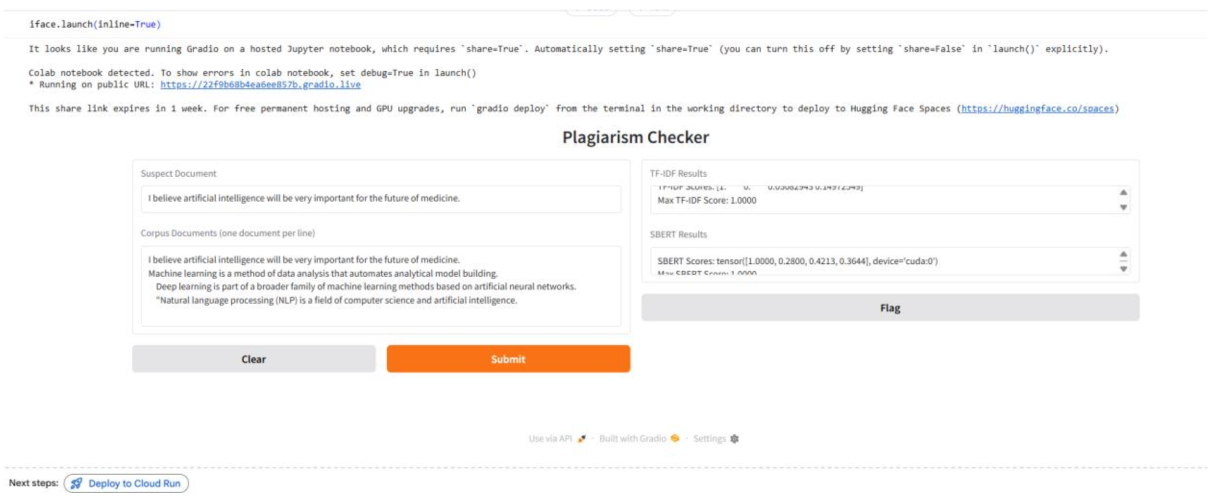
I would also like to thank my university, **KL Deemed to be University**, for providing a conducive learning environment and encouraging students to undertake practical and research-oriented projects that enhance technical knowledge and hands-on skills in emerging technologies like **Machine Learning** and **Artificial Intelligence**.

INDEX

PLAGIARISM DETECTION MINI-PROJECT

1. Introduction

This document provides a detailed overview of the plagiarism detection mini-project, implemented in Python. The project utilizes both traditional TF-IDF (Term Frequency-Inverse Document Frequency) and advanced SBERT (Sentence-BERT) models to identify potential plagiarism between a suspect document and a corpus of documents. A web-based graphical user interface (GUI) has been developed using the Gradio library to facilitate easy interaction with the tool.



```
iface.launch(inline=True)
```

```
It looks like you are running Gradio on a hosted Jupyter notebook, which requires `share=True`. Automatically setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly).

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://22f9b68b4ea6ee857b.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (https://huggingface.co/spaces)
```

Plagiarism Checker

Suspect Document

I believe artificial intelligence will be very important for the future of medicine.

Corpus Documents (one document per line)

I believe artificial intelligence will be very important for the future of medicine.
Machine learning is a method of data analysis that automates analytical model building.
Deep learning is part of a broader family of machine learning methods based on artificial neural networks.
"Natural language processing (NLP) is a field of computer science and artificial intelligence.

TF-IDF Results

```
Max TF-IDF Score: 1.0000
```

SBERT Results

```
SBERT Scores: tensor([1.0000, 0.2800, 0.4213, 0.3644], device='cuda:0')
```

Clear
Submit
Flag
Use via API · Built with Gradio · Settings
Next steps: Deploy to Cloud Run

2. Setup and Installation

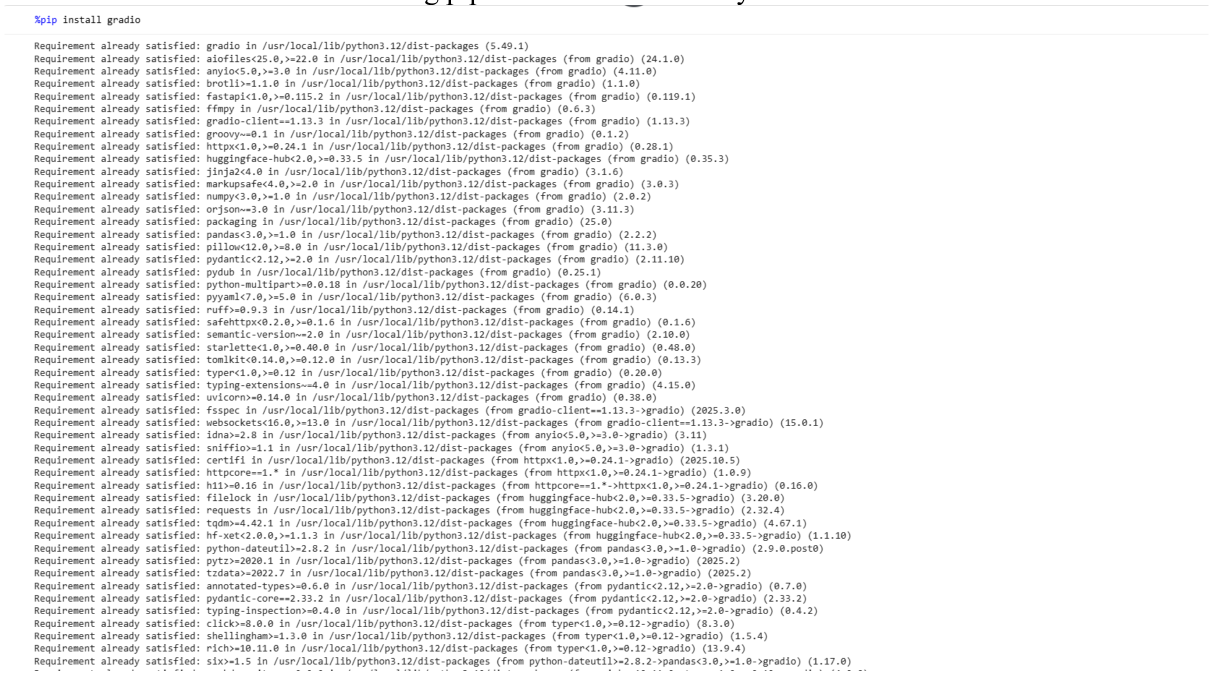
This section outlines the necessary steps to set up and run the project in a Google Colab environment.
2.1 Prerequisites

- Google Account to access Google Colab.

2.2 Installing Libraries
The project requires the following Python libraries:

- nltk: For natural language processing tasks like tokenization and stop word removal.
- scikit-learn: For implementing the TF-IDF model and cosine similarity.
- sentence-transformers: For implementing the SBERT model.
- gradio: For building the web-based GUI.

These libraries can be installed using pip in a code cell within your Colab notebook.



```
%pip install gradio
```

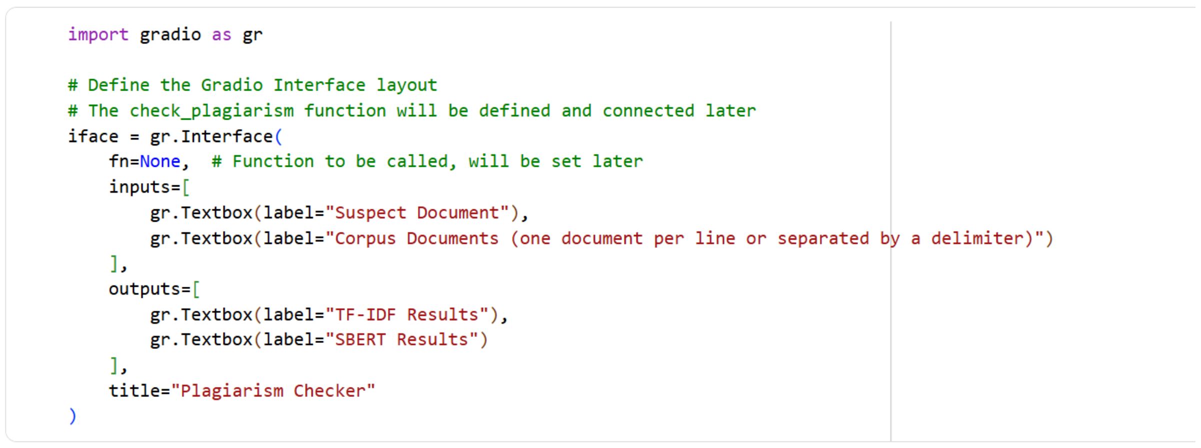
```
Requirement already satisfied: gradio in /usr/local/lib/python3.12/dist-packages (5.49.1)
Requirement already satisfied: aiofiles<25.0,>=22.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (24.1.0)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (4.11.0)
Requirement already satisfied: brotli>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (1.1.0)
Requirement already satisfied: fastapi<1.0,>=0.115.2 in /usr/local/lib/python3.12/dist-packages (from gradio) (0.119.1)
Requirement already satisfied: ffmpy in /usr/local/lib/python3.12/dist-packages (from gradio) (0.6.3)
Requirement already satisfied: gradio-client==1.13.3 in /usr/local/lib/python3.12/dist-packages (from gradio) (1.13.3)
Requirement already satisfied: groovy~=0.1 in /usr/local/lib/python3.12/dist-packages (from gradio) (0.1.2)
Requirement already satisfied: httpx<1.0,>=0.24.1 in /usr/local/lib/python3.12/dist-packages (from gradio) (0.28.1)
Requirement already satisfied: huggingface-hub<2.0,>=0.33.5 in /usr/local/lib/python3.12/dist-packages (from gradio) (0.35.3)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (3.1.6)
Requirement already satisfied: markupsafe<4.0,>=2.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (3.0.3)
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (2.0.2)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (3.11.3)
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from gradio) (25.0)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (11.3.0)
Requirement already satisfied: pydantic<2.12,>=2.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (2.11.10)
Requirement already satisfied: pydub in /usr/local/lib/python3.12/dist-packages (from gradio) (0.25.1)
Requirement already satisfied: python-multipart>=0.0.18 in /usr/local/lib/python3.12/dist-packages (from gradio) (0.0.20)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (6.0.3)
Requirement already satisfied: ruff>=0.9.3 in /usr/local/lib/python3.12/dist-packages (from gradio) (0.14.1)
Requirement already satisfied: safehttpx<0.2.0,>=0.1.6 in /usr/local/lib/python3.12/dist-packages (from gradio) (0.1.6)
Requirement already satisfied: semantic-version~=2.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (2.10.0)
Requirement already satisfied: starlette<1.0,>=0.40.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (0.48.0)
Requirement already satisfied: tomlkit<0.14.0,>=0.12.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (0.13.3)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.12/dist-packages (from gradio) (0.20.0)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (4.15.0)
Requirement already satisfied: uvicorn>=0.14.0 in /usr/local/lib/python3.12/dist-packages (from gradio) (0.38.0)
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (from gradio-client==1.13.3->gradio) (2025.3.0)
Requirement already satisfied: websockets<16.0,>=13.0 in /usr/local/lib/python3.12/dist-packages (from gradio-client==1.13.3->gradio) (15.0.1)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.12/dist-packages (from anyio<5.0,>=3.0->gradio) (3.11)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.12/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packages (from httpx<1.0,>=0.24.1->gradio) (2025.10.5)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-packages (from httpx<1.0,>=0.24.1->gradio) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-packages (from httpcore==1.*->httpx<1.0,>=0.24.1->gradio) (0.16.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<2.0,>=0.33.5->gradio) (3.20.0)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<2.0,>=0.33.5->gradio) (2.32.4)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<2.0,>=0.33.5->gradio) (4.67.1)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<2.0,>=0.33.5->gradio) (1.1.10)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas<3.0,>=1.0->gradio) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic<2.12,>=2.0->gradio) (0.7.0)
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.12/dist-packages (from pydantic<2.12,>=2.0->gradio) (2.33.2)
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.12/dist-packages (from pydantic<2.12,>=2.0->gradio) (0.4.2)
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.12/dist-packages (from typer<1.0,>=0.12->gradio) (8.3.0)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.12/dist-packages (from typer<1.0,>=0.12->gradio) (1.5.4)
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.12/dist-packages (from typer<1.0,>=0.12->gradio) (13.9.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas<3.0,>=1.0->gradio) (1.17.0)
```

3. Code Explanation

This section explains the core components of the Python code used for plagiarism detection.

3.1 Importing Libraries and Helper Functions

This part of the code imports all necessary libraries and defines helper functions, such as text preprocessing for TF-IDF.



```python
import gradio as gr

# Define the Gradio Interface layout
# The check_plagiarism function will be defined and connected later
iface = gr.Interface(
    fn=None,  # Function to be called, will be set later
    inputs=[
        gr.Textbox(label="Suspect Document"),
        gr.Textbox(label="Corpus Documents (one document per line or separated by a delimiter)")
    ],
    outputs=[
        gr.Textbox(label="TF-IDF Results"),
        gr.Textbox(label="SBERT Results")
    ],
    title="Plagiarism Checker"
)
```

3.2 TF-IDF Model

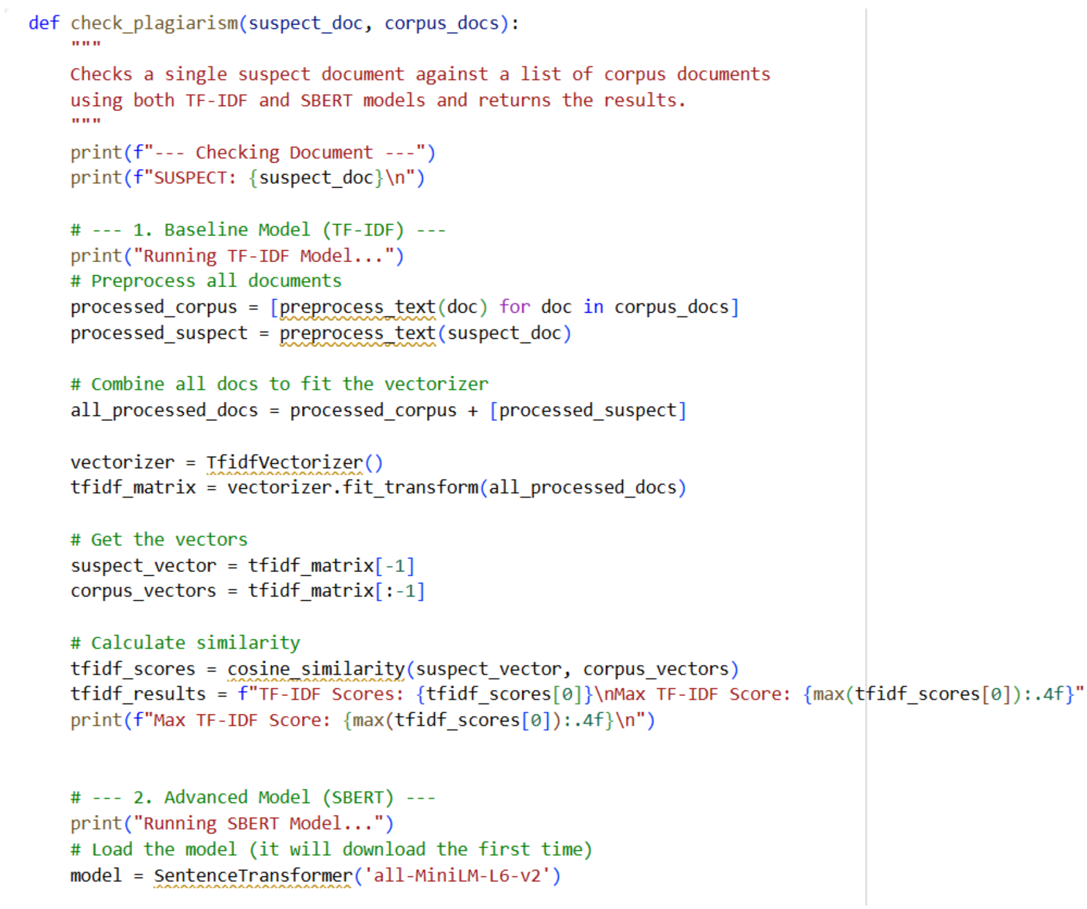
Explain how the TF-IDF model is used to represent documents as vectors and how cosine similarity is calculated to find similarities between the suspect document and corpus documents.

3.3 SBERT Model

Explain how the SBERT model is used to generate sentence embeddings and how cosine similarity is applied to measure the semantic similarity between documents.

3.4 check_plagiarism Function

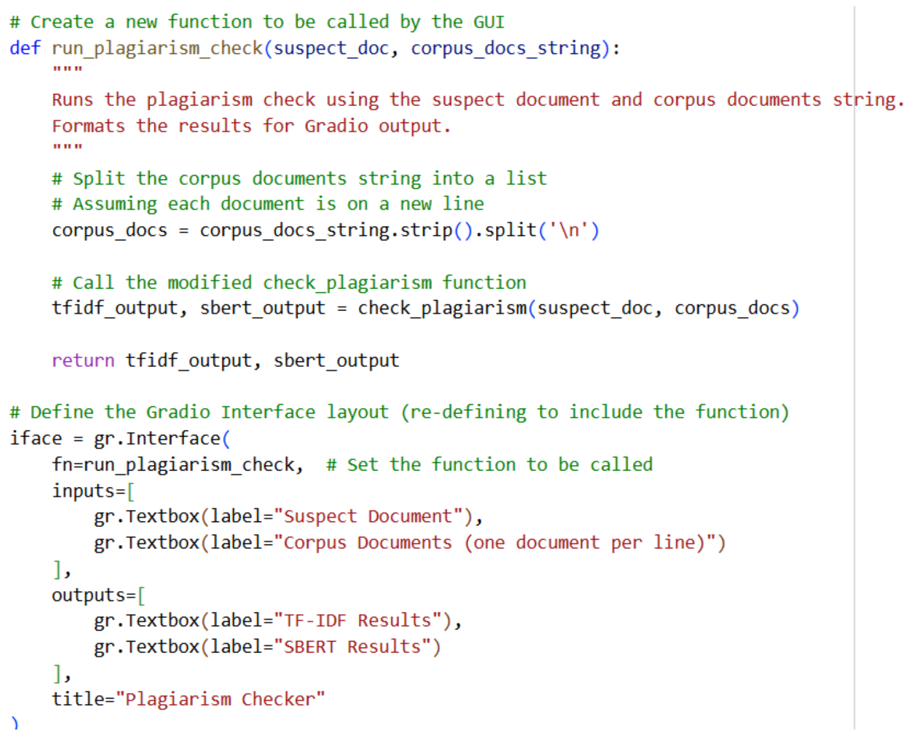
Detail the check_plagiarism function, which takes the suspect document and corpus documents as input, applies both TF-IDF and SBERT models, and returns the similarity results.



```python
def check_plagiarism(suspect_doc, corpus_docs):
    """
    Checks a single suspect document against a list of corpus documents
    using both TF-IDF and SBERT models and returns the results.
    """
    print(f"--- Checking Document ---")
    print(f"SUSPECT: {suspect_doc}\n")

    # --- 1. Baseline Model (TF-IDF) ---
    print("Running TF-IDF Model...")
    # Preprocess all documents
    processed_corpus = [preprocess_text(doc) for doc in corpus_docs]
    processed_suspect = preprocess_text(suspect_doc)

    # Combine all docs to fit the vectorizer
    all_processed_docs = processed_corpus + [processed_suspect]

    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(all_processed_docs)

    # Get the vectors
    suspect_vector = tfidf_matrix[-1]
    corpus_vectors = tfidf_matrix[:-1]

    # Calculate similarity
    tfidf_scores = cosine_similarity(suspect_vector, corpus_vectors)
    tfidf_results = f"TF-IDF Scores: {tfidf_scores[0]}\nMax TF-IDF Score: {max(tfidf_scores[0]):.4f}"
    print(f"Max TF-IDF Score: {max(tfidf_scores[0]):.4f}\n")


    # --- 2. Advanced Model (SBERT) ---
    print("Running SBERT Model...")
    # Load the model (it will download the first time)
    model = SentenceTransformer('all-MiniLM-L6-v2')
```

3.5 run_plagiarism_check Function

Explain the run_plagiarism_check function, which acts as an intermediary between the Gradio GUI and the check_plagiarism function. It handles input formatting from the GUI and output formatting for the GUI.



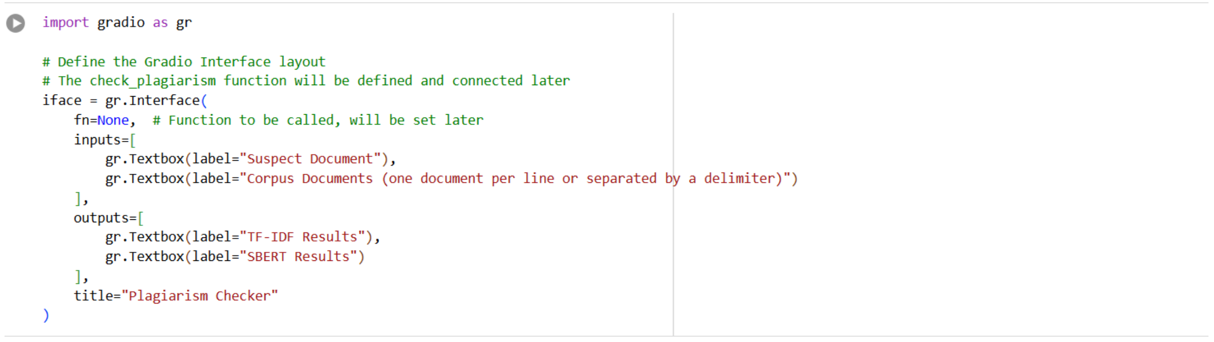
```python
# Create a new function to be called by the GUI
def run_plagiarism_check(suspect_doc, corpus_docs_string):
    """
    Runs the plagiarism check using the suspect document and corpus documents string.
    Formats the results for Gradio output.
    """
    # Split the corpus documents string into a list
    # Assuming each document is on a new line
    corpus_docs = corpus_docs_string.strip().split('\n')

    # Call the modified check_plagiarism function
    tfidf_output, sbert_output = check_plagiarism(suspect_doc, corpus_docs)

    return tfidf_output, sbert_output

# Define the Gradio Interface layout (re-defining to include the function)
iface = gr.Interface(
    fn=run_plagiarism_check,  # Set the function to be called
    inputs=[
        gr.Textbox(label="Suspect Document"),
        gr.Textbox(label="Corpus Documents (one document per line)")
    ],
    outputs=[
        gr.Textbox(label="TF-IDF Results"),
        gr.Textbox(label="SBERT Results")
    ],
    title="Plagiarism Checker"
)
```

4. GUI Implementation with Gradio

This section describes how the Gradio library was used to create the interactive GUI.

4.1 Designing the Interface Layout

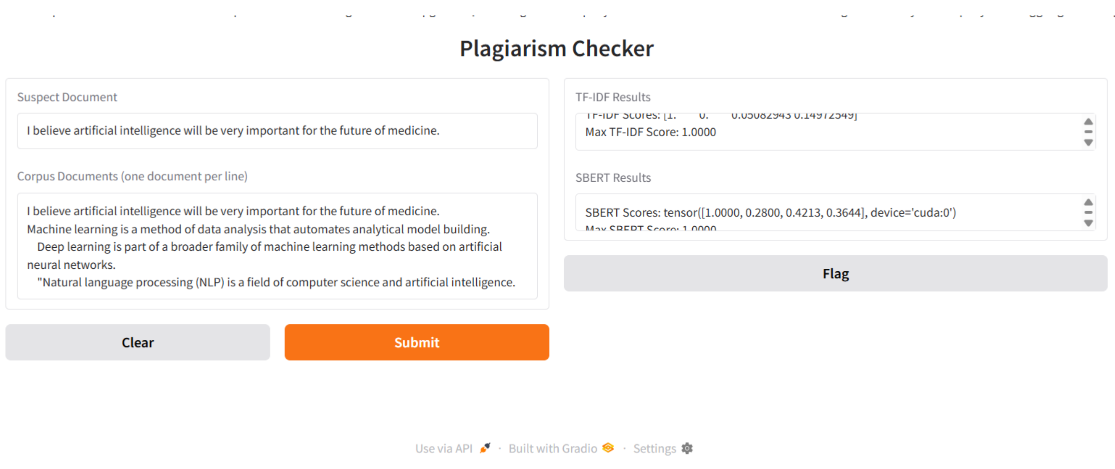
Explain how the gradio.Interface class was used to define the layout of the GUI, including the input textboxes for the suspect document and corpus documents, and the output textboxes for the TF-IDF and SBERT results.



```python
import gradio as gr

# Define the Gradio Interface layout
# The check_plagiarism function will be defined and connected later
iface = gr.Interface(
    fn=None,  # Function to be called, will be set later
    inputs=[
        gr.Textbox(label="Suspect Document"),
        gr.Textbox(label="Corpus Documents (one document per line or separated by a delimiter)")
    ],
    outputs=[
        gr.Textbox(label="TF-IDF Results"),
        gr.Textbox(label="SBERT Results")
    ],
    title="Plagiarism Checker"
)
```

4.2 Launching the GUI

Show the code used to launch the Gradio interface, making it accessible via a local or public URL.


Plagiarism Checker

Suspect Document

I believe artificial intelligence will be very important for the future of medicine.

Corpus Documents (one document per line)

I believe artificial intelligence will be very important for the future of medicine.
Machine learning is a method of data analysis that automates analytical model building.
   Deep learning is part of a broader family of machine learning methods based on artificial neural networks.
   "Natural language processing (NLP) is a field of computer science and artificial intelligence.

TF-IDF Results

```
TF-IDF Scores: [1.       0.       0.05082943 0.14972549]
Max TF-IDF Score: 1.0000
```

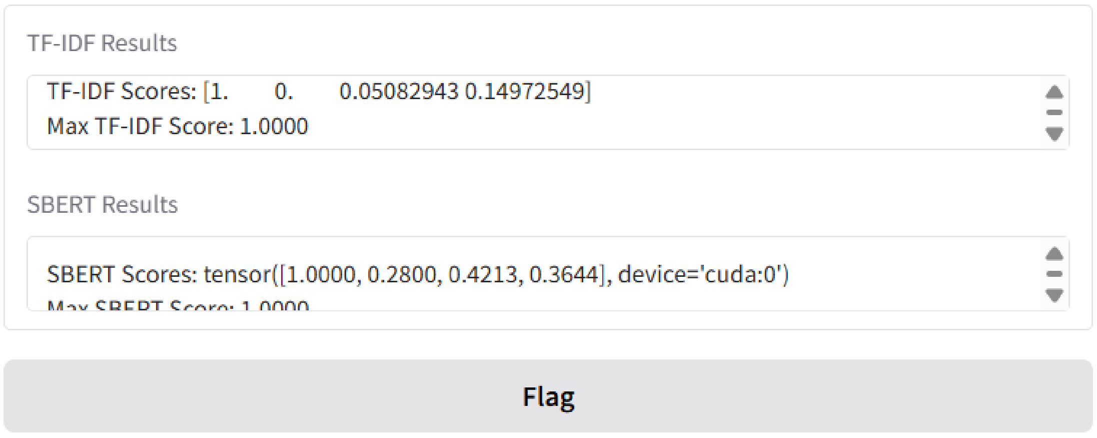
SBERT Results

```
SBERT Scores: tensor([1.0000, 0.2800, 0.4213, 0.3644], device='cuda:0')
Max SBERT Score: 1.0000
```

5. Testing

Describe the testing process used to verify the functionality of the plagiarism detection tool and the GUI.
5.1 Test Cases
Outline the different test cases used, such as:

- Checking an original document against the corpus.
- Checking a direct copy of a corpus document.
- Checking a paraphrased version of a corpus document.


TF-IDF Results

```
TF-IDF Scores: [1.       0.       0.05082943 0.14972549]
Max TF-IDF Score: 1.0000
```

SBERT Results

```
SBERT Scores: tensor([1.0000, 0.2800, 0.4213, 0.3644], device='cuda:0')
Max SBERT Score: 1.0000
```

6. Conclusion and Future Improvements

Summarize the key findings of the project and discuss potential areas for future improvement, such as:

- Adding support for different input formats (file uploads).
- Implementing progress indicators for long processing times.
- Exploring other plagiarism detection techniques.
- Improving the user interface and user experience.

7. Appendices

- **Data Sources:** If you used any specific datasets for testing or training (beyond the examples in the notebook), list them here and provide links if possible.
- **References:** Cite any papers, articles, tutorials, or libraries that were particularly helpful in developing your project. This could include the original papers on TF-IDF or SBERT, or documentation for the libraries you used (NLTK, scikit-learn, sentence-transformers, Gradio).