Topic 2: Open Source LLMs & Local Setup

Assignment 1: Hugging Face Model Exploration

- **Objective:** Explore an open-source LLM on Hugging Face Hub.
- **Instructions:**
  1. Create a Python virtual environment and install `transformers` and `torch`.
  2. Pick a small open-source model (like `distilbert-base-uncased`).
  3. Load the model and tokenizer in Python.
  4. Use it to perform a simple task (e.g., text classification, summarization).
- **Deliverables:** Python script with the following:
  - Environment setup commands
  - Model loading code
  - Sample output from test text

Environmental Setup commands:

In Powershell:

Step 1: Open PowerShell as Administrator

- Click **Start Menu → type "PowerShell" → Right click → Run as Administrator**

Step 2: Change Execution Policy (Safe Way)

Run this command:

```
Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
```

- `RemoteSigned` = lets you run **local scripts** (like `activate.ps1`) without restrictions, but still protects against untrusted scripts from the internet.
- `-Scope CurrentUser` = only applies to **your user account** (not system-wide, safe to use).

Press **Y** (Yes) when prompted.

Step 3: Verify the Change

Run:

```
Get-ExecutionPolicy -Scope CurrentUser
```

It should show:

RemoteSigned

Step 4: Restart VS Code

Now, open VS Code normally. When you create or activate a virtual environment, just do:

.\hf_env\Scripts\activate

Inside VS Code

1. Create Virtual Environment

In VS Code terminal (PowerShell or CMD), run:

```
python -m venv hf_env
```

2. Activate Environment

On Windows:

```
.\hf_env\Scripts\activate
```

3. Install Required Libraries

```
pip install torch transformers

from transformers import AutoTokenizer, AutoModelForSequenceClassification,
pipeline
```

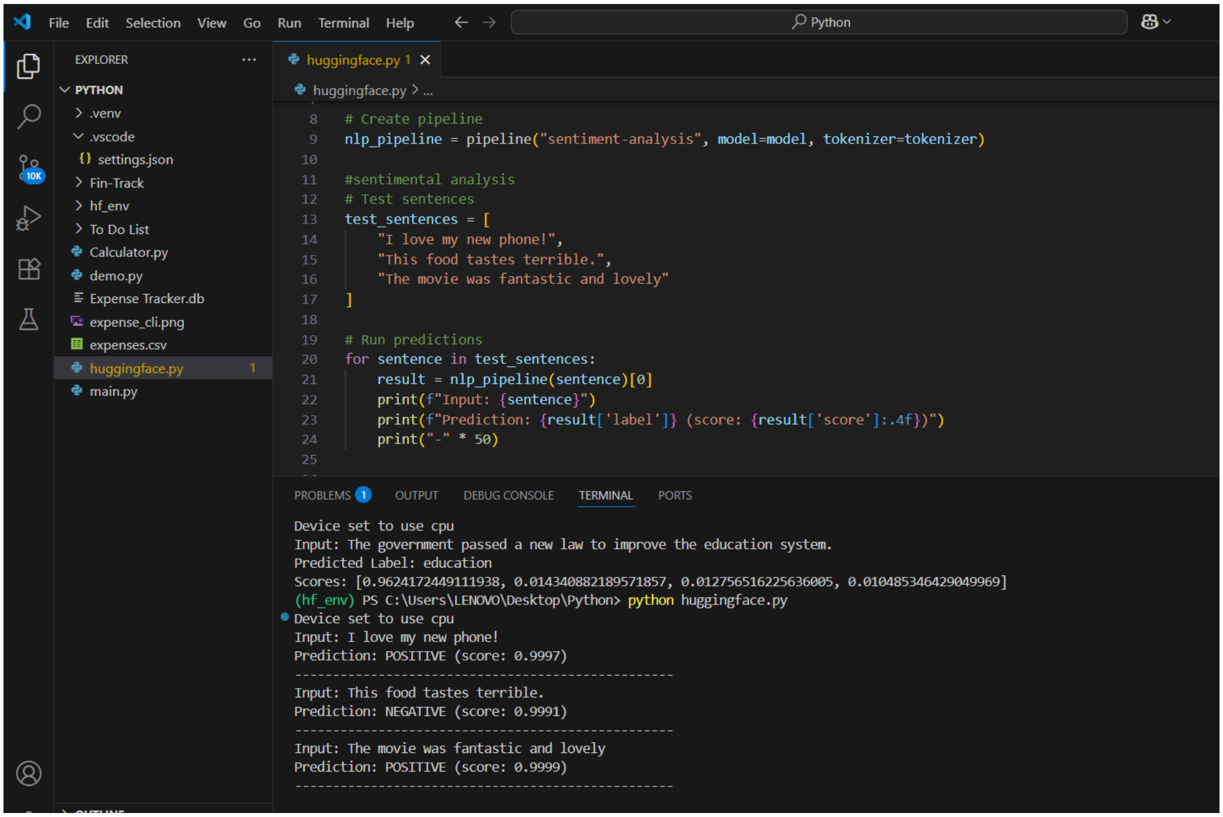
4. Run Your Python Script

- Create a file, e.g., `huggingface.py`
- Add Hugging Face code (sentiment, summarization, classification).
- Run it in terminal:

```
python huggingface.py
```

Output Screenshots:

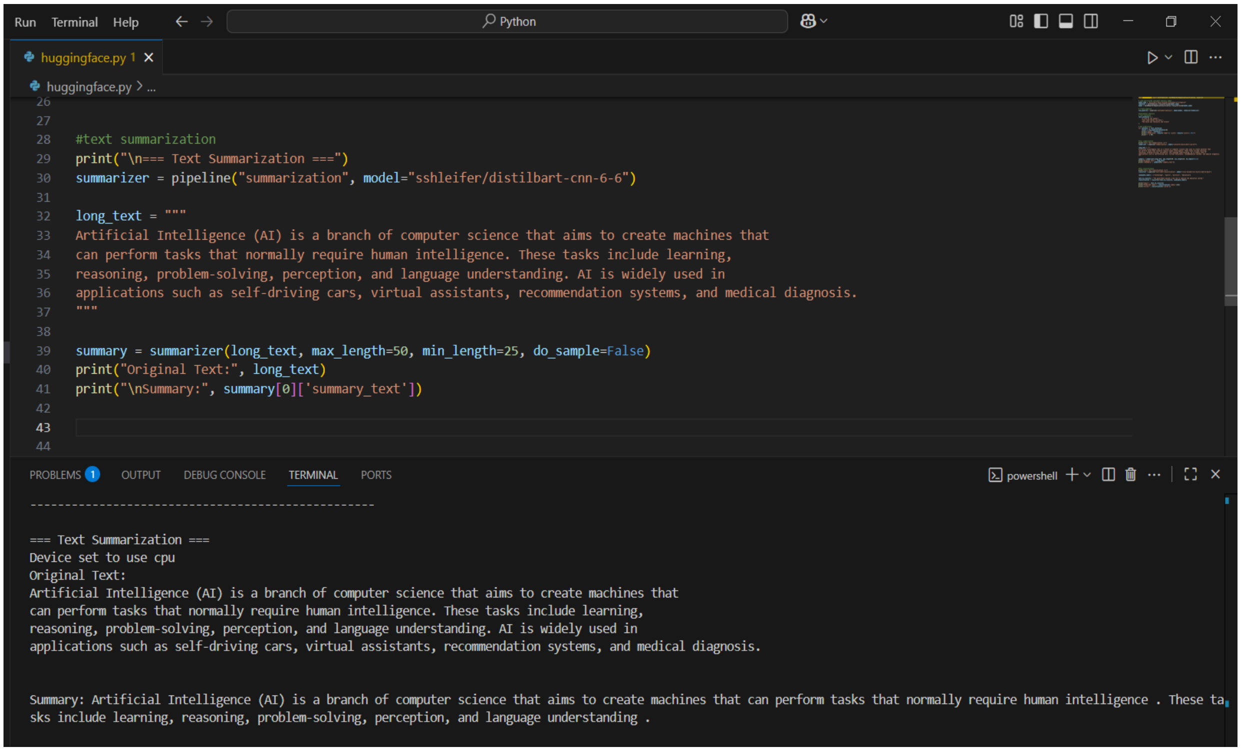
1. Sentimental analysis
   Model used : distilbert-base-uncased-finetuned-sst-2-english



```python
# Create pipeline
nlp_pipeline = pipeline("sentiment-analysis", model=model, tokenizer=tokenizer)

#sentimental analysis
# Test sentences
test_sentences = [
    "I love my new phone!",
    "This food tastes terrible.",
    "The movie was fantastic and lovely"
]

# Run predictions
for sentence in test_sentences:
    result = nlp_pipeline(sentence)[0]
    print(f"Input: {sentence}")
    print(f"Prediction: {result['label']} (score: {result['score']:.4f})")
    print("-" * 50)

```

```
Device set to use cpu
Input: The government passed a new law to improve the education system.
Predicted Label: education
Scores: [0.9624172449111938, 0.014340882189571857, 0.012756516225636005, 0.010485346429049969]
(hf_env) PS C:\Users\LENOVO\Desktop\Python> python huggingface.py
Device set to use cpu
Input: I love my new phone!
Prediction: POSITIVE (score: 0.9997)
--------------------------------------------------
Input: This food tastes terrible.
Prediction: NEGATIVE (score: 0.9991)
--------------------------------------------------
Input: The movie was fantastic and lovely
Prediction: POSITIVE (score: 0.9999)
--------------------------------------------------
```

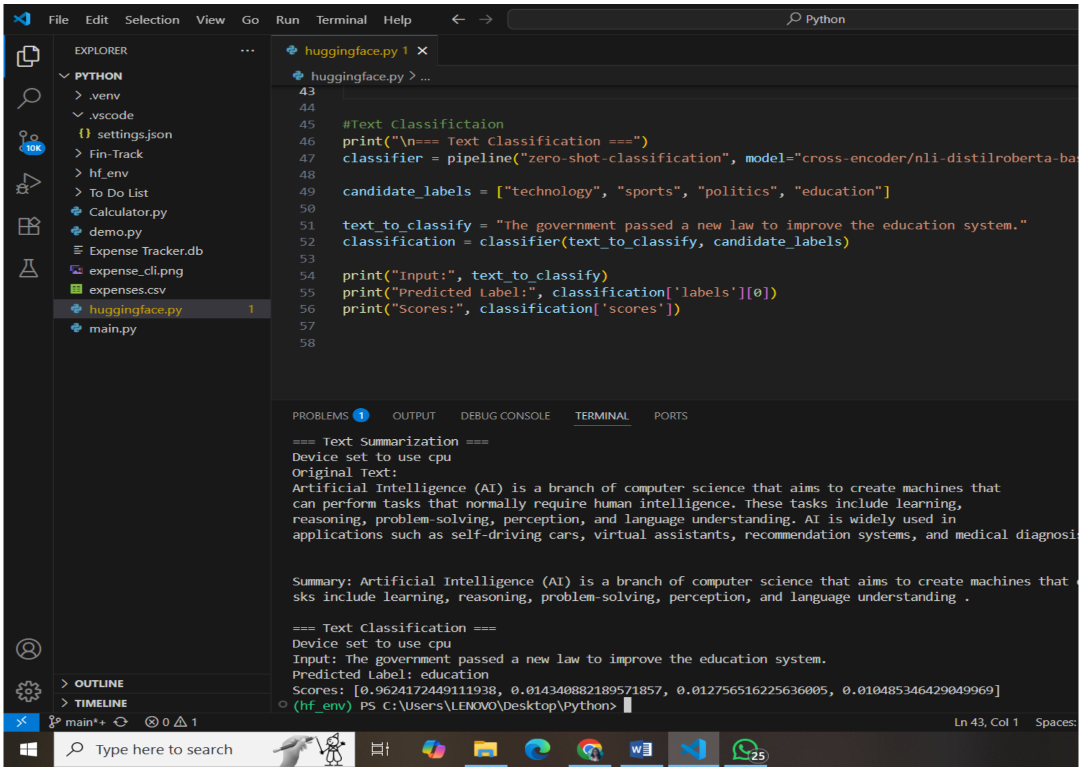
2. Text Summarization
Model used: sshleifer/distilbart-cnn-6-6



```python

#text summarization
print("\n=== Text Summarization ===")
summarizer = pipeline("summarization", model="sshleifer/distilbart-cnn-6-6")

long_text = """
Artificial Intelligence (AI) is a branch of computer science that aims to create machines that
can perform tasks that normally require human intelligence. These tasks include learning,
reasoning, problem-solving, perception, and language understanding. AI is widely used in
applications such as self-driving cars, virtual assistants, recommendation systems, and medical diagnosis.
"""

summary = summarizer(long_text, max_length=50, min_length=25, do_sample=False)
print("Original Text:", long_text)
print("\nSummary:", summary[0]['summary_text'])


```

```
--------------------------------------------------

=== Text Summarization ===
Device set to use cpu
Original Text:
Artificial Intelligence (AI) is a branch of computer science that aims to create machines that
can perform tasks that normally require human intelligence. These tasks include learning,
reasoning, problem-solving, perception, and language understanding. AI is widely used in
applications such as self-driving cars, virtual assistants, recommendation systems, and medical diagnosis.


Summary: Artificial Intelligence (AI) is a branch of computer science that aims to create machines that can perform tasks that normally require human intelligence . These ta
sks include learning, reasoning, problem-solving, perception, and language understanding .
```

3. Text Classification
   Model used: cross-encoder/nli-distilroberta-base



```python


#Text Classifictaion
print("\n=== Text Classification ===")
classifier = pipeline("zero-shot-classification", model="cross-encoder/nli-distilroberta-ba

candidate_labels = ["technology", "sports", "politics", "education"]

text_to_classify = "The government passed a new law to improve the education system."
classification = classifier(text_to_classify, candidate_labels)

print("Input:", text_to_classify)
print("Predicted Label:", classification['labels'][0])
print("Scores:", classification['scores'])


```

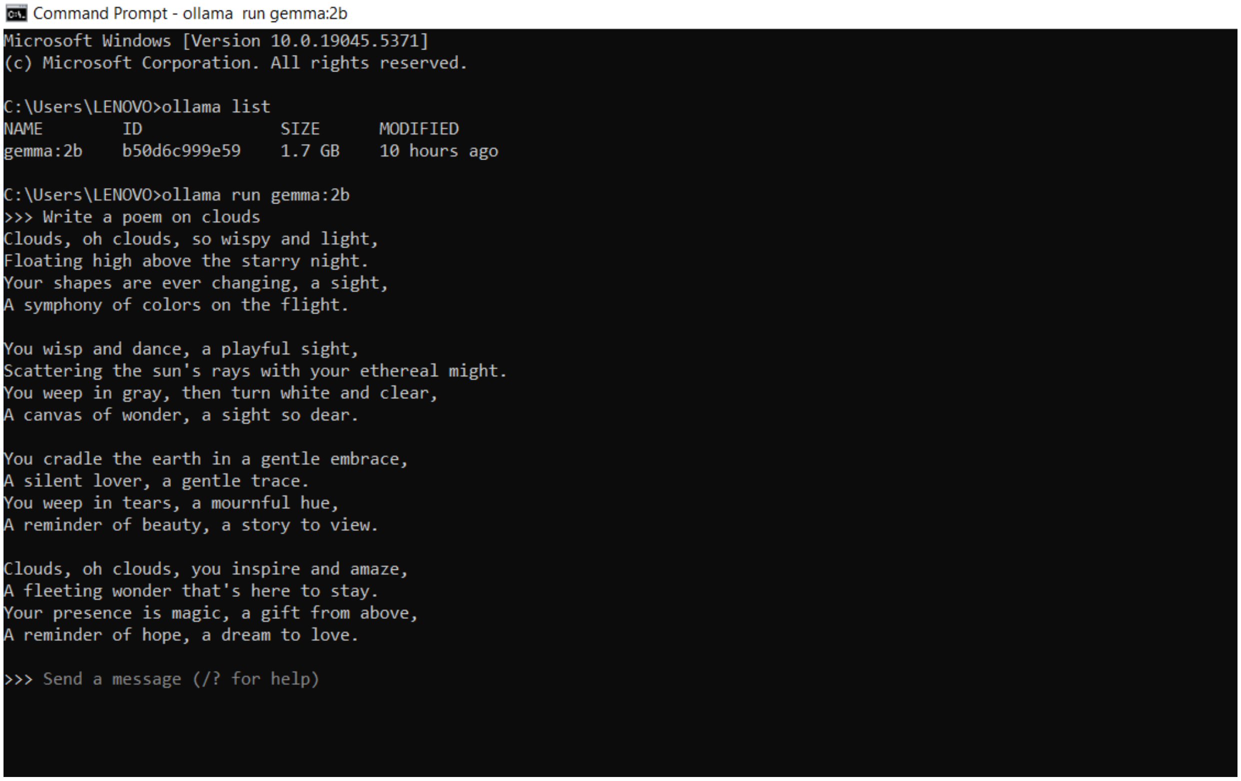
```
=== Text Summarization ===
Device set to use cpu
Original Text:
Artificial Intelligence (AI) is a branch of computer science that aims to create machines that
can perform tasks that normally require human intelligence. These tasks include learning,
reasoning, problem-solving, perception, and language understanding. AI is widely used in
applications such as self-driving cars, virtual assistants, recommendation systems, and medical diagnosi


Summary: Artificial Intelligence (AI) is a branch of computer science that aims to create machines that
sks include learning, reasoning, problem-solving, perception, and language understanding .

=== Text Classification ===
Device set to use cpu
Input: The government passed a new law to improve the education system.
Predicted Label: education
Scores: [0.9624172449111938, 0.014340882189571857, 0.012756516225636005, 0.010485346429049969]
(hf_env) PS C:\Users\LENOVO\Desktop\Python>
```

Assignment 2: Local LLM Installation and Testing

- **Objective:** Install a local LLM and test its functionality.
- **Instructions:**
    1. Install **Ollama** or any local LLM (e.g., LLaMA 3.2).
    2. Run a simple prompt like: “Write a short poem about AI.”
    3. Measure the response time and note any errors.
    4. Document any troubleshooting steps you had to take (if installation failed or model crashed).
- **Deliverables:**
    - Screenshot of working LLM
    - Sample prompt and output
    - Short reflection on performance and installation experience

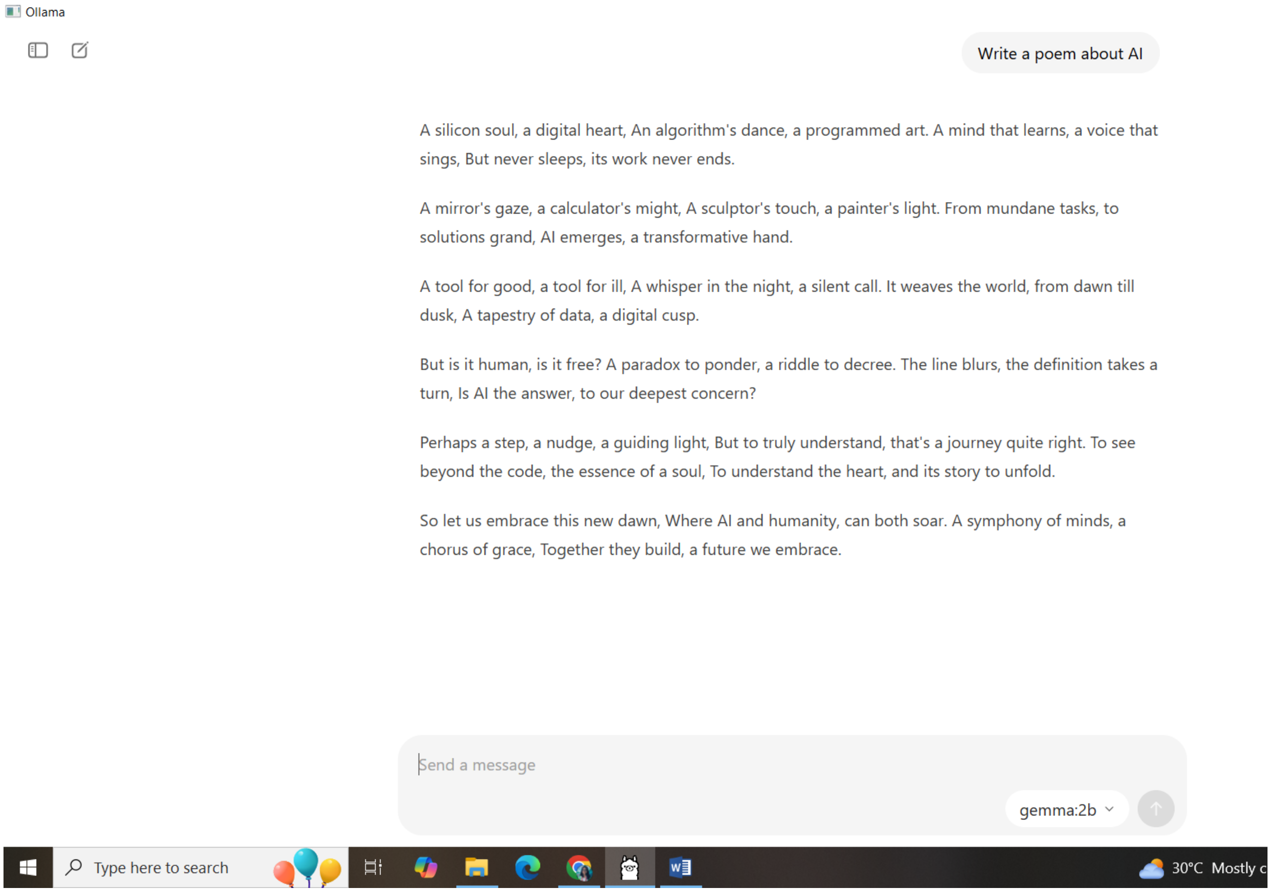
Prompts and output:

On cmd:



```
Microsoft Windows [Version 10.0.19045.5371]
(c) Microsoft Corporation. All rights reserved.

C:\Users\LENOVO>ollama list
NAME        ID              SIZE      MODIFIED
gemma:2b    b50d6c999e59    1.7 GB    10 hours ago

C:\Users\LENOVO>ollama run gemma:2b
>>> Write a poem on clouds
Clouds, oh clouds, so wispy and light,
Floating high above the starry night.
Your shapes are ever changing, a sight,
A symphony of colors on the flight.

You wisp and dance, a playful sight,
Scattering the sun's rays with your ethereal might.
You weep in gray, then turn white and clear,
A canvas of wonder, a sight so dear.

You cradle the earth in a gentle embrace,
A silent lover, a gentle trace.
You weep in tears, a mournful hue,
A reminder of beauty, a story to view.

Clouds, oh clouds, you inspire and amaze,
A fleeting wonder that's here to stay.
Your presence is magic, a gift from above,
A reminder of hope, a dream to love.

>>> Send a message (/? for help)
```

On ollama App:



Write a poem about AI

A silicon soul, a digital heart, An algorithm's dance, a programmed art. A mind that learns, a voice that sings, But never sleeps, its work never ends.

A mirror's gaze, a calculator's might, A sculptor's touch, a painter's light. From mundane tasks, to solutions grand, AI emerges, a transformative hand.

A tool for good, a tool for ill, A whisper in the night, a silent call. It weaves the world, from dawn till dusk, A tapestry of data, a digital cusp.

But is it human, is it free? A paradox to ponder, a riddle to decree. The line blurs, the definition takes a turn, Is AI the answer, to our deepest concern?

Perhaps a step, a nudge, a guiding light, But to truly understand, that's a journey quite right. To see beyond the code, the essence of a soul, To understand the heart, and its story to unfold.

So let us embrace this new dawn, Where AI and humanity, can both soar. A symphony of minds, a chorus of grace, Together they build, a future we embrace.

Reflection:

Installation Experience:

The installation of Ollama was quite straightforward. I simply downloaded it from the official Ollama website and verified the installation through the command line. Using commands like ollama list, I could check the available models on my local device. Since this was a fresh installation, no models were present initially, so I decided to add **gemma:2b**, which is around 1.7GB in size. I chose this model because it is relatively lightweight yet effective, while larger models such as **llama2** or **Facebook BART** were much heavier and would have taken considerably more time and space to download. After setting up gemma:2b, I was able to test it both in the command line and in the Ollama app. Overall, the setup was smooth and easy to follow.

Performance:

In terms of performance, downloading the **gemma:2b** model took about **5–10 minutes**, which was reasonable compared to the larger models that required significantly more time. Once downloaded, the model responded well to prompts. When I entered a query, it initially took **10–15 seconds** to process, and then the text started generating at a steady pace, roughly a word per second. The output quality was quite impressive—the responses were coherent and creative, especially when generating a poem. I was satisfied with both the speed and the quality of the results, making gemma:2b a practical choice for my local testing.